Closed Terminologies and Temporal Reasoning in Description Logic for Concept and Plan Recognition

Robert Anthony Weida

Technical Report CUCS-027-96

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

1996

ABSTRACT

Closed Terminologies and Temporal Reasoning in Description Logic for Concept and Plan Recognition

Robert Anthony Weida

Description logics are knowledge representation formalisms in the tradition of frames and semantic networks, but with an emphasis on formal semantics. A *terminology* contains descriptions of concepts, such as UNIVERSITY, which are automatically classified in a taxonomy via subsumption inferences. Individuals such as COLUMBIA are described in terms of those concepts. This thesis enhances the scope and utility of description logics by exploiting new completeness assumptions during problem solving and by extending the expressiveness of descriptions.

First, we introduce a predictive concept recognition methodology based on a new *closed terminology assumption* (CTA). The terminology is dynamically partitioned by modalities (necessary, optional, and impossible) with respect to individuals as they are specified. In our interactive configuration application, a user incrementally specifies an individual computer system and its components in collaboration with a configuration engine. Choices can be made in any order and at any level of abstraction. We distinguish between abstract and concrete concepts to formally define when an individual's description may be considered finished. We also exploit CTA, together with the terminology's subsumption-based organization, to efficiently track the types of systems and components consistent with current choices, infer additional constraints on current choices, and appropriately restrict future choices. Thus, we can help focus the efforts of both user and configuration engine. This work is implemented in the K-REP system.

Second, we show that a new class of complex descriptions can be formed via constraint networks over standard descriptions. For example, we model plans as

constraint networks whose nodes represent actions. Arcs represent qualitative and metric temporal constraints, plus co-reference constraints, between actions. By combining terminological reasoning with constraint satisfaction techniques, subsumption is extended to constraint networks, allowing automatic classification of a plan library. This work is implemented in the T-REX system, which integrates and builds upon an existing description logic system (K-REP or CLASSIC) and temporal reasoner (MATS).

Finally, we combine the preceding, orthogonal results to conduct predictive recognition of constraint network concepts. As an example, this synthesis enables a new approach to deductive plan recognition, illustrated with travel plans. This work is also realized in T-REX.

Contents

Table of Contents **i**

List of Figures **vii**

List of Tables **xi**

Acknowledgements **xii**

1 Introduction **1**

1.1 Background 1

1.2 Motivation 2

1.3 Problem Statement 4

1.4 Proposed Solutions 4

1.5 Example 8

1.6 Organization of Thesis 12

2 Foundations **15**

2.1 Description Logic 15

2.1.1 Overview 16

2.1.2 Language 18

2.1.3 Principal Inferences 25

2.1.4 Subsumption 27
2.1.5 Recognition 31
2.1.6 Classification 32
2.1.7 Consistency and Disjointness 32
2.1.8 Least Common Subsumer 34
2.1.9 Current Limitations 35
2.2 Closed World Reasoning 36
2.3 Temporal Reasoning 37
2.3.1 Qualitative Constraints 38
2.3.2 Metric Constraints 41
2.3.3 Integration of Qualitative and Metric Constraints 42
2.4 Synthesis 44

3 Predictive Concept Recognition **45**
3.1 Introduction 45
3.2 Incremental Instantiation 49
3.3 The Configuration Problem 56
3.4 Assumptions and Goals 57
3.5 Closed Terminology Consistency 60
3.6 Terminology Augmentation 67
3.7 Recognition via Terminology Partitioning 70
3.8 Constraint Derivation 79
3.9 Open vs. Closed Terminologies 81
3.10 Performance Analysis 83
3.11 Conclusion 86

4 Constraint Networks in Description Logic **94**

4.1 Introduction 94

4.2 T-REX System Overview 95

4.3 Constraint Network Descriptions 96

4.3.1 QME Constraint Network Concepts 98

4.3.2 Plan Concepts 103

4.3.3 Plan Individuals 107

4.3.4 Network Completion 107

4.3.5 Macros 112

4.3.6 Domain Theory 115

4.3.7 Conclusion 117

4.4 Constraint Network Subsumption 117

4.4.1 Node Subsumption 118

4.4.2 Arc Subsumption 118

4.4.3 Network Subsumption 121

4.4.4 Complexity (theory) 125

4.4.5 Computation (practice) 128

4.4.6 Comparison with Standard Description Logic 133

4.4.7 Conclusion 135

4.5 Empirical Analysis 135

4.6 Constraint Network Classification 154

4.7 Recognition 154

4.8 Conclusion 156

5 Predictive Recognition of Constraint Network Concepts **158**

5.1 Introduction 158

5.2 Plan Recognition . . . 159

5.3 Incremental Instantiation . . . 160

5.4 Assumptions and Goals . . . 164

5.5 Constraint Network Consistency . . . 167

5.5.1 Introduction . . . 167

5.5.2 Node Consistency . . . 168

5.5.3 Arc Consistency . . . 168

5.5.4 Direct Network Consistency . . . 171

5.5.5 Indirect Consistency in a Closed Library . . . 176

5.6 Monotonic Observation . . . 180

5.7 Library Augmentation . . . 189

5.8 Recognition via Library Partitioning . . . 193

5.9 Perfect Observation . . . 197

5.10 Unrestricted Observation . . . 203

5.11 Simultaneous Recognition . . . 204

5.12 Active Recognition . . . 205

5.13 Comparison with Predictive Recognition in Standard Description Logic . . . 207

5.14 Conclusion . . . 208

6 Related Work **209**

6.1 Configuration via Description Logic . . . 210

6.1.1 BEACON . . . 210

6.1.2 MESON . . . 211

6.1.3 PROSE . . . 211

6.2 Recognition in Description Logic . . . 213

6.3 Plans in Description Logic 213
6.3.1 SUDO-PLANNER 214
6.3.2 CLASP (Devanbu and Litman) 215
6.3.3 RAT 219
6.3.4 Implemented Plan Subsumption Systems 221
6.3.5 Artale and Franconi 221
6.3.6 Varieties of Plan Subsumption 224
6.4 Temporal Concepts 225
6.4.1 Schmiedel 225
6.4.2 Bettini 227
6.4.3 T-LITE 228
6.5 Production Rules in Description Logics 228
6.6 Deductive Plan Recognition 229
6.6.1 Kautz 229
6.6.2 Vilain 234
6.6.3 Song and Cohen 236
6.7 Candidate Elimination 238
6.8 Intentional Plan Recognition 240

7 Conclusion **241**

7.1 Contributions 241
7.1.1 Predictive Recognition in Description Logic 241
7.1.2 Constraint Networks and Plans 244
7.1.3 Constraint Network Recognition and Plan Recognition . . . 246
7.1.4 Implementation of Plan-based Reasoning 247
7.2 Limitations 248

7.3 Future Work 248

7.3.1 Predictive Recognition in Description Logic 248

7.3.2 Constraint Networks and Plans 251

7.3.3 Constraint Network Recognition and Plan Recognition . . . 255

7.3.4 Generalization to N-Dimensional Space 256

7.4 Summation 259

Bibliography **260**

A Extended Plan Recognition Example **271**

B Proofs of Theorems **296**

Index **321**

List of Figures

1.1 Sample T-REX Plans 9

2.1 Description Logic Syntax and Semantics 19

2.2 Sample Definitions 20

2.3 Sample Concept Taxonomy 21

2.4 Normalized and Completed Descriptions 29

2.5 Allen's Qualitative Relations 39

3.1 Configuration System Architecture 46

3.2 Abstract and Concrete Concepts 50

3.3 Abstract and Concrete Concepts with Individuals 51

3.4 Example of a Finished Individual 55

3.5 Terminology Augmentation 68

3.6 Initial, Intermediate, and Final Recognition States (top to bottom) 71

3.7 MGO Concept Not Subsumed By Any MSN Concept 75

3.8 Specific MGO Concept Not Subsumed By Any MSN Concept . . . 75

3.9 Example Concept Taxonomy 77

3.10 Configuration Concept Taxonomy 83

3.11 Concepts Visited During MSN Search (Non-Cumulative) 88

3.12 Concepts Visited During MSN Search (Cumulative) 88

3.13 Instantiation Tests (Non-Cumulative) 89
3.14 Instantiation Tests (Cumulative) 89
3.15 Concepts Visited During MSO Search (Non-Cumulative) 90
3.16 Concepts Visited During MSO Search (Cumulative) 90
3.17 Direct Consistency Tests (Non-Cumulative) 91
3.18 Direct Consistency Tests (Cumulative) 91
3.19 Concepts Visited During MGO Search (Non-Cumulative) 92
3.20 Concepts Visited During MGO Search (Cumulative) 92
3.21 Predictive Recognition Time (Non-Cumulative) 93
3.22 Predictive Recognition Time (Cumulative) 93

4.1 The T-REX System 97
4.2 Network with Combined Constraints 103
4.3 Sample Plan Networks 106
4.4 Sample Plan Individual 108
4.5 Equality Constraint for Completion 110
4.6 Specializing a Macro Plan 113
4.7 Plan with Macro Steps 114
4.8 Subsumption Mapping 122
4.9 Multiple Subsumption Mappings 122
4.10 Non-Structural Plan Subsumption 126
4.11 Subsumee Node Count Experiment A 144
4.12 Subsumee Node Count Experiment B 145
4.13 Subsumee Distinct Count Experiment A 146
4.14 Subsumee Distinct Count Experiment B 147
4.15 Subsumee Sequence Count Experiment A 148

4.16 Subsumee Sequence Count Experiment B 149
4.17 Subsumer Node Count Experiment A 150
4.18 Subsumer Node Count Experiment B 151
4.19 Subsumer Distinct Count Experiment A 152
4.20 Subsumer Distinct Count Experiment B 153

5.1 Revised Observations 161
5.2 Structural Consistency Mapping 172
5.3 Non-Structural Consistency Mapping 174
5.4 Indirectly CTA-consistent Plans 177
5.5 Small Travel Plan Library 182
5.6 Small Travel Plan Taxonomy 183
5.7 Small Travel Action Concept Taxonomy 183
5.8 Library Augmentation 189
5.9 Plan Library Augmentation 190
5.10 Computing Modalities in an Augmented Library 194
5.11 PLAN-B (repeated from Figure 1.1) 198
5.12 Perfectly Observed Plan Individual 199

6.1 CLASP Actions (from [Devanbu and Litman, 1991]) 216
6.2 CLASP Plans (from [Devanbu and Litman, 1991]) 217
6.3 Plan Hierarchy (after [Kautz, 1991b]) 230
6.4 A Kautz Plan (from [Kautz, 1991b]) 232

A.1 Specifying Qualitative Relations for MATS (from [Kautz, 1991a]) . 273
A.2 Travel Plan Definitions 274
A.2 Travel Plan Definitions (continued) 275

A.2 Travel Plan Definitions (continued) 276

A.2 Travel Plan Definitions (continued) 277

A.2 Travel Plan Definitions (continued) 278

A.2 Travel Plan Definitions (continued) 279

A.2 Travel Plan Definitions (continued) 280

A.2 Travel Plan Definitions (conclusion) 281

A.3 Travel Plan Taxonomy 282

A.4 Travel Concept Taxonomy 283

A.5 Travel Plan Rules 284

A.6 Travel Plan Demons 285

B.1 An Illustration of Directly CTA-consistent Networks 312

B.2 An Illustration of Indirectly CTA-consistent Networks 313

List of Tables

6.1 Implemented Plan-based Systems in Description Logic 221

Acknowledgements

First of all, I am profoundly indebted to Diane Litman for her thoughtful, expert advice throughout the development of this thesis. Her keen technical insight and superb editorial acumen have added immeasurably to the document you are now reading. On a personal level, I want to say that Diane's continuing dedication to supervising my research on a volunteer basis after returning to Bell Labs reflects her exemplary character and is much appreciated. Eric Mays has been a good friend and colleague, as well as a stimulating mentor, since I first began working as a summer student in the Mathematical Sciences Department at the IBM T. J. Watson Research Center. Eric is the guiding force behind the K-Rep project, which is the foundation for much of this work. As manager of the Knowledge Representation Principles and Practices Group, he has consistently demonstrated outstanding personal and intellectual leadership, and has earned my everlasting respect. Kathy McKeown deserves very special thanks for patiently supporting me throughout my stay at Columbia, and for introducing me to the world of research. My ongoing interest in natural language processing is due to her influence. The other members of my committee, Steve Feiner and Sal Stolfo, also contributed significantly to this thesis with their questions and careful suggestions.

I am delighted to acknowledge numerous individuals who have generously invested their time to offer valuable comments on various papers which preceded this thesis. They include Alex Borgida, Mukesh Dalal, Prem Devanbu, Bob Dionne, Michael Elhadad, Steffen Fohn, Henry Kautz, Diane Litman, Eric Mays, Deborah McGuinness, Jacques Robin, Eric Siegel, Sal Stolfo, and Bonnie Webber. Furthermore, Jacques Robin and Vasileios Hatzivassiloglou were kind enough to provide feedback on some draft chapters of this thesis. I have also benefited from discussions with Bernhard Nebel and Werner Nutt.

Henry Kautz deserves credit for making his MATS temporal reasoner available to the research community. Likewise, I appreciate the fact that AT&T and IBM made CLASSIC and K-REP, respectively, available at Columbia.

My life at Columbia has been enhanced by a number of engaging and companionable officemates. In 703 CEPSR, I have greatly enjoyed the company of Pascale Fung, Eric Siegel, and earlier, Jacques Robin — the mastermind who recruited me, along with Eric, to join him when the NLP/KR group moved to the new office building. Prior to that, I had the distinct pleasure of sharing 415 CSB for several years with Andrea Danyluk and Nat Polish. My first officemates, Avram Aronoff and the late Russell Mills, were also fine company.

My friends Steve Kearns and David Kurlander were regular dinner partners for many years, until they graduated and reminded me that there really is life after the dissertation. I've enjoyed many discussions about computers and life with Ken and Sharon Roberts, often while bicycling, rock climbing, or skiing. Many other CUCS people merit recognition for enhancing my technical and social life; I regret not being able to mention everyone by name.

My graduate student days were enlightened and enriched by my longstanding association with the IBM T. J. Watson Research Center. My colleagues there have always fostered a challenging and congenial environment for work. In particular, I have been privileged to work with the current members of the K-Rep team: Bob Dionne, Meir Laker, Chihong Liang, Eric Mays, Frank Oles, and Brian White. Earlier, as part of the Expert Systems group, I was equally lucky to work with some of those same people, as well as Chid Apté, Jim Griesmer, Se June Hong, Maurice Karnaugh, and John Kastner.

I will always be grateful to Denis Baggi, for first teaching me about artificial intelligence, and for inspiring me to pursue a Ph.D.

Finally, my family has sustained me throughout my graduate study. My aunt and uncle, Mary Lou and Tom Riethof, have always been terrific. Peter Weida is, in my humble opinion, the world's finest brother. Last, but not least, I am eternally grateful to Elisabeth and Robert L. Weida (affectionately known as Mom and Dad) for their lifelong love and support. In the end, that's more important than anything else.

To my parents, with love.

Chapter 1

Introduction

Description logics comprise a widely studied knowledge representation formalism that is beginning to emerge in commercial applications.[1] This thesis aims to enhance the scope and utility of description logic by extending the range of descriptions that can be expressed and by exploiting new completeness assumptions during problem solving. It draws on ideas from closed world reasoning, constraint satisfaction, deductive plan recognition, and temporal reasoning to better address important applications such as configuration of complex systems and travel planning.

1.1 Background

The field of *knowledge representation and reasoning* encompasses languages for encoding knowledge, inference mechanisms for reaching conclusions from the knowledge, and algorithms to implement the inference mechanisms. Description logic is a branch of knowledge representation and reasoning that focuses on creating descriptions of concepts and individuals with a well-defined language and

[1] Work in this area has gone by many names over the years, including terminological logics, term subsumption systems, structured inheritance networks, and the KL-ONE family.

making logical inferences about their relationship to one another. Informally, concepts like UNIVERSITY describe sets of possible instances, which are individuals such as COLUMBIA. Fundamental inferences in description logic include subsumption, classification, and recognition. One concept *subsumes* another when the first is a generalization of the second. Subsumption is usually given an extensional semantics, where concept *C1* subsumes concept *C2* if and only if the set of possible individuals described by *C1* is a superset of those described by *C2*. In practice, subsumption is computed intensionally by comparing the descriptions of *C1* and *C2*. *Classification* integrates a new concept within an explicit concept taxonomy, such that each concept subsumes its descendants and is subsumed by its ancestors, without exception. *Recognition* determines whether an individual *instantiates* a particular concept. Instantiation occurs when the individual is a member of the concept's extension. (A significant part of this thesis seeks to transcend traditional recognition of individuals based on their current description, by enabling *predictive recognition* based on possible future descriptions.) Classification of an individual determines the set of most specific concepts in the taxonomy that the individual instantiates.

The set of concepts explicitly defined for some application is known as a *terminology*. We will sometimes refer to a terminology as a *concept taxonomy*, or just *taxonomy*, to emphasize its subsumption-based organization. We will refer to the set of explicitly defined concepts and individuals together as a *knowledge base*.

1.2 Motivation

An interest in applying description logic to configuration of complex systems, plan recognition, and other important applications led us to address certain limitations of contemporary description logic systems.

First, description logic systems make uniform assumptions that fail to account for important differences between knowledge engineering and problem solving activities. One assumption has to do with the completeness of the terminology. Description logic systems traditionally assume an "open" terminology, i.e., that there may be relevant concepts not explicitly defined therein. Conversely, they are incapable of "closed world" reasoning about the set of concepts in a terminology, despite the prominence of closed world reasoning in other areas of knowledge representation. For some important applications, such as system configuration, the terminology is in fact closed while solving a particular problem. Failure to take advantage of this fact means that significant conclusions may go unnoticed. Another assumption involves the finality of individual descriptions. Many description logic systems permit the description of an individual to be updated incrementally; when the individual's description is updated, they revise its classification accordingly. However, at any moment they operate as if the individual's description is final and only identify the concepts that it currently instantiates. Present-day description logic systems neither examine whether the description is plausibly finished, nor do they contemplate how it might be revised in the future. That is, they do not adequately determine *potential* instantiations. Ameliorating these limitations would be extremely valuable for applications such as system configuration and plan recognition.

Second, most extant description logic languages are intended to describe single concepts and individuals according to their relationship with other concepts and individuals. With very few exceptions, description logics do not support concept aggregation, i.e, the formation of complex descriptions composed from collections of peer concepts. Likewise, they do not support aggregation of individuals. Consider (individual) plans, which may be viewed in part as patterns of temporally related (individual) actions. Notwithstanding the prevalence of plan-based reasoning in artificial intelligence, including both plan generation and plan recognition, very few description logic systems can describe the temporal aspects of plans. None can approach the expressiveness of today's powerful temporal reasoning systems, e.g., MATS [Kautz and Ladkin, 1991].

The limitations we have just described motivate this thesis. They may be crystallized in the following problem statement.

1.3 Problem Statement

This thesis addresses three principal questions:

1. Can we exploit a closed terminology assumption in description logic, i.e., an assumption that the terminology contains every concept of interest, to make stronger inferences?

2. Can we extend traditional description logic with complex descriptions composed of multiple concepts (and likewise multiple individuals) and constraints among them, including temporal patterns of events, e.g., plans?

3. Do the preceding questions admit compatible solutions, so we can also take advantage of a closed terminology assumption when the terminology consists of complex descriptions such as plans?

1.4 Proposed Solutions

This thesis answers the three principal questions as follows:

1. **Closed terminology reasoning in description logic can be exploited for predictive concept recognition and applied to problems such as system configuration.**

   We introduce a distinction between the knowledge engineering and problem solving phases of terminology usage in description logic, for those applications where the terminology remains fixed during problem solving. In a radical departure from previous description logic work, we demonstrate that a *closed terminology assumption* during problem solving enables useful inferences that would not be possible otherwise. Essentially, our closed terminology assumption means that all relevant concepts from the domain of interest are explicitly defined, and that every individual, once fully specified, will correspond directly to at least one concept.[2] Chapter 3 will precisely define the notion of "direct correspondence." This assumption has several important benefits. First, we use closed terminology reasoning to *predictively* recognize concepts while they are being instantiated. That is, given a partial description of an individual, we ascertain which concepts it already instantiates, which it might eventually instantiate, and which it cannot instantiate. If the individual's description is updated, we revise our findings accordingly. Second, we leverage these findings to extract implicit constraints on the individual from the closed terminology. In particular, if every concept that is consistent with the individual has certain properties, then those properties can be attributed to the individual by virtue of the closed terminology. This entire process applies recursively to other individuals referred to in a given individual's description.

Consider the problem of system configuration, where description logic has already found notable success [Wright *et al.*, 1993]. The standard *open terminology assumption*[3], which presumes an incomplete terminology, is appropriate during knowledge engineering, when we are still constructing a model of systems, components, functionality, and related concepts. However, during a specific configuration task, closed terminology reasoning is more suitable because all relevant concepts (e.g., types of systems) are known in advance. Let us concentrate on interactive configuration, where the user and the configuration engine collaborate to arrive at a satisfactory result. As choices are made about components and functionality, we can track the types of systems that remain consistent with those choices, and also infer constraints on future choices. Both kinds of information can be utilized to focus the efforts of the user and also the configuration engine. (In fact, our methods

[2] We do not make a closed world assumption over individuals.

[3] We coined the phrase *open terminology assumption* to describe standard practice.

apply equally well to non-interactive usage, with the obvious qualification that focusing the user's effort is not an issue in non-interactive cases; the crucial consideration is that the individual system is being specified incrementally.) For example, if a user requires a certain kind of specialized input device for her computer system, and it happens that the only systems in the terminology which support such a device are made by IBM, then we can (a) limit further consideration to IBM computer systems, and (b) assert that the user's individual system will be an IBM system. Although we stress system configuration in this thesis, our methodology is domain-independent. Some other possible application areas are identified in Section 7.3.

In short, the first major contribution of this thesis is a general-purpose predictive recognition methodology for description logic based on a closed terminology assumption.

2. **Traditional description logic can be extended to represent and reason with constraint networks, including support for temporally expressive plans.**

   Along a different dimension, we show that a new class of complex concepts / individuals can be described via constraint networks where the nodes (unary constraints) are represented by regular description logic concepts / individuals and the arcs (binary constraints) relate the concepts / individuals to each other. We define a general methodology for computing subsumption between constraint network concepts and also propose a specific, implemented algorithm for doing so. As a result, we are able to classify constraint network concepts into a taxonomy. Recognition of individuals is defined as well; we show that it yields to a similar algorithm.

   To make these results concrete, we show how an expressive class of plans and their instances are handled within this framework. In particular, we classify a library of plan descriptions whose constituent actions are in turn described with standard description logic concepts. To avoid confusion, we will speak of a *library* of constraint network concepts (e.g., a *plan library*) as opposed to a *terminology* of standard concepts. Elements of a library are defined in terms of a terminology. For demonstration purposes, we deal with three different sorts of constraints among actions:

(a) Qualitative temporal constraints as disjunctive sets of possibilities, e.g., ACTION1 *before or after* ACTION2.

(b) Metric temporal constraints as numeric intervals, e.g., ACTION2 *within two to three hours* after ACTION1.

(c) Co-reference constraints as equality requirements, e.g., ACTION1 *has the same agent as* ACTION2.

Concisely, the second major contribution of this thesis is an extension to description logic for complex descriptions in the form of constraint networks.

3. **The preceding results can be combined to conduct predictive recognition of constraint network concepts and applied to deductive plan recognition.**

   For synergy, we unify the two previous solutions in an elegant, orthogonal manner to conduct closed terminology reasoning over a taxonomy composed of constraint network concepts. In particular, we predictively recognize instances of plans while they are being observed. Our methods take full advantage of the well-founded semantics for actions and plans, along with the subsumption-based organization of the action and plan taxonomies.

   Therefore, the third major contribution of this thesis is a deductive theory of plan recognition that handles temporally rich plans in the description logic framework.

Beyond the theoretical results just outlined, our work has a substantial practical component. Our ideas about closed terminology reasoning and predictive recognition in standard description logic [Weida, 1996] are implemented as an extension to the K-REP system [Mays *et al.*, 1991a]. Reasoning with plan descriptions and predictive plan recognition are both implemented in the T-REX system [Weida and Litman, 1992; Weida and Litman, 1994; Weida, 1995b]. To date, T-REX is the only system to harness a description logic system, a choice of either K-REP or CLASSIC [Borgida *et al.*, 1989], in tandem with a powerful temporal reasoner, MATS [Kautz and Ladkin, 1991].

Further details on the contributions of this thesis are introduced in subsequent chapters and summarized in the conclusion.

1.5 Example

To illustrate the nature of our work on closed terminology reasoning and temporal reasoning in description logic, we now offer a simple sketch of T-REX in operation on a plan recognition problem. A more formal treatment will be developed in Chapters 4 and 5. Through a combination of description logic, temporal constraints, and co-reference constraints, T-REX supports a rich, well-defined plan language currently focused on *plan bodies*. Each step in a plan body denotes an action and an associated time interval. Steps are related by qualitative temporal constraints between intervals, metric temporal constraints between endpoints of intervals, and co-reference constraints on roles (attributes) of actions. Plan bodies are viewed as constraint networks where nodes correspond to steps and arcs correspond to constraints between steps.

One application where plan recognition could prove useful is travel planning. Well-funded efforts to develop Internet-based travel services are starting to emerge, but substantial challenges exist. The following quote appears in [Markoff, 1996]:

> The hardest part of doing a travel agent's job is not making the reservations, but understanding what the customer wants to do.
> — Richard Shaffer, editor of *Technologic Computerletter*

Several examples of travel plan descriptions appear in Figure 1.1. Action concepts such as ATTEND-WORKSHOP are referenced, but their descriptions are omit-

In PLAN-A, an agent attends a workshop for at most 2880 minutes, either before or after attending a conference.

(define-plan **PLAN-A**
:steps ((step1 ATTEND-WORKSHOP)
(step2 ATTEND-CONFERENCE))
:qualitative-constraints ((step1 (*before* $\vee$ *after*) step2))
:metric-constraints (($0 <$ step1_{finish} $-$ step$1_{start} \leq 2880$))
:co-reference-constraints ((agent(step1) = agent(step2))))

In PLAN-B, an agent visits a city, during which time s/he attends a workshop for between 240 and 480 minutes, then attends an AI conference, all in the same location.

(define-plan **PLAN-B**
:steps ((step1 VISIT-CITY)
(step2 ATTEND-WORKSHOP)
(step3 ATTEND-AI-CONFERENCE))
:qualitative-constraints ((step2 (*during*) step1)
(step3 (*during*) step1)
(step2 (*before*) step3))
:metric-constraints (($240 <$ step2_{finish} $-$ step$2_{start} \leq 480$))
:co-reference-constraints ((agent(step1) = agent(step2) = agent(step3))
(location(step1) = location(step2) = location(step3))))

In PLAN-C, an agent climbs a mountain while in a national park.

(define-plan **PLAN-C**
:steps ((step1 VISIT-NATIONAL-PARK)
(step2 CLIMB-MOUNTAIN))
:qualitative-constraints ((step1 (*contains*) step2))
:co-reference-constraints ((agent(step1) = agent(step2))))

Figure 1.1: Sample T-REX Plans

ted for brevity. Informally, PLAN-A entails two actions: attending a workshop and attending a conference. For identification, they are arbitrarily labeled *step1* and *step2*, respectively. The steps of a plan can be viewed as existential constraints on the occurrence of actions. PLAN-A has several other kinds of constraints:

1. *Qualitative temporal constraints:* Attending the workshop occurs either before or after attending the conference.
2. *Metric temporal constraints:* Attending the workshop consumes no more than two days (2880 minutes).
3. *Co-reference constraints:* The same agent (presumably a person in the relevant concept descriptions) undertakes both steps of the plan.

Within these constraints, PLAN-A can be carried out in numerous ways. For example, the author might attend a two day description logic workshop some unspecified time after he attends a conference on medical informatics.

PLAN-B involves three actions: visiting a city, attending a workshop, and attending an AI conference, along with some constraints among them:

1. *Qualitative temporal constraints:* Attending the workshop occurs strictly before attending the conference and both occur during the visit to the city.
2. *Metric temporal constraints:* Attending the workshop consumes between four and eight hours.
3. *Co-reference constraints:* All three steps are undertaken by the same agent. Furthermore, each step occurs in the same location.

For instance, the author might attend a one-day plan recognition workshop in Montréal just before he attends an AI conference there. Notice that this latter scenario also meets the requirements of PLAN-A. In fact, it can be shown that every scenario that satisfies the description of PLAN-B must also satisfy the description of PLAN-A. Informally, the correspondence can be seen by mapping from *step1* and *step2* of PLAN-A to *step2* and *step3* of PLAN-B, respectively.[4] For this purpose, the step names themselves are inconsequential. PLAN-B specializes PLAN-A by adding a VISIT-CITY step and stipulating that the conference attended is an AI conference. Moreover, the constraints across steps in PLAN-B are more restrictive than their counterparts in PLAN-A:

1. *Qualitative temporal constraints:* Attending the workshop occurs strictly before, and not after attending the conference.

2. *Metric temporal constraints:* The duration of attending the workshop is more tightly bounded.

3. *Co-reference constraints:* The actions must occur in the same location. Also, the agent of the extra VISIT-CITY step must be the same as the agent of the other two steps.

Finally, none of the constraints on PLAN-B contradict their counterparts in PLAN-A under the aforementioned mapping. Hence, T-REX is able to deduce that PLAN-A subsumes PLAN-B, and it classifies PLAN-B underneath PLAN-A in the plan taxonomy. On the other hand, PLAN-C neither subsumes nor is subsumed by either of the other plans. Intuitively, this is because their steps are incomparable.

Now we are ready to portray a simple plan recognition problem. Imagine that we are given reports of events as they occur and suppose we know that all reported events are intended to carry out one or more plans from Figure 1.1. Like many plan recognition systems, e.g., [Kautz, 1991b], T-REX assumes that its plan library is complete. While this is a strong assumption, each plan can describe a wide variety of possible instantiations. Suppose we are initially told that an agent visits Seattle. The concept taxonomy indicates that VISIT-SEATTLE is a subconcept of VISIT-CITY, and PLAN-B is the only plan involving a visit to a city. Therefore,

[4] A view of these plans as constraint networks will be developed in Chapter 4, and the mapping is illustrated in Figure 4.8 on page 122.

T-REX determines that the agent must be following PLAN-B. At a higher level of abstraction, T-REX indirectly discovers that s/he must also be following PLAN-A, even though we have received no direct report of PLAN-A's steps being carried out. Using the principle of Occam's razor, T-REX prefers to explain events by the smallest possible number of plans. Since PLAN-B alone is an adequate explanation for the act of visiting Seattle, PLAN-C is provisionally ruled out. However, if we next learn that the agent is atop Mount Rainier, T-REX will conclude that s/he must be pursuing both PLAN-B and PLAN-C.

Much of our thesis is concerned with formal methods for this sort of reasoning in description logic, plus techniques for implementing it. Our predictive recognition methodology is quite general, as evidenced by its application to such divergent tasks as plan recognition and computer system configuration. If anything, the artificial domain of computer systems is even more amenable to this sort of deductive recognition, and our results for configuration are of more immediate practical value.

1.6 Organization of Thesis

The remainder of the thesis is arranged as follows.

Chapter 2 (Foundations) provides an overview of the key work in knowledge representation that forms the basis for our thesis. First, we introduce description logic as a practical and well-defined tool for modeling conceptual knowledge. This point is illustrated with examples from the domain of computer system configuration. Our presentation covers the syntax and semantics of a description logic language, typical inferences made by a description logic system, and the essential tradeoff that every such system must make between the expressiveness of its language and the computational complexity of its inference procedures. Next, we touch on the notion of closed world reasoning and show how completeness assumptions permit an inference system to draw conclusions that could not be made otherwise. Finally, we summarize work on reasoning with temporal constraints. Attention is devoted to qualitative constraints on temporal intervals, metric constraints on time points, and the relationship between them. We focus on the MATS system [Kautz and Ladkin, 1991], which coordinates separate qualitative and metric temporal constraint systems in a single temporal reasoner whose power exceeds the sum of its parts.

Chapter 3 (Predictive Concept Recognition) presents our methodology for predictive recognition in description logic. Predictive recognition is used during problem solving (when the terminology is used), rather than during knowledge engineering (when the terminology is created). It is based on a closed terminology assumption, along with a provisional assumption of monotonic updates to individuals. This work is motivated by more examples from computer system configuration. We demonstrate how to track the set of concepts that an individual may eventually instantiate as the individual's description is incrementally refined. Thanks to our assumptions, the remaining concepts are provisionally ruled out. We specify criteria to gauge the termination of incremental instantiation, define a set of rules for *consistency* inferences to drive the recognition process, and prove the correctness of those rules. Predictive recognition findings are shown to be informative in their own right. We then exploit them further to derive constraints on individuals that follow from their current descriptions and the limited possibilities admitted by the closed terminology. Our work in this area includes an incremental recognition algorithm that accommodates changes to an individual's description. We analyze the performance of our predictive recognition methodology on a configuration knowledge base of moderate size and conclude that it has practical potential. This work is implemented as an extension to K-REP.

Chapter 4 (Constraint Networks in Description Logic) shows how description logic can be extended with constraint network concepts, where each node corresponds to a standard description logic concept. To motivate this extension, we model plan bodies as constraint networks where the nodes represent types of actions. The arcs represent temporal constraints between actions as well as co-reference constraints among their roles. The ideas of subsumption and classification from description logic are extended naturally to constraint networks. We describe the implementation of this work in the T-REX system, which integrates and builds upon a description logic system (either K-REP or CLASSIC) together with a temporal reasoner (MATS), to manage a library of plan descriptions.

Chapter 5 (Predictive Recognition of Constraint Network Concepts) combines the orthogonal results on predictive concept recognition from Chapter 3, and on constraint network descriptions from Chapter 4, to achieve predictive recognition of constraint network concepts. As a concrete example, we show how this synthesis enables a new approach to deductive plan recognition in the spirit of Kautz's landmark work [Kautz, 1991b], but with significant advantages. Our approach is illustrated with travel plans such as those embodied in package tours offered by travel companies. This work is also realized in T-REX.

Chapter 6 (Related Work) places this thesis in the context of relevant research. We compare and contrast our results with other contributions from the literature of description logic, temporal reasoning, system configuration, machine learning, and plan recognition.

Chapter 7 (Conclusions) begins with a review and summary of our contributions. It closes with an assessment of the work's current limitations as part of a prospectus for future work.

Chapter 2

Foundations

This thesis is founded on work in two distinct areas of artificial intelligence: description logic and temporal reasoning. In a nutshell, our interest in these areas led us to (1) introduce closed terminology reasoning in description logic to obtain a new class of concept recognition inferences, (2) integrate temporal reasoning with description logic to achieve more expressive descriptions, and (3) combine those results in closed terminology reasoning over a library of temporally rich descriptions. This chapter provides requisite background material for the thesis by summarizing relevant ideas from description logic, closed world reasoning, and temporal reasoning, in that order. Other important related work is compared and contrasted with our own work in Chapter 6.

2.1 Description Logic

We begin with a summary of description logic as it relates to our thesis. After an overview, this section presents a core description logic language and summarizes the principal inferences of description logics. Further detail is then provided on the following inferences: subsumption, recognition, classification, consistency and disjointness, and least common subsumer. Limitations of current description logic systems are then discussed.

2.1.1 Overview

Description logics are an object-centered approach to knowledge representation in the tradition of semantic networks and frames. A specific description logic provides a formal language for defining concepts and individuals, along with an inference mechanism for reasoning about them [MacGregor, 1991a; Woods and Schmolze, 1992]. Most description logic languages concentrate on a carefully chosen set of description forming operators intended to strike a favorable balance between expressiveness and performance. Description logic is distinguished from other areas of knowledge representation by its emphasis on formal taxonomic reasoning. Key description logic inferences include subsumption, classification, and recognition. Concept *C1* *subsumes* (is more general than) concept *C2* when every possible instance of *C2* is also an instance of *C1*. Conversely, concept *C2* is said to *specialize* concept *C1*. We write this as C2 $\Rightarrow$ *C1*. Subsumption is reflexive, transitive, and anti-symmetric. Implementations of description logic maintain an explicit concept taxonomy, or terminology, with the property that each concept subsumes its descendants and is subsumed by its ancestors, without exception. The terminology is used to model the entities of interest in some domain. Whenever a new concept is defined, *classification* automatically integrates it into the taxonomy so that its parents are its most specific subsumers and its children are its most general subsumees. One can also describe individuals in description logic. The *recognition* inference determines if individual I is an instance of concept C, written $I \epsilon C$. If so, we say that the individual *instantiates* (matches the description of) the concept. Individual classification has also been referred to as *realization*.

Classification via subsumption endows a taxonomy with formal meaning. Since the proper location of every concept and individual is uniquely determined by its definition, we have a *definitional* taxonomy. This yields numerous advantages. A definitional taxonomy facilitates incremental specialization of concepts by inheritance and differentiation from more general concept(s). Definitional taxonomies are especially valuable for visualizing relationships among concept descriptions, some of which may turn out to be unexpected or undesirable. Visualization can be enhanced by a graphical user interface with facilities for exploring the taxonomy. Classification maintains the integrity of the taxonomy as concepts are added and modified over time. Classification also enables a powerful pattern matching or *query* facility for retrieving concepts and individuals. Other inferential support by a description logic system includes automatic type checking and detection of redundant, inconsistent and vacuous definitions. Elaboration on all these benefits may be found in [Brachman and Schmolze, 1985; MacGregor, 1991a; Woods, 1986; Woods, 1991].

One important consequence of definitional taxonomies is the fact that they preclude *cancellation* of inheritance, i.e., properties of a concept may be further specialized by its descendents, but never overridden.[1] The notion of cancelling inheritance seems perfectly natural, but perhaps it could also be described as seductive. Cancellation allows any concept to be placed anywhere in the taxonomy, and requires a search for matching concepts to be exhaustive. Consider placing an arbitrary concept X underneath any other concept Y in the taxonomy. With cancellation this can always be done. Simply specify X by cancelling all properties of Y that are not true of X and adding all properties of X that Y does not exhibit. This point is neatly illustrated by a delightful anecdotal example due to Brachman, consisting of a question and a series of answers [Brachman, 1985]:

> **Q:** What's big and gray, has a trunk, and lives in the trees?
>
> **A1:** An elephant, I lied about the trees.
>
> **A2:** A giraffe – I lied about the color, the trunk and the trees.
>
> **A3:** An idea – I lied about the color, the trunk, the trees, and about the "lives".

[1] However, some recent description logic work has explored default inheritance of properties, e.g., [Baader and Hollunder, 1992; Quantz and Royer, 1992].

The problem, however, is a serious one. When cancellation is permitted in a taxonomy, concepts no longer constitute definitions, but merely a collection of defaults. One consequence is that concepts can no longer be classified automatically.

Description logics evolved from the highly influential KL-ONE system [Brachman and Schmolze, 1985] and its immediate successor, NIKL [Schmolze and Mark, 1991]. Description logics enjoy a wide variety of contemporary implementations, including BACK [von Luck *et al.*, 1987], CLASSIC [Borgida *et al.*, 1989], CRACK [Bresciani *et al.*, 1995], K-REP [Mays *et al.*, 1991a], KRIS [Baader and Hollunder, 1991], LOOM [MacGregor, 1991b], and SB-ONE [Kobsa, 1991]. It has seen a correspondingly diverse range of applications, e.g., financial marketing expertise [Apté *et al.*, 1992], software information retrieval [Devanbu *et al.*, 1991], knowledge-based presentation of multimedia information for equipment repair [Feiner and McKeown, 1990] and for equipment operation [Wahlster *et al.*, 1993], conflict resolution in production systems [Yen *et al.*, 1991], and system configuration [Owsnicki-Klewe, 1988; Searls and Norton, 1990; Wright *et al.*, 1993; Weida, 1996]. K-REP is currently being used in a clinical information system to represent medical knowledge, including drugs, treatments, and so on. That system is now in around-the-clock production use [Mays *et al.*, 1996].

All extant description logic systems operate under an *open terminology assumption*, i.e., there may be pertinent concepts that are not explicitly defined in the knowledge base. Thus, an individual might not be fully described by any concept or group of concepts in the knowledge base.

2.1.2 Language

Having introduced description logic at a high level, we now present the description logic language used in this thesis. A formal specification appears in Figure 2.1; the following discussion is somewhat informal.

In description logic, a *concept* is an intensional description of a class of *individ-*

A formal specification of the description logic used in this thesis is based on [Baader *et al.*, 1991] and [Patel-Schneider and Swartout, 1993]. These (possibly subscripted) symbols are used:

C	concept	CN	concept name
R	role	RN	role name
I	individual	IN	individual name

Concepts are either concept names or composed via the operators of the following table (where $R^{\mathcal{I}}(d) = \{ \text{e} \mid \text{<d,e>} \in R^{\mathcal{I}}\}$):

Concrete Form	**Abstract Form**	**Semantics**
thing	$\top$	$\Delta^{\mathcal{I}}$
nothing	$\bot$	$\emptyset$
(and $C_1 \ldots C_n$)	$C_1 \sqcap \ldots \sqcap C_n$	$C_1^{\mathcal{I}} \cap \ldots \cap C_n^{\mathcal{I}}$
(all R C)	$\forall R : C$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$
(at-least n R)	$\geq nR$	$\{d \in \Delta^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d)\rvert \geq n\}$
(at-most n R)	$\leq nR$	$\{d \in \Delta^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d)\rvert \leq n\}$
(exactly n R)	$= nR$	$\{d \in \Delta^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d)\rvert = n\}$
(the R C)	$= 1R : C$	$\{d \in \Delta^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d) \cap C^{\mathcal{I}}\rvert = 1\}$
(fills R I)	$R : I$	$\{d \in \Delta^{\mathcal{I}} \mid I^{\mathcal{I}} \in R^{\mathcal{I}}(d)\}$

Roles are just role names. A knowledge base is defined by a sequence of statements according to this table:

Concrete Form	**Abstract Form**	**Semantics**
(define-concept CN C)	$CN \doteq C$	$CN^{\mathcal{I}} = C^{\mathcal{I}}$
(define-primitive-concept CN C)	$CN \sqsubseteq C$	$CN^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
(define-primitive-role RN)	$RN \sqsubseteq \top$	$RN^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
(create-individual IN C)	$IN \in C$	$IN^{\mathcal{I}} \in C^{\mathcal{I}}$

Names must be defined exactly once, before being used (for brevity, our examples omit introduction of role names, which is trivial in this language). Meaning is assigned by an interpretation, $\mathcal{I}$. The interpretation consists of a domain $\Delta^{\mathcal{I}}$ and a mapping $\cdot^{\mathcal{I}}$, which maps names to their extensions in the interpretation. Extensions of concepts are subsets of $\Delta^{\mathcal{I}}$, extensions of roles are subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and extensions of individuals are elements of $\Delta^{\mathcal{I}}$. Distinct individual names have distinct extensions.

Figure 2.1: Description Logic Syntax and Semantics

```
(define-primitive-concept COMPANY)

(create-individual IBM
    COMPANY)

(define-primitive-concept CPU)

(define-primitive-concept RISC)

(define-primitive-concept RAM)

(define-primitive-concept DISK)

(define-primitive-concept SYSTEM)

(define-primitive-concept OS
    SYSTEM)

(define-primitive-concept UNIX
    OS)

(define-concept IBM-CPU
    (and CPU
        (fills vendor IBM)))

(define-concept RISC-CPU
    (and CPU
        (the technology RISC)))

(define-concept IBM-RISC-CPU
    (and CPU
        (fills vendor IBM)
        (the technology RISC)))

(define-concept IBM-PROCESSOR-DEVICE
    (all processor IBM-CPU))

(define-primitive-concept COMPUTER-SYSTEM
    (and SYSTEM
        (the vendor COMPANY)
        (all processor CPU)
        (at-least 1 processor)
        (all primary-storage RAM)
        (at-least 1 primary-storage)
        (all secondary-storage DISK)
        (the operating-system OS)))

(define-concept UNIPROCESSOR-SYSTEM
    (and COMPUTER-SYSTEM
        (exactly 1 processor)))

(define-concept DUALPROCESSOR-SYSTEM
    (and COMPUTER-SYSTEM
        (exactly 2 processor)))

(define-concept DISKLESS-SYSTEM
    (and COMPUTER-SYSTEM
        (exactly 0 secondary-storage)))

(define-concept RISC-MULTIPROCESSOR-SYSTEM
    (and COMPUTER-SYSTEM
        (all processor RISC-CPU)
        (at-least 2 processor)))

(define-concept DUAL-IBM-PROCESSOR-SYSTEM
    (and COMPUTER-SYSTEM
        (all processor IBM-CPU)
        (exactly 2 processor)))

(define-concept UNIX-RISC-SYSTEM
    (and COMPUTER-SYSTEM
        (all processor RISC-CPU)
        (the operating-system UNIX)))
```

Figure 2.2: Sample Definitions

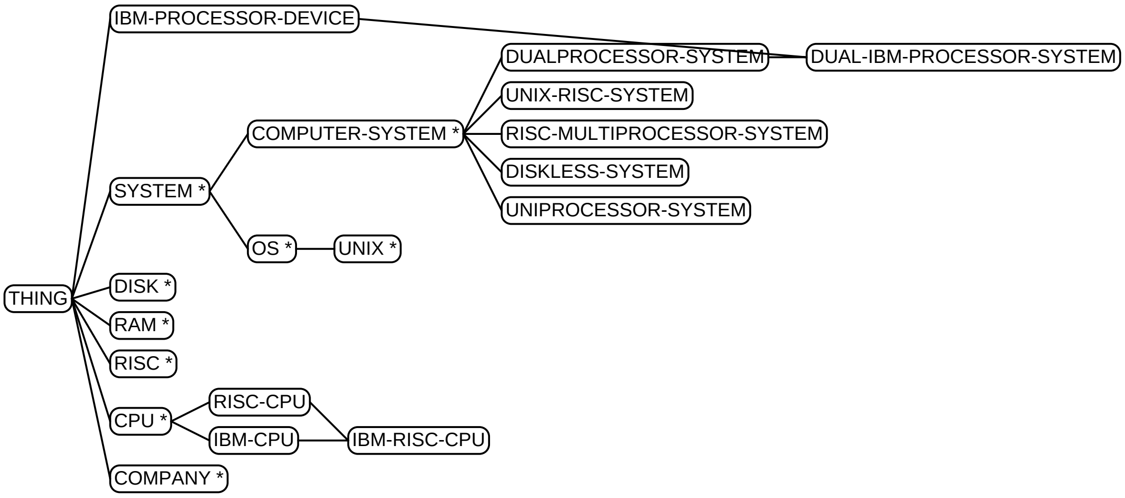


Figure 2.3: Sample Concept Taxonomy

uals. In a configuration application, a COMPUTER-SYSTEM concept (as defined in Figure 2.2 and explained later) may denote the set of possible computer systems, including a particular individual computer system named COMPUTER-SYSTEM123. We adopt the notational convention of writing concept and individual names in small capitals and suffixing individual names with unique identification numbers. A concept definition is introduced by the *define-concept* or the *define-primitive-concept* operator; the distinction between the two will be made clear later on. The definition of an individual is introduced with the *create-individual* operator. The name of a concept or individual can be used as shorthand for its description when forming subsequent descriptions.

A *role* is a first-class entity that denotes a binary relation between individuals. When a role denotes a partial function, it is also referred to as an *attribute*. In both of the description logic systems we use, K-REP and CLASSIC, concepts and individuals are described with the same language, largely by stating restrictions on roles. The *role restrictions* on a concept constrain relations pertaining to every individual in the concept's extension; the role restrictions on an individual pertain to the individual itself. In this thesis, we will consider value, fills, at-least, and at-most restrictions on roles. The *value* restriction of a role is a concept that constrains the range of the relationship; only instances of the value restriction may *fill* the role (together, they are its *fillers*). For example, the processor role of COMPUTER-SYSTEM has a value restriction of CPU. A role also has *at-least* and *at-most* restrictions which constrain how many fillers it may have. For example, COMPUTER-SYSTEM's processor role has a minimum cardinality of one, whereas its maximum cardinality is unrestricted. When the at-least restriction is zero, filling a role is optional. For example, since networked computer systems need not have their own secondary storage, COMPUTER-SYSTEM's secondary-storage role has a minimum cardinality of zero. Restrictions on role R are referred to as *value-restriction(R)*, *fillers(R)*, *at-least(R)*, and *at-most(R)*; they default to THING (the universal concept), the empty set, 0, and ∞ respectively. Conjunctive descriptions are composed via the *and* operator. The other description-forming operators of K-REP and CLASSIC are not addressed in this thesis.

It is important to note that we make the standard description logic assumption of acyclic descriptions. That is, the definition of a concept or individual cannot refer to itself either directly or indirectly.[2]

A concept may inherit from one or more *base* concepts, which are subsumers explicitly stated in its definition. For example, UNIPROCESSOR-SYSTEM has a base concept of COMPUTER-SYSTEM, which in turn has a base concept of SYSTEM. Local and inherited properties are combined by logical intersection. Consequently, UNIPROCESSOR-SYSTEM shares all the properties of COMPUTER-SYSTEM, with the additional restriction of at most one processor.[3] Likewise, an individual instantiates base concepts explicitly included in its definition, e.g., COMPANY is the base concept of IBM in Figure 2.2. The set of roles restricted by a concept or individual (locally and/or via inheritance) is written *restricted-roles(X)* and is specified as follows, where R_X denotes role R of concept or individual X:

[2]Cyclic concept descriptions pose thorny problems for subsumption. Details are beyond the scope of this thesis, but the interested reader is referred to [Nebel, 1990a].

[3]The stated constraint of (exactly 1 processor) is decomposable into (at-least 1 processor), which is inherited from COMPUTER-SYSTEM, and (at-most 1 processor), which is not inherited.

Definition 1 *The* restricted roles *of concept or individual* X *are the roles* R *of* X *for which*

1. *value-restriction*(R_X) *is properly subsumed by* THING*, or*

2. | *fillers*(R_X) | > 0*, or*

3. *at-least*(R_X) > 0*, or*

4. *at-most*(R_X) $< \infty$*.*

This chapter uses examples from the pedagogical knowledge base defined in Figure 2.2, where *all* gives a value restriction. As shorthand, *exactly* combines at-least and at-most restrictions of the same cardinality, and *the* combines a value restriction with a cardinality restriction of exactly one. The resulting concept taxonomy appears in Figure 2.3, with primitive concepts (see below) marked by an asterisk. Individuals such as IBM are not shown in the diagram. Notice that classification has discovered implicit relationships, e.g., IBM-RISC-CPU is subsumed by IBM-CPU and RISC-CPU, even though its base concept is CPU. Now consider the simplified description of a COMPUTER-SYSTEM, repeated from Figure 2.2:

```
(define-primitive-concept COMPUTER-SYSTEM
  (and SYSTEM
       (the vendor COMPANY)
       (all processor CPU)
       (at-least 1 processor)
       (all primary-storage RAM)
       (at-least 1 primary-storage)
       (all secondary-storage DISK)
       (the operating-system OS)))
```

It has a base concept, SYSTEM, and five restricted roles: vendor, processor, primary-storage, secondary-storage, and operating-system. Assuming that 486DX-

33MHz-123 is defined as an individual PROCESSOR, the following definition of an individual COMPUTER-SYSTEM is well-formed:

(create-individual COMPUTER-SYSTEM123
    (and COMPUTER-SYSTEM
        (fills vendor IBM)
        (fills processor 486DX-33MHz-123)))

Although its primary-storage, secondary-storage, and operating-system roles are not yet filled, COMPUTER-SYSTEM123 still inherits restrictions on those roles from its base concept, COMPUTER-SYSTEM. For instance, it is known to have at least one primary-storage, i.e., a memory board.

COMPUTER-SYSTEM is a primitive concept; whereas fully defined concepts specify necessary and sufficient conditions for class membership, *primitive* concepts specify only necessary conditions. Primitive concepts do not subsume other concepts unless the subsumption is explicitly sanctioned, e.g., UNIPROCESSOR-SYSTEM is subsumed by COMPUTER-SYSTEM. A description's primitiveness is characterized by the primitive concepts among its ancestors (inclusive) in the taxonomy:

Definition 2 *The* primitives *of concept* C *are the primitive concepts among the set consisting of* C *and the transitive closure of its base concepts.*

For example, primitives(COMPUTER-SYSTEM) = {COMPUTER-SYSTEM, SYSTEM}. An individual's primitiveness is defined similarly:

Definition 3 *The* primitives *of individual* I *are the primitive concepts among the transitive closure of its base concepts.*

Thus, primitives(COMPUTER-SYSTEM123) = {COMPUTER-SYSTEM, SYSTEM}.

Two concepts are inferred to be disjoint if their role restrictions are mutually exclusive. For example, UNIPROCESSOR-SYSTEM and DUALPROCESSOR-SYSTEM are disjoint by virtue of the cardinality restrictions on their processor roles, which are exactly one and exactly two, respectively. Concepts might also be disjoint on account of their primitives, but a description logic system has no basis for inferring that. Instead, many description logic systems, including K-REP and CLASSIC, support explicit declarations that sets of primitive concepts are pairwise disjoint. Our sample terminology has no disjointness declarations for brevity, but in practice it should declare that COMPANY, CPU, RISC, RAM, DISK, and SYSTEM are mutually disjoint, etc. When concepts are disjoint, all subsumees of one are disjoint from all subsumees of the others. Declaring suitable disjointness conditions in a terminology is crucial to sound knowledge engineering. Moreover, it helps to guide recognition as we will see in Chapter 4.

2.1.3 Principal Inferences

The knowledge representation service provided by a description logic system centers on a set of core inferences, which we now summarize. It is natural to give these inferences a set-theoretic interpretation. Some inferences pertain to a single description:

- **Vacuity:** Is a concept without significant restrictions? If so, it denotes the universal set and is equivalent to the distinguished concept named THING which constitutes the top of the concept lattice.
- **Coherence:** Are a concept's primitives and restrictions satisfiable? Otherwise, it denotes the empty set and is equivalent to the distinguished concept named NOTHING which constitutes the bottom of the concept lattice.[4]

Several inferences make decisions by comparing two descriptions:

- **Subsumption:** Is the set of possible individuals described by concept *C1* a superset of that described by concept *C2*? If so, *C1* subsumes *C2* and

[4]To avoid clutter, NOTHING is not shown in concept taxonomy diagrams, e.g., Figure 2.3.

conversely *C2* specializes *C1*. This is written as $C1 \Leftarrow C2$ or interchangeably as $C2 \Rightarrow C1$.

- **Equivalence:** Do concepts *C1* and *C2* describe the same set of possible individuals? We write this as $C1 \equiv C2$.

- **Consistency and Disjointness:** Do the sets of possible individuals described by concepts *C1* and *C2* have a non-empty intersection? If so, *C1* and *C2* are consistent. If not, they are disjoint.

- **Recognition:** Is individual I a member of the set described by concept C? If so, I instantiates C, written $I \epsilon C$.

Other inferences compare a description with the set of concepts in a terminology:

- **Classification:** Given an existing concept taxonomy[5] and an additional concept C, classification determines the proper location for C within the taxonomy. In effect, classification answers a pair of questions:

  1. What are the most specific concepts in the taxonomy that subsume C?
  2. What are the most general concepts in the taxonomy that C subsumes?

- **Realization (individual classification):** Given a concept taxonomy, and an individual I, what are the most specific concepts that I instantiates?

Some inferences use the taxonomy as a basis for pattern matching:

- **Concept Retrieval:** Which concepts are subsumed by concept C?

- **Individual Retrieval:** Which individuals instantiate concept C?

Numerous other inferences supported by description logic systems are not discussed here. However, subsequent sections will elaborate on subsumption, recognition, classification, consistency and disjointness, due to their importance in our work.

[5]Upon initialization, a concept taxonomy contains just THING and NOTHING.

2.1.4 Subsumption

Set theoretically, one concept *subsumes* another only when every possible instance of the second is also an instance of the first. In practice, subsumption is often determined by syntactic comparison of the concept descriptions. Many systems, including K-REP, CLASSIC, and LOOM, employ (largely) *structural* subsumption algorithms. First, descriptions are preprocessed for *normalization* and *completion*. Normalization converts the intrinsic description into canonical form. The most salient aspects of normalization involve (1) gathering together all the information about a particular role that is explicitly stated within the description, and (2) migrating role-related information to the top level of the description. Algebraically, normalization uses the associativity and commutativity of conjunction, along with the law that the two following expressions are equivalent:

```
(all r (and x y))
(and (all r x) (all r y))
```

To convey the flavor of normalization without delving into algorithmic details, we simply observe that the second concept below is, roughly speaking, a normalized equivalent of the first:

```
(define-concept ABNORMAL
    (and (all r x)
         (and (all s y)
              (all r z))))

(define-concept NORMAL
    (and (all r (and x z))
         (all s y)))
```

During completion, role restrictions are inherited from base concepts. These inherited restrictions are combined (logically intersected) with local restrictions contained within the concept description to form a fully explicit concept description.[6] In particular, at-least restrictions are maximized, at-most restrictions are minimized, value restrictions are conjoined, and sets of fillers are unioned (each filler is included once, whether it is specified locally, inherited, or both). See [Nebel, 1990a] for further details on this process. The normalized and completed description comprises a set of structural components. For the language treated in this thesis, structural components include primitives and elementary role restrictions: at-least, at-most, value, and fillers. After this preprocessing, subsumption can be computed efficiently by matching structural components. One concept *structurally subsumes* another just in case every structural component of the first subsumes some structural component of the second (recursively). For example, the normalized and completed forms of RISC-CPU and IBM-RISC-CPU essentially amount to the descriptions shown in Figure 2.4, where each conjunct of the description corresponds to a structural component. Each component of NORMALIZED-AND-COMPLETED-RISC-CPU has an underlined counterpart in NORMALIZED-AND-COMPLETED-IBM-RISC-CPU. This structural correspondence establishes the subsumption relationship.

Structural subsumption works well for the restricted languages of systems like K-REP, CLASSIC, and LOOM, and its merits are extolled in [Borgida, 1992]. How well it can be extended for extremely expressive languages is unclear, but see [MacGregor, 1994] for an interesting proposal regarding "a description classifier for the predicate calculus." A completely different approach, found notably in KRIS, relies on a *satisfiability checking* algorithm [Baader *et al.*, 1992]. Noting that concept *C1* subsumes concept *C2* just in case $\neg C1 \wedge C2$ is unsatisfiable, a satisfiability-based algorithm determines subsumption by attempting to generate a model where $\neg C1 \wedge C2$ is interpreted as a non-empty set.

Several subsumption algorithms have been discussed in the literature. The most thorough and modern presentation of a structural subsumption algorithm is found in [Borgida and Patel-Schneider, 1994]. Satisfiability-checking algorithms are given in [Hollunder *et al.*, 1990; Donini *et al.*, 1991].

[6] In more expressive description logics, completion includes additional logical deductions.

(define-concept NORMALIZED-AND-COMPLETED-RISC-CPU
  (and CPU
    (at-least 1 technology)
    (at-most 1 technology)
    (all technology RISC)))

(define-concept NORMALIZED-AND-COMPLETED-IBM-RISC-CPU
  (and CPU
    (at-least 1 vendor)
    (at-most ∞ vendor)
    (all vendor THING)
    (fills vendor IBM)
    (at-least 1 technology)
    (at-most 1 technology)
    (all technology RISC)))

Figure 2.4: Normalized and Completed Descriptions

A seemingly innocuous extension to the expressiveness of a representation language may dramatically compromise the tractability (polynomial time complexity in the worst case) of associated inferences. Brachman and Levesque's landmark paper on this phenomenon launched a significant thread in description logic research over the past decade [Brachman and Levesque, 1984].[7] They focused their analysis on one such "crossover point" in the computation of subsumption relationships. First, they examined a typical language for which subsumption can be computed in $O(n^2)$ time. Next, they showed that a simple variant of that language is co-NP-complete. They concluded that designers of description logic systems must make careful choices in trading expressiveness for tractability. Moreover, there is no single best choice. Instead, different choices may be appropriate depending on the application.

[7] Much of which is included in [Levesque and Brachman, 1985].

The result of Brachman and Levesque has practical significance because their co-NP-complete language is a subset of the languages employed by such systems as KL-ONE. Nebel later showed that another subset of the languages used in systems like KL-ONE and BACK creates an NP-hard subsumption problem[Nebel, 1988]. Patel-Schneider then demonstrated that subsumption in NIKL, LOOM, and similar systems is undecidable as well [Patel-Schneider, 1989]. By showing that no complete algorithm for such languages is possible, his result underscored a trend towards sound but consciously incomplete subsumption and classification algorithms. Schmidt-Schauß proved that a very simple concept language limited to conjunction of concepts, restrictions on values of roles, and equality among role chains is undecidable. However there is no problem when the chains are restricted to attributes (functional roles) [Schmidt-Schauß, 1989]. More recently, Nebel showed that subsumption in terminologies, which permit concepts to reference previously defined concepts, is inherently intractable (co-NP-complete) [Nebel, 1990b]. Today, a fairly comprehensive view of complexity in description logic has emerged [Donini *et al.*, 1991; Buchheit *et al.*, 1993].

Since our predictive concept recognition methodology and our work on constraint network reasoning in description logic both rely on subsumption, classification, and similar inferences, the preceding results are pertinent. However, the impact is attenuated by the fact that in our work, we can classify the underlying concepts in advance of problem solving.

Given that tractable subsumption is impossible for expressive description logic languages, designers of description logic systems are faced with a range of potentially reasonable compromises:

1. At one extreme are languages of limited expressiveness, with sound, complete, and tractable subsumption algorithms.

2. Many fairly expressive languages admit sound but somewhat incomplete subsumption algorithms that are quite fast.

3. At the other practical extreme are more expressive, yet still decidable, languages with sound and complete, but presumably exponential subsumption algorithms.

The language studied in this thesis and specified in Figure 2.1 falls into the first category. Current description logic systems stake out different positions in this design space. CLASSIC approximates the first choice [Borgida and Patel-Schneider, 1994]. Subsumption in CLASSIC is tractable and also complete with respect to concepts. It is complete with respect to individuals only under a non-standard semantics for individuals. The CLASSIC algorithm is fast in practice [Heinsohn *et al.*, 1992b]. LOOM exemplifies the second choice, with a more expressive language and a fast but incomplete algorithm. K-REP also has an incomplete algorithm which places it somewhere between CLASSIC and LOOM. The third strategy has been pursued by KRIS. Recently, intriguing experimental results suggest that this strategy may not prevent KRIS from competing on performance with less expressive systems for the subset of the KRIS language that they have in common [Heinsohn *et al.*, 1992b].

While intractability results for the subsumption problem are sobering, it must be emphasized that they are worst-case analyses. Moreover, as discussed in Section 2.1.9, Doyle and Patil have argued that worst-case performance should not be the predominant factor in judging subsumption algorithms [Doyle and Patil, 1991].

2.1.5 Recognition

The *recognition* inference determines whether a given individual instantiates a given concept. We will not go into detail about computing recognition, because it is substantially similar to computing subsumption. One important difference is the *unique name assumption*, where uniquely named individuals are considered unique. That is, two individuals with identical descriptions are presumed to be distinct on the strength of their distinct names.[8] In contrast, a pair of concepts with different names and equivalent descriptions are treated as synonyms.

2.1.6 Classification

Concept classification is a process that places concepts into an explicit taxonomy such that each concept subsumes its descendants and is subsumed by its ancestors. As a result, a concept's location is uniquely determined such that its parents are its most specific subsumers and its children are its most general subsumees. Equivalent concepts share the same location. In practice, classifiers install concepts in the explicit taxonomy one at a time, taking advantage of the relationships among all previously installed concepts. Classification algorithms naturally operate in two phases [Woods, 1991]. The first, which Woods calls the MSS search, traverses downward from the root concept, THING, to find the most specific subsumers. The second, called the MGS search, proceeds from there to identify the most general subsumees. Classification of individuals entails a search essentially similar to the MSS search. There is no analogue to the MGS search for individuals because, by definition, they cannot be specialized.

2.1.7 Consistency and Disjointness

Two concepts are consistent just in case an individual can instantiate them simultaneously. Under the usual open terminology assumption (OTA), this means that their conjunction is satisfiable. Thus *OTA-consistency* between a pair of concepts is merely the inverse of the standard disjointness inference. We elaborate on it here for the sake of later comparison; in Chapter 3, we will propose the idea of closed terminology consistency inferences in description logic. As an example, DISKLESS-SYSTEM is OTA-consistent with UNIPROCESSOR-SYSTEM, even though

[8]Some systems may allow explicit assertions of individual equality [Patel-Schneider and Swartout, 1993].

neither subsumes the other. This is because an individual can instantiate them both simultaneously. For instance:

(create-individual DISKLESS-UNIPROCESSOR-SYSTEM01
    (and COMPUTER-SYSTEM
        (exactly 0 secondary-storage)
        (exactly 1 processor)))

An individual and a concept are consistent when the individual either already instantiates the concept or can be monotonically updated[9] to do so. For example, COMPUTER-SYSTEM123 defined on page 24 is OTA-consistent with COMPUTER-SYSTEM, which it already instantiates, plus DISKLESS-SYSTEM, UNIPROCESSOR-SYSTEM, etc., which it can be monotonically updated to instantiate, e.g., it can assume the following description:

(and COMPUTER-SYSTEM
    (fills vendor IBM)
    (fills processor 486DX-33MHZ-123)
    (exactly 0 secondary-storage)
    (exactly 1 processor))

We can compute OTA-consistency directly from concept descriptions:

Theorem 1 *Concepts* C1 *and* C2 *are* OTA-consistent *iff*

1. *No primitive of* C1 *is disjoint from any primitive of* C2
2. *For every role* R *restricted by both* C1 *and* C2
    (a) *The cardinality restrictions on* R_{C1} *and* R_{C2} *intersect*

[9] Monotonic updates are defined on page 58.

(b) *If at-least(*$\mathrm{R_{C1}}$*) > 0 or at-least(*$\mathrm{R_{C2}}$*) > 0, then value-restriction(*$\mathrm{R_{C1}}$*) and value-restriction(*$\mathrm{R_{C2}}$*) are OTA-consistent*

(c) $\mid$ *fillers(*$\mathrm{R_{C1}}$*)* $\cup$ *fillers(*$\mathrm{R_{C2}}$*)* $\mid \; \leq$ *minimum(at-most(*$\mathrm{R_{C1}}$*), at-most(*$\mathrm{R_{C2}}$*))*

(d) *Every filler of* $\mathrm{R_{C1}}$ *is OTA-consistent with value-restriction(*$\mathrm{R_{C2}}$*) and every filler of* $\mathrm{R_{C2}}$ *is OTA-consistent with value-restriction(*$\mathrm{R_{C1}}$*)*

Proof: See Appendix B.

For example, UNIPROCESSOR-SYSTEM and UNIX-RISC-SYSTEM are OTA-consistent:

1. First of all, we have primitives(UNIPROCESSOR-SYSTEM) = primitives(UNIX-RISC-SYSTEM) = {COMPUTER-SYSTEM, SYSTEM}. Since SYSTEM subsumes COMPUTER-SYSTEM, those primitives are not disjoint.

2. UNIPROCESSOR-SYSTEM and UNIX-RISC-SYSTEM inherit the same set of restricted roles from COMPUTER-SYSTEM; neither restricts any other roles. Considering their processor roles, UNIPROCESSOR-SYSTEM further restricts the cardinality to exactly one, while UNIX-RISC-SYSTEM further restricts the value to a be RISC-PROCESSOR. Considering their operating-system roles, UNIX-RISC-SYSTEM further specifies that the operating system must be (a version of) UNIX. In both cases, it is easy to see that the requirements of Theorem 1, clause 2 are met.

Computing OTA-consistency between an individual and a concept is basically the same as between a pair of concepts.

2.1.8 Least Common Subsumer

Cohen, et al., have shown how to compute the *least common subsumer* (LCS), or most specific generalization, of a set of description logic concepts [Cohen *et al.*, 1992]. Their work in machine learning used LCS inferences to learn description logic concepts from sample individuals. Despite its new name, the notion of a least common subsumer will be familiar to computer scientists as the least upper bound. Roughly speaking, the LCS captures the restrictions placed on an individual when it is known to instantiate one of a set of concepts. As a simple illustration, the LCS of UNIPROCESSOR-SYSTEM and DUAL-IBM-PROCESSOR-SYSTEM (both defined in Figure 2.2 on page 20) is:

(and COMPUTER-SYSTEM
    (at-least 1 processor)
    (at-most 2 processor))

For languages such as K-REP and CLASSIC, the LCS always exists and is essentially unique; although syntactic variants may exist, they are semantically equivalent [Cohen *et al.*, 1992]. Note that the restricted nature of these languages makes the least common subsumer inference interesting: disjunction would allow the LCS of a set of concepts to be specified trivially without revealing anything about their commonality. In Chapter 3, we will take advantage of the LCS inference to help derive constraints imposed on a partially specified individual by a closed terminology.

2.1.9 Current Limitations

Levesque and Brachman have argued that a knowledge representation facility should be correct, i.e., sound and complete, yet dependably quick enough for the most critical applications [Levesque and Brachman, 1987]. Classification is the most expensive operation provided by standard description logic systems. It would follow that description logic languages should be restricted so that tractable and guaranteed correct classification operations can be assured. Doyle and Patil have called this conclusion the *restricted language thesis* [Doyle and Patil, 1991]. They counter with several strong arguments, based on their experience using NIKL in medical expert systems:

- Pursuing the restricted language thesis "destroys the generality of the language and the system."
- Desired asymptotic efficiency can be achieved without omitting useful but problematic constructs entirely; rather they should be employed judiciously.
- In most domains, the proportion of "natural kind" primitive concepts is substantial. Severe language restrictions make it impossible to define many other concepts, which must therefore be declared primitive as well. Since primitive concepts cannot be classified freely, the utility of classification is diminished by these "fake" primitives.
- The emphasis on tractability of classification is misguided, and general purpose systems should not focus on worst-case performance for the most critical applications. Instead, (1) the general case is also important, (2) other costs, such as space, should be considered, and (3) other inferences besides classification should be taken into account.

In any event, Nebel's recent result presumably rules out polynomial time performance in the worst-case for any reasonably useful classification-based system. Inspired by the outlook of Doyle and Patil, a significant portion of this thesis looks to extend the expressiveness of description logic languages by encompassing temporal information.

2.2 Closed World Reasoning

We now briefly characterize closed world reasoning.

Closed world reasoning entails an assumption that a database or knowledge base is complete, in the sense that it contains all the information relevant for the problem of interest. This sort of assumption has long been implicit in the use of traditional databases. For example, a company's employee database is expected to contain records for precisely those people who are employed by the company. One can then determine the highest paid employee by examining all records. Obviously this would not be possible if some employees were omitted from the database. Closed world reasoning is more interesting in the case of deductive database systems, such as Prolog, that reason about the contents of their knowledge base at an abstract level. These systems assume that any conclusion that is not logically derivable from the database is false. This latter notion of a *closed world assumption* was formalized in artificial intelligence by [Reiter, 1978].

Completeness assumptions have also been common in plan recognition work, e.g., [Kautz, 1991b]. For example, plan recognition systems often assume that they have at their disposal an explicit representation for every goal and/or plan that might account for observations of the user's activities. While this strong assumption is admittedly a limitation, it is also important to point out that a system's inherently limited ability to recognize plans is intimately connected with its power to draw conclusions.

In Chapter 3, our thesis couples a sort of "closed world" reasoning at the schema level with description logic for the first time. Since our completeness assumption applies at the conceptual level rather than the individual level, we will call it the *closed terminology assumption.* Our work on plan recognition in Chapter 3 makes a similar assumption about the completeness of the given plan library, which we call the *closed library assumption.*

2.3 Temporal Reasoning

We now summarize temporal reasoning as it relates to our thesis.

The representation of temporal knowledge and concomitant reasoning plays an important part in many areas of artificial intelligence, including natural language processing, diagnosis, scheduling, planning, and plan recognition. A useful survey is [Shoham and McDermott, 1990]. Here, we focus on constraint-based temporal languages suitable for describing temporal patterns. Broadly speaking, temporal constraints can express qualitative relationships such as (either) *before or after* and metric relationships such as *five hours before*. Temporal constraints can be applied to time points and/or time intervals. In practice, most research in this area has addressed either systems of binary qualitative constraints between intervals in the tradition of [Allen, 1983a] or systems of linear inequalities between metric points. Qualitative and metric constraints overlap in their ability to convey temporal relationships. As we shall see in Section 2.3.3, their complementary powers can be integrated profitably.

2.3.1 Qualitative Constraints

Allen, in his influential work on maintaining knowledge about temporal intervals, enumerated a total of seven primitive qualitative relationships, plus their inverses, that might hold between any ordered pair of intervals [Allen, 1983a]. These primitives are illustrated in Figure 2.5. A *qualitative constraint* records the possible relationships between a particular pair of intervals as a disjunctive subset of these primitive relationships [Allen, 1983a]:

Definition 4 *A* qualitative constraint *between a pair of intervals* i *and* j *is a disjunctive subset of Allen's 13 primitive qualitative relations r_x, written* i $(r_1 \vee \cdots \vee r_n)$ j.

For example, the constraint *before* $\vee$ *meets* $\vee$ *met-by* $\vee$ *after* mandates temporal disjointness.[10] This condition, which is important for planning, cannot be represented using metric constraints limited to a set of simple linear inequalities [Vilain and Kautz, 1986]. As more information becomes available, a qualitative constraint may be refined by eliminating disjuncts, e.g., *before* $\vee$ *after* may be refined to *after*.

A *qualitative temporal network* consists of nodes that represent intervals and

[10] We will omit the arguments related by a constraint when their identity is unimportant.

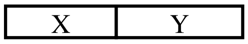
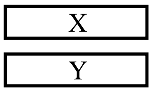
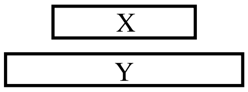
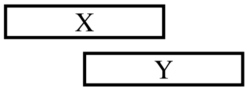
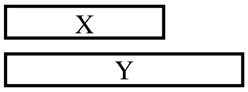
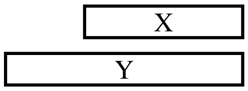
Temporal Relationship	Illustration	Inverse Relationship
X before Y		Y after X
X meets Y		Y met-by X
X equals Y		Y equals X
X during Y		Y contains X
X overlaps Y		Y overlapped-by X
X starts Y		Y started-by X
X finishes Y		Y finished-by X

Figure 2.5: Allen's Qualitative Relations

arcs that represent qualitative constraints between pairs of intervals. Allen proposed a simple polynomial-time constraint propagation algorithm to *complete* (or *close*) a qualitative temporal network by computing the implicit consequences of explicitly stated constraints, i.e., a transitive closure. Allen's algorithm is an instantiation of the path consistency algorithm for constraint satisfaction [Montanari, 1974; Mackworth, 1977].

Allen's constraint propagation algorithm is sound, but unfortunately not complete [Vilain *et al.*, 1989]. This is important because in practice, we cannot always expect that all temporal relations will be made explicit in a system's input. For example, our ability to compare different plans in light of their temporal constraints depends on the extent to which those constraints are made explicit. The incompleteness of Allen's algorithm stems from the expressive power of the constraints. Specifically, his algorithm is only guaranteed to produce correct results with respect to subgraphs of three vertices or less [van Beek, 1989]. This is called 3-consistency. Sound and complete closure is NP-hard [Vilain and Kautz, 1986]. As a result, we are faced with three standard alternatives:

1. Restrict the expressiveness of the qualitative constraints so that exact solutions can be obtained tractably. For example, [Vilain *et al.*, 1989] identified a subset of Allen's interval calculus derived from a point-based representation. Their subset admits complete polynomial-time constraint propagation.

2. Adopt an approximation algorithm such as Allen's, and live with the possible consequence that some relationships will remain undetected. Allen postulates that his algorithm's inferences correspond to those that humans find natural [Allen, 1983a]. There is a family of variations on Allen's algorithm that produce successively better approximations, but at increasingly onerous cost [van Beek, 1989]. In general, *n-consistency* can be computed in polynomial time for any specific value of n.

3. Use an exact, presumably exponential algorithm, and simply accept the amount of time it takes to finish. One such algorithm is proposed in [Valdes-

Perez, 1986]. This may be a reasonable option for relatively small problems.

Notice how these alternatives parallel the options for dealing with the intractability of subsumption in description logic (Section 2.1.4). Our work on the T-REX system implicitly exercises the second option through its use of the MATS temporal reasoner [Kautz and Ladkin, 1991]. However, it is important to note that T-REX is independent of this decision in the following sense: if MATS were redesigned to adopt one of the other choices, T-REX would reflect the change without requiring any modification itself.

2.3.2 Metric Constraints

A separate body of work on temporal reasoning has dealt with systems of linear inequalities to capture metric relations involving time points [Dechter *et al.*, 1991; Malik and Binford, 1983; Valdes-Perez, 1986]. These systems are independent of metric quanta, which can be chosen to meet the needs of a particular application. Linear inequalities can express bounds on absolute times as constraints on a single time point, e.g., TIME-POINT1 occurs before time 1000:

- TIME-POINT1 < 1000

TIME-POINT2 occurs no earlier than time 2000:

- $2000 \leq$ TIME-POINT2

and TIME-POINT3 occurs between times 3000 and 3001, inclusive:

- $3000 \leq$ TIME-POINT3 ≤ 3001

For notational convenience, the last of these examples combines two linear inequalities on the same time point. Linear inequalities can also express elapsed time as the difference between two time points, e.g., TIME-POINT5 occurs more than ten time units after TIME-POINT4:

- $10 <$ TIME-POINT5 $-$ TIME-POINT4

Again, inequalities can be combined, e.g., TIME-POINT6 and TIME-POINT7 occur within twenty time units of each other, inclusive:

- $-20 \leq$ TIME-POINT6 $-$ TIME-POINT7 ≤ 20

A set of metric constraints forms a *metric temporal network*, where the nodes represent time points and the arcs represent metric constraints between them. In a *simple temporal problem* there is at most one metric constraint between any pair of time points, and each metric constraint denotes a single numeric interval. Sound and complete closure for a simple temporal problem is computable in $O(n^3)$ time [Dechter *et al.*, 1991].

When the time points are extrema of time intervals as in Section 2.3.1, metric constraints allow us to say much more about how time intervals are related. Examples and discussion follow.

2.3.3 Integration of Qualitative and Metric Constraints

As we have seen, qualitative and metric constraint systems have complementary capabilities. Kautz and Ladkin designed a constraint reasoner which integrates reasoning over an Allen-style qualitative constraint network for time intervals and a simple metric constraint network for the starting and ending points of those intervals [Kautz and Ladkin, 1991]. These ideas were implemented by Kautz in the MATS system [Kautz, 1991a]. Metric information accounts for durations of intervals and gaps between endpoints of distinct intervals, along with absolute times. Kautz and Ladkin define such metric constraints formally [Kautz and Ladkin, 1991][11]:

[11] We have chosen to write *start* and *finish* where they wrote *left* and *right*, respectively.

Definition 5 *A* metric constraint *on the endpoints of intervals is the conjunction of two difference inequalities:*

- $(i_F - j_G \; R \; n) \wedge (j_G - i_F \; Q \; m)$*, abbreviated as* $-m \; Q \; i_F - j_G \; R \; n$*, where*

  1. i_{start} *and* i_{finish} *are the starting and ending points of interval* i
  2. $F, G \in \{start, finish\}$
  3. R, $Q \in \{\leq, <\}$
  4. m *and* n *are numerals or* $(-)\infty$

Thus, the difference between the absolute times of two time points is restricted to within a numeric interval. For example, the following states that the time interval known as EVENT1 has a duration within the range (7,11].

$$7 \;<\; \text{EVENT1}_{finish} - \text{EVENT1}_{start} \;\leq\; 11$$

MATS handles absolute starting and ending times of intervals by relating interval extrema to a distinguished time point which occurs at absolute time zero.

Notice that metric constraints can imply qualitative constraints and *vice versa.* For example, the following metric constraint implies that EVENT1 is *before* EVENT2.

$$0 \;<\; \text{EVENT2}_{start} - \text{EVENT1}_{finish} \;\leq\; \infty$$

To reach closure across its twin constraint networks, MATS alternates between qualitative and metric constraint propagation phases, passing results back and forth until nothing further can be concluded. Kautz and Ladkin prove that information loss is minimized in their metric-to-qualitative and qualitative-to-metric translation schemes [Kautz and Ladkin, 1991].

We chose to use MATS in the T-REX prototype because of its superior expressiveness. When necessary or appropriate for some application, we could restrict T-REX to a subset of MATS' capabilities. Meiri has proposed an alternative model which integrates qualitative and metric constraints in a single constraint network [Meiri, 1991]. Finally, a performance evaluation of several temporal reasoners is reported in [Yampratoom and Allen, 1993].

2.4 Synthesis

This chapter began with a review of description logic. It then presented relevant aspects of two other threads of artificial intelligence research, closed world reasoning and temporal reasoning, which to date have been mostly unconnected with description logic. The preceding material provides the necessary background for understanding our work. Chapter 6 will discuss other important related work.

In the ensuing chapters we aim to forge new connections. Chapter 3 introduces "closed world" reasoning over a description logic terminology and Chapter 4 integrates description logic with temporal reasoning. Chapter 5 brings all three of them together for the first time.

Chapter 3

Predictive Concept Recognition

3.1 Introduction

A description logic terminology is developed over a period of time through a *knowledge engineering* process. New concepts may be added, and existing concepts modified, until the terminology is satisfactory. This chapter pursues the idea of explicitly closing the set of concepts in a description logic knowledge base after the terminology is developed and before problem solving begins.[1] Our *closed terminology assumption* (CTA) also requires that the properties of every individual be fully accounted for by at least one single concept explicitly defined in the terminology.[2] For certain applications, such as system configuration, this CTA is fully appropriate during problem solving. Moreover, CTA enables useful inferences that would not be possible otherwise. In particular, this chapter introduces *predictive concept recognition*: given an unfinished description of an individual and some assumptions, we will show how to ascertain not only the concepts it already instantiates, but those it might eventually instantiate, and those it cannot. For example, INTEL-CPU and RISC-CPU concepts might well be defined in such a way

[1]This work was introduced in [Weida, 1995a], a preliminary version of [Weida, 1996].

[2]The closed terminology assumption is defined more precisely in Section 3.4, after the requisite groundwork has been established.

that they are OTA-consistent, but not CTA-consistent in the context of a certain terminology. In that context, if an INTEL-CPU is chosen, CTA enables the inference that it is not a RISC-CPU.

Configuration is an important, traditional application area for artificial intelligence research, e.g. [McDermott, 1982]. In our view, description logic is ideal for describing artifacts such as computer systems and their components, and for maintaining the consistency of large configuration knowledge bases as they evolve over time. Figure 3.1 shows the kind of configuration system we have in mind: a configuration engine that specializes in solving configuration problems utilizes a description logic system such as K-REP, both to access a configuration terminology and to represent a model of the particular configuration being worked on. The configuration engine can make significant use of traditional description logic inferences, as well as new inferences to be introduced here. The overall configuration system also contains other modules that are not shown, such as a user interface.

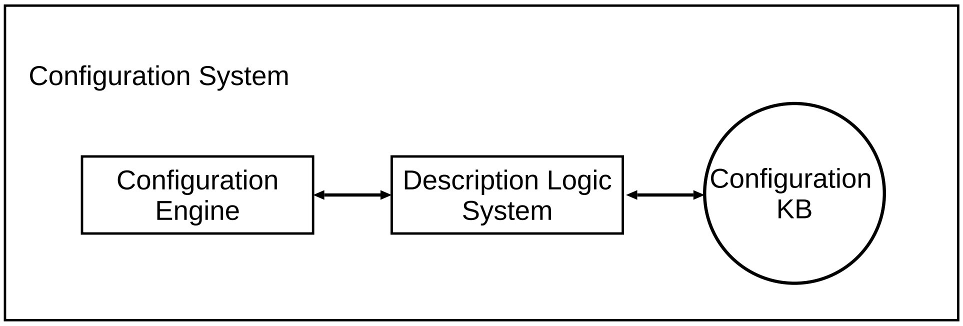


Figure 3.1: Configuration System Architecture

Previous work shows that description logic is well suited to configuration problems. MESON was used to address configuration [Owsnicki-Klewe, 1988], BEACON reached the advanced prototype stage at UNISYS[3] [Freeman, 1986; Searls and Norton, 1990], and CLASSIC is successfully deployed within a series of configurators for telephone switching devices at AT&T [Wright *et al.*, 1993]. At AT&T, configuration applications based on description logic have been used to process orders totalling more than two and a half billion dollars since 1990 [McGuinness

[3] Previously Burroughs.

and Resnick, 1995]. The UNISYS and AT&T solutions fit into the paradigm of Figure 3.1. However, in each case the description logic system makes the standard *open terminology assumption* (OTA) that the set of concepts in the terminology may be incomplete. Under OTA, descriptions of individuals are constrained only by the syntax and semantics of the description logic language; they are not otherwise constrained by the terminology. Unfortunately, OTA limits the ability to rule out the possibility of some instantiations and thus draw conclusions from the remainder. Indeed, all previous work in description logic operates under the OTA. We aim to enhance the utility of description logic for tasks such as configuration through our work with the K-REP system [Mays *et al.*, 1991a].

For configuration, the OTA is appropriate during knowledge engineering, when we construct a model of systems, components, and related concepts. During the knowledge engineering phase, knowledge engineers may continue to add new concepts. At some point, the knowledge engineering phase ends, and the knowledge base is "frozen" so it can be used to solve configuration problems. Thus, during a specific configuration task, closed terminology reasoning is more suitable because all relevant concepts, e.g., types of systems, are presumed to be known.[4] We also distinguish between abstract and concrete concepts to formally define when an individual's description may be considered finished, given a closed terminology. Imagine that a user incrementally specifies an individual computer system, along with its individual components, in collaboration with a configuration engine. The configuration engine is responsible for ensuring practical instantiations and recommending desirable ones. In general, the user can make choices in any order and at any level of abstraction. Hence, it may remain unclear for some time precisely which type of system will result. We can further exploit the closed terminology and its subsumption-based organization to:

1. Efficiently track the types of systems and components that are consistent

[4]Indeed, all possible configurations of an individual system are implicit in the concept taxonomy. However, performance-motivated restrictions on description logic expressiveness prevent it from reasoning with all relevant constraints on configurations. Consequently, system concepts may admit instantiations that cannot be realized in practice.

with the user's current choices. Reiterating an earlier example, suppose a user initially specifies an individual computer system with a certain kind of specialized input device. We will show how to efficiently focus our attention on the portion of the concept taxonomy representing computer systems that support the device of interest.

2. Infer constraints on the system and components that follow from current choices and the terminology's specific contents. Continuing with our example, if only IBM computer systems support that kind of device, we will assert that the user's system is made by IBM.

3. Offer the user and the configuration engine guidance by suitably restricting future choices. In our example, computer systems which do not support the desired input device are not included among the options.

4. Characterize the most general choices available for refining the description of a particular individual. Concluding our example, we can offer the user a choice among the set of most general computer system concepts capable of supporting the desired input device.

Thus, we can help focus the efforts of both the user and the configuration engine.

Once an individual system is finished, the standard description logic recognition inference establishes which concepts it instantiates. However, we want to recognize potential instantiations throughout the configuration process to inform both the user and the configuration engine. That is, given an unfinished description of an individual system, we want to reason about which system concepts it may come to instantiate. Such concepts are *consistent* with the individual system. We can also offer guidance by characterizing future options. For example, the user interface can present a choice of just those disk drives that are consistent with all previous choices. In a related vein, Goodman and Litman investigated the use of plan-constrained user interaction and applied it to a system for chemical process design [Goodman and Litman, 1992].

Stated more generally, the objective of the work presented in this chapter is to exploit the CTA as individuals are incrementally specified. In particular, we infer those concepts that are consistent with an individual at any moment. This new inference is called *potential instantiation*. Furthermore, we infer constraints on an individual that follow from the interaction between its current description, the closed set of concepts, and an assumption that any updates to the individual will be monotonic.[5] Moreover, when the terminology is revised (between problem solving sessions), K-REP will supply constraints that reflect those revisions, e.g., the configuration engine itself need not change due to the introduction or withdrawal of products. This chapter explains how we achieve these goals. The work described here is implemented in K-REP.

The following section details a framework for reasoning about the incremental instantiation of description logic individuals, such as the components of a particular configuration. After characterizing those aspects of the configuration problem we are concerned with in Section 3.3, and stating our assumptions and goals for predictive concept recognition in Section 3.4, we introduce in Section 3.5 the closed terminology consistency inferences that enable our solution. Section 3.6 shows how to mechanically augment a terminology to speed the recognition process, which is in turn presented in Section 3.7. Section 3.8 shows how to derive new constraints from predictive recognition. We compare open and closed terminologies in Section 3.9. Performance analysis is considered in Section 3.10, and Section 3.11 concludes.

3.2 Incremental Instantiation

To help decide if incremental specification of an individual may be finished, we distinguish between *concrete* and *abstract* concepts. A similar distinction was made in configuration work by [Kramer, 1991], but we will formally define the notion of a finished individual. In configuration, only individuals that are instances of concrete concepts can be included *per se* in a finished system. Abstract concepts

[5] Our *monotonic update assumption* will be defined in Section 3.4.

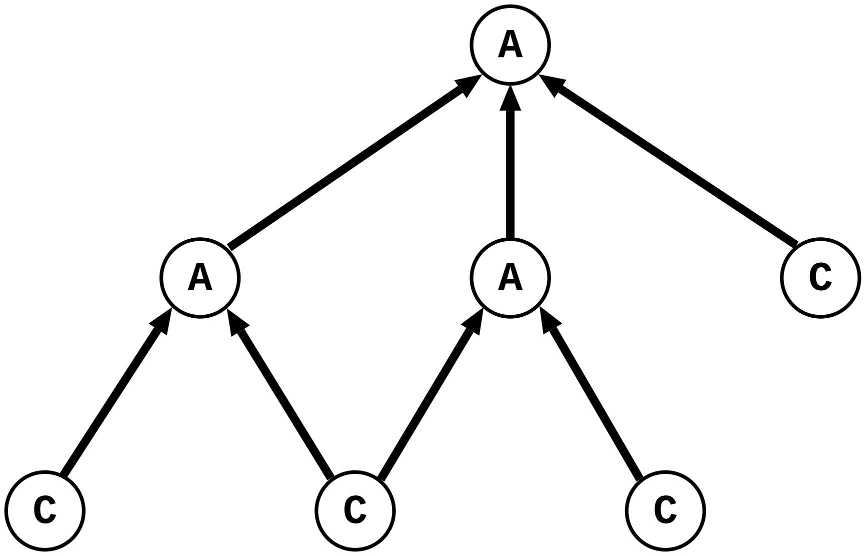


Figure 3.2: Abstract and Concrete Concepts

represent the commonality among a class of concrete concepts. For example, an actual system's processor may be of type 486DX-33MHZ,which is concrete (fully specific), but not merely of type CPU, which is abstract (too general). Note that all concepts in Figure 2.2 are abstract; concrete concepts are omitted for brevity. (We will add some concrete concepts to the terminology later in this section.) Figure 3.2 illustrates the taxonomic relationship among abstract concepts, labeled 'A', and concrete concepts, labeled 'C'; wide arrows connect a concept to its most specific subsumer(s). Notice that all leaf concepts are concrete.[6] Figure 3.3 adds several individuals labeled 'I' to the taxonomy of Figure 3.2; thin dashed arrows connect an individual to the most specific concepts that it instantiates. We will see that finished individuals must instantiate concrete concepts; before then they may perhaps only instantiate abstract concepts. We will further characterize the abstract / concrete distinction after introducing the helpful notion of *bijective instantiation*.

When an individual bijectively instantiates a concept, their primitives and restricted roles stand in one-to-one correspondence. A bijective instantiation demon-

[6]We detail this point at the end of this section.

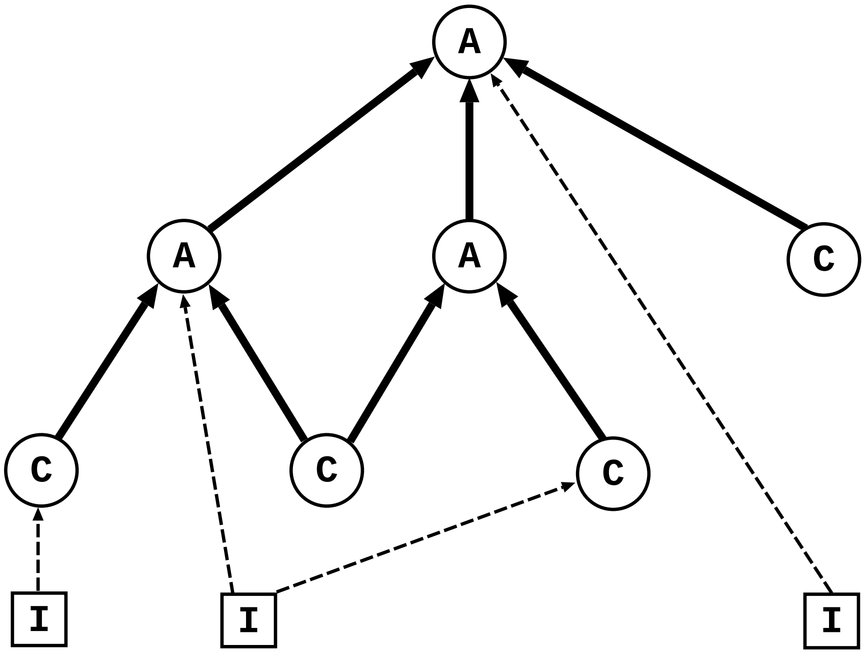


Figure 3.3: Abstract and Concrete Concepts with Individuals

strates that concept C explicitly accounts for each of individual I's primitives and role restrictions, and at the same time, I explicitly respects all of C's primitives and role restrictions:

Definition 6 *Individual* I bijectively instantiates *concept* C *iff*

1. *primitives(*I*)* $\equiv$ *primitives(*C*)*
2. *restricted-roles(*I*)* $\equiv$ *restricted-roles(*C*)*
3. *For every role* R *on restricted-roles(*I*)*
   (a) *at-least(*$\mathrm{R_I}$*)* $\geq$ *at-least(*$\mathrm{R_C}$*)*
   (b) *at-most(*$\mathrm{R_I}$*)* $\leq$ *at-most(*$\mathrm{R_C}$*)*
   (c) *value-restriction(*$\mathrm{R_I}$*)* $\Rightarrow$ *value-restriction(*$\mathrm{R_C}$*)*
   (d) *fillers(*$\mathrm{R_I}$*)* $\supseteq$ *fillers(*$\mathrm{R_C}$*)*

For example, COMPUTER-SYSTEM123 as defined on page 24 bijectively instantiates COMPUTER-SYSTEM in Figure 2.2 on page 20. It inherits all of COMPUTER-

SYSTEM's primitives and restricted roles, and does not add any others (it just adds fillers to already restricted roles). Instead of going into more detail with this example of bijective instantiation, we will shortly define and give a detailed example of a "finished" individual, which also exemplifies bijective instantiation.

The distinction between *concrete* and *abstract* concepts is application-specific, but can be characterized in terms of bijective instantiation:

Definition 7 *Concept* C *is* concrete *iff an individual which bijectively instantiates* C *might be both complete and sufficiently specific for the purposes of the intended application.*

In configuration, 486DX-33MHZ would be designated concrete, whereas more general concepts such as CPU and THING would be designated abstract:

Definition 8 *A concept is* abstract *iff it is not concrete.*

We model the concrete / abstract distinction as a boolean status associated with each concept. Like the distinction between individuals and concepts, the distinction between concrete and abstract concepts is clear in principle, but can be a knowledge engineering choice in practice. Then, subjective choices must be made to meet the needs of the application. For example, a 486DX-33MHZ might be produced by more than one manufacturer. e.g., IBM and INTEL. If this distinction were important for a particular configuration system, one alternative would be to make 486DX-33MHZ an abstract concept and create concrete specializations IBM-486DX-33MHZ and INTEL-486DX-33MHZ.

The "possibility" alluded to in Definition 7 is realized if the individual in question is also *finished* with sufficient role fillers that are in turn finished:

Definition 9 *Individual* I *is* finished *when*

1. I *bijectively instantiates at least one concrete concept, and*

2. *For every role* R *on restricted-roles(*I*)*

   (a) $|fillers(\mathrm{R_I})| \geq$ *at-least*$(\mathrm{R_I})$

   (b) *Every filler of* $\mathrm{R_I}$ *is finished*

The K-REP description language forbids cyclic descriptions, so this definition is well-founded. In fact, cycles are forbidden by all contemporary, implemented description logic systems that we are aware of. When the description of an individual computer system is finished, the configuration problem is solved.

As a simplified example, suppose the knowledge base of Figure 2.2 on page 20 also includes the concrete concepts and related individuals described on the left-hand side of Figure 3.4 on page 55.[7] The right-hand side of the figure describes a finished individual computer system, COMPUTER-SYSTEM-99, preceded by the other individuals that it references as role fillers. We now show that COMPUTER-SYSTEM-99 is finished according to Definition 9.

1. By its definition, COMPUTER-SYSTEM-99 instantiates the concrete concept SPARCSTATION20. The instantiation is shown to be bijective according to Definition 6 as follows:

   (a) It is immediately clear that primitives(COMPUTER-SYSTEM-99) $\equiv$ primitives(SPARCSTATION20) because COMPUTER-SYSTEM-99 inherits all the primitives of SPARCSTATION20 and adds no other primitives.

   (b) It is immediately clear that restricted-roles(COMPUTER-SYSTEM-99) $\equiv$ restricted-roles(SPARCSTATION20) because COMPUTER-SYSTEM-99 inherits all the restricted roles of SPARCSTATION20, and restricts no other roles.

   (c) For every restricted role of COMPUTER-SYSTEM-99, it is easy to see that COMPUTER-SYSTEM-99 meets the restrictions placed on that role by

[7] Concrete primitive concepts are introduced by the *define-concrete-primitive-concept* operator, which is otherwise equivalent to the *define-primitive-concept* operator in Figure 2.1.

SPARCSTATION20. Specifically, COMPUTER-SYSTEM-99 just adds one filler to the processor, primary-storage, secondary-storage, and operating system roles inherited from SPARCSTATION20 (which in turn inherits from COMPUTER-SYSTEM). In each case, the filler instantiates SPARCSTATION20's value restriction for the role. The cardinality restrictions are inherited unchanged, except for the secondary-storage role, where the filler (i.e., 1.05GB-DISK-99) induces an at-least restriction of one, which is consistent with the inherited at-least restriction of zero.

2. Every restricted role of COMPUTER-SYSTEM-99 has an at-least restriction of one after completion (which takes the fillers themselves into account). Each restricted role also has a single filler that fulfills its at-least restriction. Without going into excessive detail, it can easily be checked that those fillers (SUN, SUPER-SPARC-II-CPU-99, 64MB-SIMM-99, 1.05GB-DISK-99, and SOLARIS-99) are all themselves finished.

Notice that the number of explicit individual fillers for each role must fall within that role's cardinality restrictions. For example, if the processor role of a certain computer system concept has a minimum cardinality of one and a maximum cardinality of four, then one filler is required; up to three others are permitted but not required. Referring to Figure 3.3 on page 51, the rightmost individual can not be already finished because it does not instantiate a concrete concept. Since the figure does not indicate whether instantiation links are bijective, etc., the other two individuals might or might not be finished at present.

For problem-solving applications like configuration, a well-formed concept taxonomy should respect certain criteria regarding the relationship among abstract and concrete concepts. First, all leaf concepts should be concrete. Since all individuals must eventually be finished in accordance with Definition 9, an abstract concept with no concrete descendants could never be instantiated under CTA. As a simplifying assumption, all non-leaf concepts should be abstract.[8] In configuration, where the very goal is to describe a finished individual system with finished

[8] An argument can be made that concrete concepts other than leaves might occasionally be

```
(define-concrete-primitive-concept COMPUTER-COMPANY
    COMPANY)

(create-individual SUN
    COMPUTER-COMPANY)

(define-concrete-primitive-concept SPARC
    RISC)

(create-individual SUPER-SPARC-II
    SPARC)

(define-concrete-primitive-concept SUPER-SPARC-II-CPU
    (and RISC-CPU
        (fills technology SUPER-SPARC-II)))

(define-concrete-primitive-concept 64MB-SIMM
    RAM)

(define-concrete-primitive-concept 1.05GB-DISK
    DISK)

(define-concrete-primitive-concept SOLARIS
    UNIX)

(define-concrete-primitive-concept SPARCSTATION20
    (and COMPUTER-SYSTEM
        (fills vendor SUN)
        (all processor SUPER-SPARC-II-CPU)))

(create-individual SUPER-SPARC-II-CPU-99
    SUPER-SPARC-II-CPU)

(create-individual 64MB-SIMM-99
    64MB-SIMM)

(create-individual 1.05GB-DISK-99
    1.05GB-DISK)

(create-individual SOLARIS-99
    SOLARIS)

(create-individual COMPUTER-SYSTEM-99
    (and SPARCSTATION20
        (fills processor SUPER-SPARC-II-CPU-99)
        (fills primary-storage 64MB-SIMM-99)
        (fills secondary-storage 1.05GB-DISK-99)
        (fills operating-system SOLARIS-99)))
```

Figure 3.4: Example of a Finished Individual

individual components, etc., every concrete concept should admit a finished instantiation as in Definition 9. These criteria could be enforced by the system. It may also be appropriate to require that the extensions of all concrete concepts (leaves), are mutually disjoint. This property could be enforced in standard description logic by mechanically adding suitable disjointness declarations.

This section has characterized incremental instantiation in description logic as a process directed towards finishing an individual's description. The next section uses these ideas to characterize the configuration problem as it pertains to the description logic component of our configuration system architecture (refer to Figure 3.1 on page 46).

3.3 The Configuration Problem

From the perspective of the description logic system, the configuration problem consists of describing a finished individual computer system. Recall the definition of COMPUTER-SYSTEM123 on page 24 and the set of concepts defined in Figure 2.2. Observe that COMPUTER-SYSTEM123 instantiates SYSTEM and bijectively instantiates COMPUTER-SYSTEM, but is not yet *finished*. To finish COMPUTER-SYSTEM123, we could add suitable role fillers that are themselves finished, including an operating system and one or more memory boards, so it instantiates a concrete subsumee of COMPUTER-SYSTEM (as noted earlier, concrete concepts are not shown in Figure 2.2). A detailed example of a finished system, COMPUTER-SYSTEM-99, was presented on page 53. With respect to configuration, a finished individual system is complete and specific such that it can be ordered from the

convenient, but the knowledge base can always be reformulated so that all concrete concepts are leaves. For example, a concrete MODEL-100 computer system may offer an optional CD-ROM drive, and marketing considerations might lead the vendor to name the version with a CD-ROM drive MODEL-100A. As described, MODEL-100 subsumes MODEL-100A, which is also concrete. Of course, we might define MODEL-100 with (exactly 0 CD-ROM) and MODEL-100A with (exactly 1 CD-ROM), with both specializing a common subsumer. However, some might find this latter domain model less intuitive. In any case, it would be counterintuitive for a concrete concept to subsume an abstract concept.

manufacturer. Of course, it may still be updated, e.g., by adding fillers to a role, subject to the role's at-most restriction.[9]

Some domain-independent assumptions and goals regarding our methodology are considered next.

3.4 Assumptions and Goals

Before plunging into the details of our predictive concept recognition methodology, we pause in this section to carefully review our assumptions and goals. We make two essential assumptions about descriptions:

1. A closed terminology assumption
2. A monotonic update assumption

The CTA expects that all relevant concepts are explicitly defined in the terminology when problem solving begins. We can now define CTA precisely:

Definition 10 *Under the* closed terminology assumption*, it is assumed that no concepts will be added to the terminology during problem solving*[10]*, and that every individual will ultimately be finished according to Definition 9.*

An application may not define any additional concept, even indirectly, e.g., as the conjoined value restriction of an individual's role. CTA further implies that all coherent (satisfiable) roles of a concrete concept are explicitly restricted. Thus, we will assume that all roles not explicitly restricted by a concrete concept are

[9]Upgrades over time are an interesting topic for future work.

[10]Although no concepts may be added by the *application*, we will relax this assumption so the system can add concepts strictly for its own internal use, e.g., in Section 3.6 to improve efficiency. Also, note that in some description logic systems, including K-REP, concept description can be modified (and perhaps reclassified) after they are created. The CTA also forbids modification of concepts.

unsatisfiable.[11] Consistency under CTA is different than under OTA. CTA is essential for ruling out all concepts that an individual cannot instantiate, and thus drawing conclusions from the remainder. Section 3.9 compares CTA with OTA in more detail. CTA applies only at the level of concepts and should not be confused with a closed world assumption, which would apply at the level of individuals. We neither assume that the set of individuals is closed, nor do we attempt closed world reasoning over individuals (which was treated in a description logic context by [Donini *et al.*, 1992]).

CTA is just right for applications like configuration and plan recognition (see Chapter 5), where domain modeling is completed prior to problem solving.[12] By Definition 9, once an individual is finished, it will bijectively instantiate at least one concrete concept. Such concepts are referred to as *ultimate* concepts:

Definition 11 *Given that an individual, when finished, will bijectively instantiate one or more concrete concepts, those concepts are its* ultimate *concepts.*

Note that every unfinished individual can become finished via monotonic updates. While an ultimate concept restricts every property of an individual, it may do so at an abstract level, in the same sense that COMPUTER-SYSTEM requires a processor, but only constrains it to be some CPU.[13] Furthermore, when a role of an ultimate concept does not have a positive at-least restriction, the individual may decline to fill that role by specifying an at-most restriction of zero, e.g., (at-most 0 SECONDARY-STORAGE) for a diskless workstation.

We also assume, provisionally, that updates to an individual will be monotonic:

Definition 12 *Under the* monotonic update assumption*, it is assumed that all updates of an individual will entail adding base concepts and/or adding role restrictions (introducing a restriction on a role or tightening an existing one).*

[11] Alternatively, we could just assume that the set of roles is closed. Then for every role R and for every concept C that does not explicitly restrict R, the system could add an at-most restriction of zero on R_C.

[12] Configuration stands in contrast with *design*, where the task involves extending the domain.

[13] COMPUTER-SYSTEM itself can not be an ultimate concept, since it is not concrete.

This assumption is essential because it licenses conclusions based on the individual's current description. Provided that an individual is updated monotonically, CTA guarantees that it is always CTA-consistent with its ultimate concept(s). The incremental recognition process can be seen as continually narrowing down the set of concepts that may turn out to be ultimate concepts. Still, if our monotonic update assumption proves unfounded, we can back out of it gracefully (see Section 3.7).

The first major goal of our predictive recognition methodology is to track the potential instantiation relationships between an individual and the concepts in the terminology while the individual is being specified. Given our assumptions, a concept can be categorized as either necessary, optional, or impossible, relative to an individual's current description:

Definition 13 *Concept* C *is* necessary *with respect to individual* I *iff* I *instantiates* C*.*

Definition 14 *Concept* C *is* optional *with respect to individual* I *iff* I *does not instantiate* C *but can be monotonically updated to do so.*

We will say that a concept is *possible* if it is either necessary or optional. Otherwise it is impossible:

Definition 15 *Concept* C *is* impossible *with respect to individual* I *iff* I *neither instantiates* C *nor can be monotonically updated to do so.*

The next section details inferences for performing this triage. Section 3.6 proposes an optimization that further exploits the subsumption-based terminology, and Section 3.7 presents our overall recognition methodology. Another major goal is to exploit the necessary / optional / impossible trichotomy to provide information and to derive additional constraints on an individual from the commonality among its optional concepts. That aspect of the work is covered in Section 3.8.

3.5 Closed Terminology Consistency

During configuration, we want to know which concepts are CTA-consistent with a partially described individual, and which are not, so we can distinguish between its possible and impossible concepts. CTA is vital to our methodology: CTA holds that an individual must eventually be finished in accord with Definition 9. Thus, an individual can be monotonically updated to instantiate concept C under CTA iff it can be monotonically updated to bijectively instantiate an explicitly defined concept (perhaps C itself) when it does so. We are also concerned with CTA-consistency between concepts, both in this section, to help determine CTA-consistency of value restrictions on roles, and in Section 3.6, to help speed predictive recognition. The remainder of this section focuses on how to determine (1) when two concepts are consistent under CTA, and (2) when an individual is consistent with a concept under CTA.

We decide CTA-consistency via the mutually recursive definitions that follow. For a pair of concepts, we must consider two cases. First, a pair of concepts are directly consistent under CTA whenever a bijective instantiation of one can also instantiate the other. To capture this case, we introduce a *direct consistency* inference from concept *C1* to concept *C2*, written *C1* $\mapsto$ *C2*:

Definition 16 C1 $\mapsto$ C2 *iff*

1. *primitives(*C1*)* $\subseteq$ *primitives(*C2*)*
2. *No primitive of* C1 *is disjoint from any (additional) primitive of* C2
3. *restricted-roles(*C1*)* $\subseteq$ *restricted-roles(*C2*)*
4. *For every role* R *on restricted-roles(*C1*),* $\mathrm{R_{C1}}$ *and* $\mathrm{R_{C2}}$ *are CTA-consistent*

Observe that *C2* must have at least as many primitives and restricted roles as *C1*. Intuitively, *C1* $\mapsto$ *C2* demonstrates that the primitives and role restrictions of *C2* admit a bijective instantiation of *C2* which also instantiates *C1*. Although subsumption implies direct consistency, the converse is not true. For example, IBM-PROCESSOR-DEVICE $\mapsto$ COMPUTER-SYSTEM; even though neither one subsumes the other, some individual computer systems may have IBM-CPU processors (role consistency under CTA will be defined shortly). Direct CTA-consistency between concepts can occur in either direction:

Definition 17 *Concepts* C1 *and* C2 *are* directly CTA-consistent *iff* C1 $\mapsto$ C2 *or* C2 $\mapsto$ C1*.*

We must also define CTA-consistency for roles:

Definition 18 *Roles* $\mathrm{R_X}$ *and* $\mathrm{S_Y}$ *are* CTA-restriction-consistent *iff*

1. *Their cardinality restrictions intersect*
2. *If at-least(*$\mathrm{R_X}$*)* > 0 *or at-least(*$\mathrm{S_Y}$*)* > 0*, then value-restriction(*$\mathrm{R_X}$*) and value-restriction(*$\mathrm{S_Y}$*) are CTA-consistent*
3. $|$ *fillers(*$\mathrm{R_X}$*)* $\cup$ *fillers(*$\mathrm{S_Y}$*)* $| \leq$ *minimum(at-most(*$\mathrm{R_X}$*), at-most(*$\mathrm{S_Y}$*))*
4. *Every filler of* $\mathrm{R_X}$ *is CTA-consistent with value-restriction(*$\mathrm{S_Y}$*), and every filler of* $\mathrm{S_Y}$ *is CTA-consistent with value-restriction(*$\mathrm{R_X}$*)*

The third condition of Definition 18 checks that the combined fillers of the two roles do not exceed the maximum cardinality restriction of either one. Notice that the combined fillers need not meet the minimum cardinality restriction of either role: An individual can be consistent with both roles simultaneously if it has a consistent at-least restriction but no fillers (however, it would need to acquire sufficient fillers before being considered finished).

For the remainder of this chapter, we are only concerned with restriction consistency of roles having the same name.[14] Therefore, we define *CTA-consistency* as

[14]However, in Chapter 5, restriction consistency will also figure in direct consistency inferences with respect to co-reference constraints on differently named roles.

a simplified special case of CTA-restriction-consistency where the roles in question are identically named:

Definition 19 *Roles* $\mathrm{R_X}$ *and* $\mathrm{R_Y}$ *are* CTA-consistent *iff they are CTA-restriction-consistent.*

When R_X and R_Y are *CTA-consistent*, their restrictions can be simultaneously satisfied under CTA. Clearly, the processor roles of UNIPROCESSOR-SYSTEM and UNIX-RISC-SYSTEM are CTA-consistent, as are their operating-system roles. In addition, both concepts inherit their primitives and their remaining role restrictions from a common source, COMPUTER-SYSTEM, so UNIPROCESSOR-SYSTEM $\mapsto$ UNIX-RISC-SYSTEM (and *vice versa*). We conclude that they are CTA-consistent. This conclusion will be used in Section 3.6.

There is a second, indirect case of CTA-consistency between concepts. Even if two concepts are not directly consistent, their consistency may still be explicitly sanctioned by a third concept consistent with both. This situation can arise when each concept has a primitive or restricted role that the other lacks. For example, IBM-CPU lacks a technology role and RISC-CPU lacks a vendor role. Therefore, only from the existence of a concept such as IBM-RISC-CPU can we conclude that they are CTA-consistent. As a result, we have:

Definition 20 *Concepts* C1 *and* C2 *are indirectly* CTA-consistent *iff there exists an explicitly defined concept* C3 *such that*

1. C1 $\mapsto$ C3
2. C2 $\mapsto$ C3
3. *For every role* R *restricted by both* C1 *and* C2*,* $\mathrm{R_{C1}}$ *and* $\mathrm{R_{C2}}$ *are CTA-consistent*

The first two clauses of Definition 20 ensure that *C1* and *C2* are separately consistent with *C3*. The third clause ensures that wherever restrictions on *C1* and *C2* interact, they do so in a mutually consistent way. Hence, an instance of *C3* can simultaneously instantiate *C1* and *C2*. In particular, it can be seen that IBM-CPU (*C1*) and RISC-CPU (*C2*) are indirectly CTA-consistent through the existence of IBM-RISC-CPU (*C3*):

1. IBM-CPU $\mapsto$ IBM-RISC-CPU (a subsumee of IBM-CPU)
2. RISC-CPU $\mapsto$ IBM-RISC-CPU (a subsumee of RISC-CPU)
3. IBM-CPU and RISC-CPU restrict no roles in common

As a special case, the indirect CTA-consistency of IBM-CPU and RISC-CPU could have been concluded immediately from the existence of their explicit common subsumee, IBM-RISC-CPU. This can not be done in general. For example, SYSTEM and IBM-PROCESSOR-DEVICE are not directly consistent.[15] Still, it can be seen that SYSTEM (*C1*) and IBM-PROCESSOR-DEVICE (*C2*) are indirectly CTA-consistent through the existence of COMPUTER-SYSTEM (*C3*):

1. SYSTEM $\mapsto$ COMPUTER-SYSTEM (a subsumee of SYSTEM)
2. IBM-PROCESSOR-DEVICE $\mapsto$ COMPUTER-SYSTEM (as shown earlier in this section)
3. There are no roles restricted by both SYSTEM and IBM-PROCESSOR-DEVICE

Putting the direct and indirect cases together, CTA-consistency identifies pairs of concepts that can be instantiated simultaneously:

Definition 21 *Concepts* C1 *and* C2 *are* CTA-consistent *iff they are directly or indirectly CTA-consistent.*

This definition is justified as follows:

[15] SYSTEM $\not\mapsto$ IBM-PROCESSOR-DEVICE because primitives(SYSTEM) $\equiv$ {SYSTEM} and SYSTEM $\notin$ primitives(IBM-PROCESSOR-DEVICE). IBM-PROCESSOR-DEVICE $\not\mapsto$ SYSTEM because SYSTEM lacks a processor role.

Theorem 2 *Under CTA, the extensions of concepts* C1 *and* C2 *intersect iff they are CTA-consistent.*

Proof: See Appendix B.

Now we examine CTA-consistency between an individual and a concept. We again distinguish between direct and indirect cases. A *direct consistency* inference from individual I to concept C, written $I \mapsto C$, essentially duplicates Definition 16:

Definition 22 I $\mapsto$ C *iff*

1. *primitives(*I*)* $\subseteq$ *primitives(*C*)*
2. *No primitive of* I *is disjoint from any (additional) primitive of* C
3. *restricted-roles(*I*)* $\subseteq$ *restricted-roles(*C*)*
4. *For every role* R *on restricted-roles(*I*),* $\mathrm{R_I}$ *and* $\mathrm{R_C}$ *are CTA-consistent*

Direct consistency of an individual and a concept follows immediately:

Definition 23 *Individual* I *and concept* C *are* directly CTA-consistent *iff* I $\mapsto$ C.

Intuitively, direct consistency of an individual with a concept establishes that the individual can bijectively instantiate the concept, perhaps after monotonic updates. Under CTA and our monotonic update assumption, an individual is always directly consistent with its ultimate concept(s). Consider this individual:

```
(create-individual COMPUTER-SYSTEM45
    (and COMPUTER-SYSTEM
        (all processor RISC-CPU)))
```

Although COMPUTER-SYSTEM45 is directly consistent with DUAL-IBM-PROCESSOR-SYSTEM[16] and also with RISC-MULTIPROCESSOR-SYSTEM[17], it instantiates neither: regarding DUAL-IBM-PROCESSOR-SYSTEM, its processor role is neither restricted to fillers of type IBM-CPU nor a cardinality of exactly 2, and regarding RISC-MULTIPROCESSOR-SYSTEM, the cardinality of its processor role is not known to be at least 2. However, monotonic updates might still add those restrictions later.

In the context of a terminology, the fact that an individual is directly consistent with some concept may imply consistency with other concepts. Clearly, individual I is indirectly CTA-consistent with concept C if there exists a concept C' such that $I \mapsto C'$ and $C' \Rightarrow C$, i.e., if I can be monotonically updated to instantiate C' then it can be monotonically updated to instantiate any concept that subsumes C'. For example, COMPUTER-SYSTEM45 is directly consistent with DUAL-IBM-PROCESSOR-SYSTEM, but not with IBM-PROCESSOR-DEVICE. However, COMPUTER-SYSTEM45 is indirectly consistent with IBM-PROCESSOR-DEVICE because the latter subsumes DUAL-IBM-PROCESSOR-SYSTEM. Not all indirectly consistent concepts can be identified via subsumption (but see Section 3.6). Consider:

```
(create-individual COMPUTER-SYSTEM67
    (and COMPUTER-SYSTEM
        (exactly 1 processor)))
```

We can not make a direct consistency inference from COMPUTER-SYSTEM67 to IBM-PROCESSOR-DEVICE[18] or its one and only subsumee, DUAL-IBM-PROCESSOR-

[16] Referring to Definition 22, COMPUTER-SYSTEM45 has the same primitives and restricted roles as DUAL-IBM-PROCESSOR-SYSTEM (both inherit from COMPUTER-SYSTEM). Their restrictions differ only for the processor role: COMPUTER-SYSTEM45 has at least one processor which is a RISC-CPU, while DUAL-IBM-PROCESSOR-SYSTEM has exactly two processors which are IBM-CPUs. Considering Definition 18, it is easy to see that restrictions on those roles are consistent (we showed earlier that RISC-CPU and IBM-CPU are CTA-consistent due to the presence of IBM-RISC-CPU in the terminology).

[17] Referring to Definition 22, COMPUTER-SYSTEM45 has the same primitives and restricted roles as RISC-MULTIPROCESSOR-SYSTEM (both inherit from COMPUTER-SYSTEM). Their restrictions differ only for the processor role: COMPUTER-SYSTEM45 has at least one processor which is a RISC-CPU, while RISC-MULTIPROCESSOR-SYSTEM has at least two such processors. Those restrictions are obviously in accord with Definition 18.

[18] e.g., primitives(COMPUTER-SYSTEM67) $\not\subseteq$ primitives(IBM-PROCESSOR-DEVICE)

SYSTEM[19]. Nonetheless, we can monotonically update the definition of COMPUTER-SYSTEM67 to directly instantiate UNIPROCESSOR-SYSTEM such that it also instantiates IBM-PROCESSOR-DEVICE, as follows:

(and COMPUTER-SYSTEM
    (exactly 1 processor)
    (all processor IBM-CPU))

To generalize, if individual I can potentially instantiate concept C' such that it also instantiates concept C, then I is *indirectly consistent* with C via C':

Definition 24 *Individual* I *and concept* C *are* indirectly CTA-consistent *iff there exists a concept* C′ *such that:*

1. $\mathrm{I} \mapsto \mathrm{C}'$
2. $\mathrm{C} \mapsto \mathrm{C}'$
3. *For every role* R *restricted by both* I *and* C*,* $\mathrm{R_I}$ *and* $\mathrm{R_C}$ *are CTA-consistent*

Either directly or indirectly, CTA-consistency identifies the concepts an individual might instantiate once it is finished:

Definition 25 *Individual* I *and concept* C *are* CTA-consistent *iff they are directly or indirectly CTA-consistent.*

We say that I potentially instantiates C when they are CTA-consistent. CTA-consistency is essential to our overall recognition methodology, to be presented in Section 3.7. Its correctness is established by the following:

Theorem 3 *Under CTA, individual* I *can be monotonically updated to instantiate concept* C *iff* I *and* C *are CTA-consistent.*

[19]The cardinality restrictions on their processor roles don't intersect.

Proof: See Appendix B.

The inferences presented in this section allow us to distinguish between optional and impossible concepts under CTA. Section 3.6 shows how precomputation permits all indirect cases of CTA-consistency to be discovered via subsumption.

3.6 Terminology Augmentation

To implement indirect consistency testing straight from Definition 24 would mean repeated searches at run-time for a concept C' through which indirection is licensed. The problem stems from the possibility that two consistent concepts may have extensions that partially overlap, i.e., neither concept subsumes the other. This section proposes an improvement based on preprocessing. Our goal is to minimize inference during recognition. Therefore, we will trade space for time when figuring consistency relationships between an individual and the concepts in a terminology. Notice that whenever individual I instantiates concepts C and C' simultaneously, it must instantiate their conjunction. This insight will allow us to quickly identify all cases of indirect consistency by traversing explicit subsumption links in the taxonomy. We can augment the terminology with concepts for internal use[20] as follows:

Definition 26 *A terminology is* augmented *if for all concepts* C1 *and* C2 *such that* C1 $\mapsto$ C2*, there exists an explicitly defined concept* C3 *such that* C3 $\equiv$ C1 $\wedge$ C2*.*

We will discuss how to improve on Definition 26 later in this section. When *C1* subsumes or is subsumed by *C2*, we do nothing because the subsumee *is* their conjunction. Otherwise, the relationship among *C1*, *C2*, and *C3* is diagrammed in Figure 3.5.

[20]These auxiliary concepts are used internally by K-REP and are not considered part of the domain model. Thus, they do not violate the CTA.

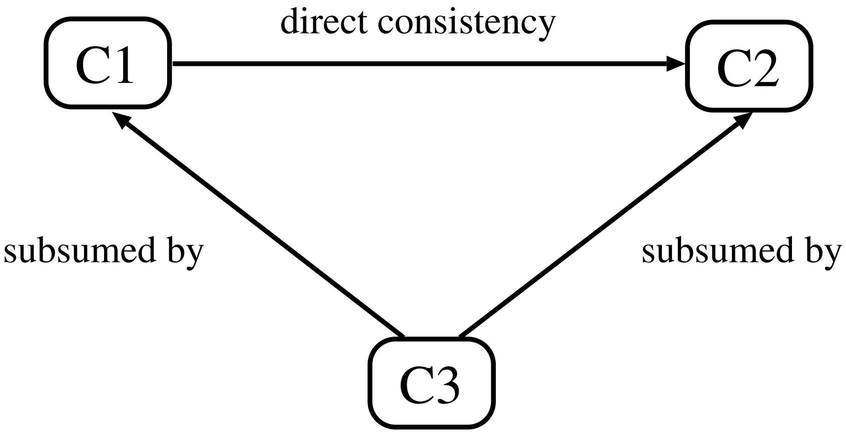


Figure 3.5: Terminology Augmentation

For example, recalling that UNIPROCESSOR-SYSTEM (*C1*) $\mapsto$ UNIX-RISC-SYSTEM (*C2*), we would add their conjunction (*C3*) defined as follows:

(and COMPUTER-SYSTEM
    (the processor RISC-CPU)
    (the operating-system UNIX))

Observe that its processor role reflects a value restriction of RISC-CPU from UNIX-RISC-SYSTEM and a cardinality restriction of exactly one from UNIPROCESSOR-SYSTEM.

In an augmented terminology, conjunctions of directly CTA-consistent concepts are made explicit as system-defined concepts when the user has not already defined them. Thereafter, two concepts are CTA-consistent just in case they have a common subsumee (subsumption is reflexive). Consequently, we can reduce the indirect aspect of consistency testing to subsumption checking:

Definition 27 *Individual* I *is* indirectly CTA-consistent *with concept* C *in an augmented terminology iff there exists a concept* C$'$ *such that* I $\mapsto$ C$'$ *and* C$'$ $\Rightarrow$ C*.*

Readers may note that $C' \Rightarrow C$ in this definition, while $C \mapsto C'$ in Definition 24; both are correct. With an augmented terminology, Definition 27 produces correct results:

Theorem 4 *Under CTA and with an augmented terminology, individual* I *can be monotonically updated to instantiate concept* C *iff* I *and* C *are CTA-consistent using Definition 27 for indirect consistency instead of Definition 24.*

Proof: See Appendix B.

Recall our assumption that an individual is always directly consistent with one or more ultimate concepts, which are concrete concepts according to Definition 9. This assumption means that augmentation as in Definition 26 can be limited to cases where *C2* is a concrete concept. We need only augment a terminology once, after its development concludes and before problem solving begins. An augmented terminology offers simple, fast identification of indirectly consistent concepts by traversing explicit subsumption links. In fact, after augmentation, we can cache consistency relationships for even faster retrieval. For example, consistency among n concepts could be encoded in an n x n bit array. The possible drawback of augmentation is proliferation of system-defined concepts, which in the worst case could exponentially increase the size of the terminology. Such concepts can be rendered invisible to the user. The real issue is impact on performance. The number of system-defined concepts in an augmented terminology should be manageable in practice, assuming sufficiently distinct primitiveness and role restrictions among user-defined concepts.

In this section, we have shown how to augment a terminology to speed the identification of concepts that are indirectly consistent with an individual, a task which is essential in identifying every optional concept. Identifying optional concepts, in turn, is a crucial part of our predictive recognition process. The next section explains how we conduct that process by partitioning a terminology with respect to an individual.

3.7 Recognition via Terminology Partitioning

Given an individual I, a knowledge base, and our assumptions, predictive recognition determines every concept's status, or *modality*, with respect to I. Recall from Section 3.4 that concept C is *necessary* with respect to I if I instantiates C, else *optional* if I can be monotonically updated to instantiate C, else *impossible*. When C is optional, it is possible but not necessary that I will ultimately instantiate C. These definitions implicitly partition the taxonomy into regions as shown in Figure 3.6, where necessary, optional, and impossible concepts are black, gray, and white, respectively. Such a partition constitutes the *recognition state* of I. Different individuals have distinct recognition states. Our objective is to efficiently bound the portion of the taxonomy containing optional concepts, thereby limiting the number of concepts that must be directly compared with the individual (empirical performance results will be presented in Section 3.10). We record the individual's recognition state in a manner which leads naturally to further benefits. As we will see later in this section, it enables us to prompt the user with a succinct and appropriate set of choices for refining the description of a given individual. Furthermore, in the next section, we will exploit the recognition state to derive implicit constraints on the individual from the terminology.

As an individual is incrementally updated, we can track its recognition state. Initially, all concepts are optional except the vacuous root concept, THING, which is trivially necessary. The monotonic update assumption implies that each update will change the modality of zero or more optional concepts to necessary or impossible. Referring to Figure 3.6, observe that the necessary and impossible regions expand as the individual is updated, further confining the optional region. Finally, (at least) one concrete concept and its ancestors are necessary; the remainder are impossible. (It is straightforward to test if an update is nonmonotonic. When that happens, we may need to re-expand the optional region. While, we only elaborate here on monotonic updates, we see no difficulty in handling nonmonotonic updates.) We exploit the terminology's subsumption-based organization in

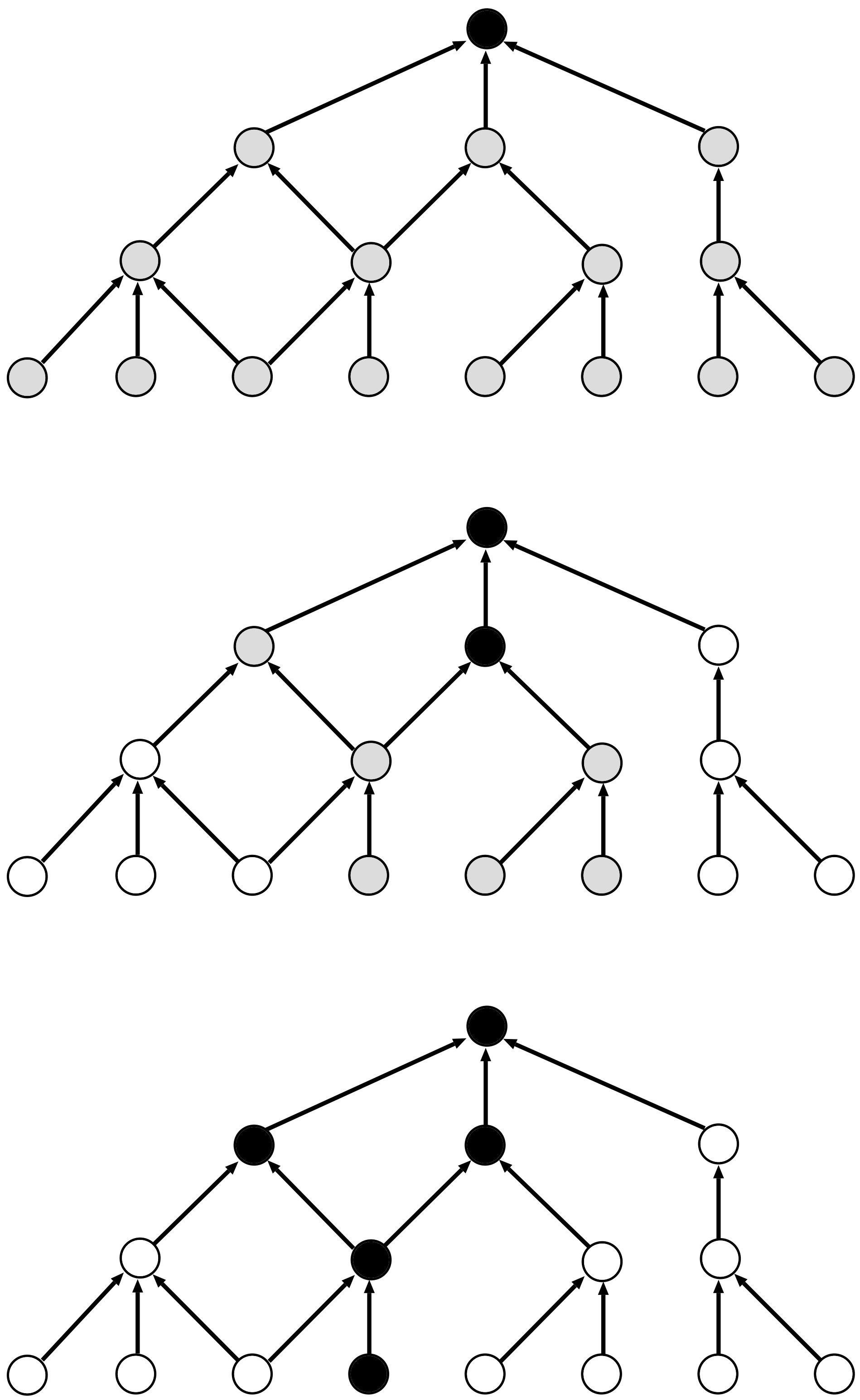


Figure 3.6: Initial, Intermediate, and Final Recognition States (top to bottom)

two ways. First, the current partition is used to compute the next one. Second, the consequences of comparing an individual with a concept are propagated to other concepts, so those other concepts need not be directly compared with the individual. To these ends, we distinguish three sets of concepts:

1. The most specific necessary concepts, or MSNs.

2. The most general optional concepts, or MGOs.

3. The most specific optional concepts, or MSOs.

Membership in these sets is determined according to the following definitions:

Definition 28 *A concept is an* MSN *concept iff it is necessary and none of its children are necessary.*

Definition 29 *A concept is an* MGO *concept iff it is optional and none of its parents are optional.*

Definition 30 *A concept is an* MSO *concept iff it is optional and none of its children are optional.*

The frontiers of the optional region are maintained by the MGOs and the MSOs. Those sets need not be disjoint. The MSNs serve to speed computation of the MSOs. If we require that all concrete concepts are leaves in the concept taxonomy, as we do in this thesis, then all MSO concepts must be leaves.[21]

Initially, the MSN set contains the root concept THING, the MGO set contains THING's children, and the MSO set contains the leaf concepts. When an individual is updated, our algorithm operates in three successive phases to update the MSNs,

[21]Again, recall our assumption that an individual is directly consistent with one or more ultimate concepts, which are concrete concepts according to Definition 11.

MSOs and MGOs, respectively. The first phase discovers all newly necessary concepts, the second all newly impossible ones. The third simply prepares the new MGOs for the MSN phase of the next round of incremental recognition. Details of each phase are given later in this section. We associate a modality with each concept in the individual's recognition state and only update the modality of a concept when required by an update to the individual. In particular, individuals are never compared with concepts already know to be necessary or impossible. After an example, we will present a sample algorithm for incremental update of the MSN, MSO, and MGO sets.

Turning to an example, an initial user-specified configuration might be:

```
(create-individual COMPUTER-SYSTEM89
    (and COMPUTER-SYSTEM
        (at-least 2 processor)
        (at-least 1 secondary-storage)))
```

For this example, we will consider the terminology of Figure 2.2 on page 20, but with IBM-PROCESSOR-DEVICE *omitted.* Then, the necessary concepts are THING, SYSTEM, and COMPUTER-SYSTEM. The set of optional concepts includes DUALPROCESSOR-SYSTEM, DUAL-IBM-PROCESSOR-SYSTEM, UNIX-RISC-SYSTEM, and RISC-MULTIPROCESSOR-SYSTEM. The remainder are impossible. This recognition state is captured as[22]:

$$\begin{aligned}
\text{MSNs} &= \{\text{COMPUTER-SYSTEM}\}\\
\text{MGOs} &= \{\text{DUALPROCESSOR-SYSTEM},\\
&\quad\ \text{UNIX-RISC-SYSTEM},\\
&\quad\ \text{RISC-MULTIPROCESSOR-SYSTEM}\}\\
\text{MSOs} &= \{\text{DUAL-IBM-PROCESSOR-SYSTEM},\\
&\quad\ \text{UNIX-RISC-SYSTEM},\\
&\quad\ \text{RISC-MULTIPROCESSOR-SYSTEM}\}
\end{aligned}$$

[22]Ordinarily, the MSOs would of course be concrete concepts, but recall that we omitted concrete concepts from our sample taxonomy for brevity.

This recognition state indicates precisely which concepts COMPUTER-SYSTEM89 might eventually instantiate. Notice that UNIPROCESSOR-SYSTEM and DISKLESS-SYSTEM are ruled out, as are many other concepts, e.g., COMPANY is impossible. These conclusions follow from CTA and the monotonic update assumption. If the user now states that COMPUTER-SYSTEM89 should have exactly four processors, the subsequent recognition state will also rule out DUALPROCESSOR-SYSTEM and DUAL-IBM-PROCESSOR-SYSTEM:

$$\begin{aligned} \text{MSNs} &= \{\text{COMPUTER-SYSTEM}\} \\ \text{MGOs} &= \{\text{UNIX-RISC-SYSTEM, RISC-MULTIPROCESSOR-SYSTEM}\} \\ \text{MSOs} &= \{\text{UNIX-RISC-SYSTEM, RISC-MULTIPROCESSOR-SYSTEM}\} \end{aligned}$$

Notice that the MGOs succinctly capture the most general choices available in the current state. A user interface can prompt the user to choose a subset of the MGOs which describe the desired system. In Section 3.8, we will also use this recognition state to infer new information about COMPUTER-SYSTEM89.

We conclude this section by further detailing a sample algorithm for incremental update of an individual's recognition state, followed by an example of the algorithm in operation. This algorithm assumes that the terminology has already been augmented. As mentioned, there are three successive phases to update the MSN, MSO, and MGO sets, respectively.

MSN Phase: We begin with the MSN phase so we can exploit necessary conditions on the ultimate concept(s) in the subsequent MSO phase. We consider each member of the existing MSN set in turn, searching recursively downward in depth-first order through its descendants. Upon visiting a descendant, we determine if that concept is instantiated by the individual. If so, we change that concept's modality from optional to necessary and continue downward. When we encounter a necessary concept with no necessary children, we have discovered a member of the new MSN set. Note, in our subsumption-based taxonomy, all ancestors of a necessary concept must be necessary. Since the taxonomy is a directed acyclic graph, we then mark as necessary any parents of the new MSN concept not already

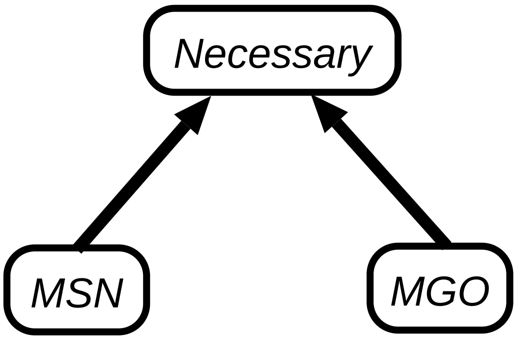


Figure 3.7: MGO Concept Not Subsumed By Any MSN Concept

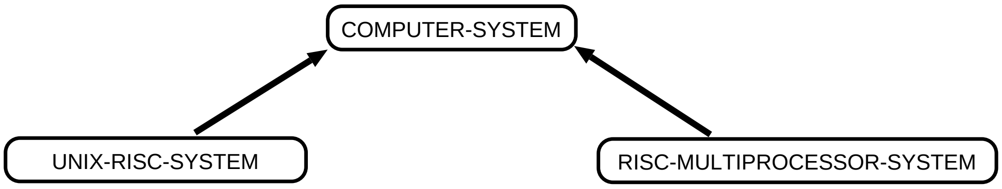


Figure 3.8: Specific MGO Concept Not Subsumed By Any MSN Concept

marked necessary, continuing recursively upward as required.[23] We must also examine members of the existing MGO set because they may have become necessary, yet they may not have been subsumed by *any* MSN concept. This situation is illustrated in Figure 3.7, where the arrows indicate immediate child-to-parent links. If such a concept is now necessary, we go on to treat it as if it had been a member of the previous MSN set.

For example, given the terminology of Figure 2.2 on page 20, suppose we have an individual which instantiates UNIX-RISC-SYSTEM and has at least one processor such that UNIX-RISC-SYSTEM is an MSN concept. Then COMPUTER-SYSTEM is also necessary (but not an MSN), and RISC-MULTIPROCESSOR-SYSTEM is an MSO concept. The relationship among these concepts, extracted from Figure 2.3, is shown in Figure 3.8. If we add a second processor to that individual, RISC-MULTIPROCESSOR-SYSTEM becomes necessary.

[23]Such a concept could be revisited when searching down from another MSN concept or as we are about to see, from an MGO concept. Then we need not prove its necessity again, however it may still have unvisited children to be tested.

MSO Phase: Since concepts, and MGO concepts in particular, may be indirectly consistent with the individual via MSO concepts, we must update the MSO set before the MGO set. MSO concepts are the most specific concepts in the taxonomy that could be the ultimate concept(s); they can only be *directly* consistent with the individual. Updating the MSO set inherently requires bottom-up search from each existing MSO concept. Thus, updating MSO sets in concept taxonomies with greater breadth will tend to be more expensive. Fortunately, because MSN concepts represent necessary constraints on the individual, we know that each MSO concept must be subsumed by *every* MSN concept (in an augmented terminology). Moreover, we can efficiently identify concepts subsumed by every MSN concept and mark them accordingly[Dionne, 1991]. When visiting a concept in this phase, there are three cases:

1. It may have been marked necessary during the MSN phase,
2. it may have become impossible, or
3. it may remain an MSO concept.

In the first case, we go on to examine any remaining members of the previous MSO set. We must work to distinguish between cases 2 and 3. First, we check if the concept is subsumed by all necessary concepts, and if not, we immediately mark it impossible. Otherwise, we attempt to make a direct consistency inference from the individual. When the attempt is successful, we place the concept in the new MSO set and go on to examine any remaining members of the previous MSO set. When it fails, the concept is impossible. With an impossible concept, we may continue to search upward because a concept may be optional even though all its children are impossible.[24] However, since it is also true that a concept is impossible only if all of its children are impossible, if we now visit a concept with a child still marked optional, we do nothing at this time.[25]

[24]We search upward only when concrete concepts need not be leaves in the concept taxonomy; assuming that we require them to be leaves, as we did earlier, no upward search is needed.

[25]Should we revisit it via another concept in the previous MSO set, the result may differ.

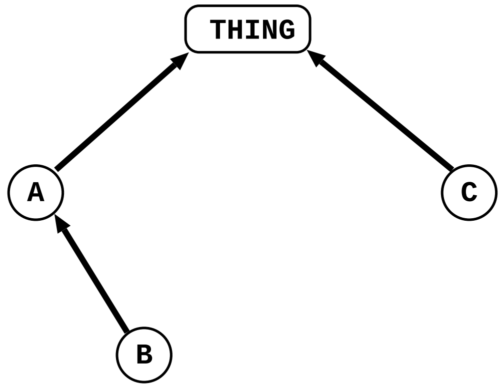


Figure 3.9: Example Concept Taxonomy

MGO Phase: Finally, we update the MGO set. The MSN phase may have discovered necessary concepts included in or subsumed by members of the previous MGO set. The MSO phase may have found impossible concepts extending up to and including members of the MGO set. Therefore, we search downward from each concept in the previous MGO set, proceeding past any concepts now marked necessary and stopping at concepts marked either optional or impossible. Such a concept is placed in the new MGO set if and only if it is marked optional and all its parents are marked necessary. Observe that this phase never changes the modality of any concept.

As an example, consider the simple concept taxonomy shown in Figure 3.9. Before considering the description of some individual I, THING is necessary and all other concepts are optional. Thus, the recognition state of I is initialized as:

$$\begin{aligned} \text{MSNs} &= \{\text{THING}\} \\ \text{MGOs} &= \{\text{A, C}\} \\ \text{MSOs} &= \{\text{B, C}\} \end{aligned}$$

Now, suppose that I is described such that it instantiates only A and THING, is directly consistent with B, and is not consistent with C. The MSN phase starts with the existing MSN set, in this case only THING. Its first child, A, is visited and marked necessary because I instantiates A. Continuing downward, A's only child B is visited and left optional because I does not instantiate it. Since A has no necessary children, it is placed in the new MSN set. The MSN phase continues with the existing MGO set, in this case A and C. A is processed as before, with no new results. C is visited and left optional. Thus, the new MSN set consists of A.

The MSO phase begins by determining those members of the existing MSO set subsumed by every member of the new MSN set; only B meets this criterion, because C is not subsumed by A. The members of the existing MSO set, B and C, are then processed. Since B is directly consistent with I, it is left optional and placed in the new MSO set. Since it was determined that C is not subsumed by every new MSN concept, it is marked impossible.

The MGO phase starts with the existing MGO set, which contains A and C. First, A has been marked necessary, so we proceed to its child B. Since B is marked optional and its only parent (A) is marked necessary, it is placed in the new MGO set. Second, C has been marked impossible, so we do nothing.

As a result of the preceding process, the new recognition state of I is:

$$\begin{aligned} \text{MSNs} &= \{\text{A}\} \\ \text{MGOs} &= \{\text{B}\} \\ \text{MSOs} &= \{\text{B}\} \end{aligned}$$

Whenever an individual's description is updated, corresponding changes to its recognition state may be in order. Of course, other individuals may have descriptions that reference the updated individual, in which case it may be necessary to revise their recognition states as well.

The partition of a terminology for an individual shows what classes of choices can still be made for that individual and what choices cannot, i.e., the optional and impossible concepts, respectively. In the next section, further use of the partition is made.

3.8 Constraint Derivation

We now take advantage of predictive recognition to derive additional constraints (primitives and/or role restrictions) on an individual. Suppose the terminology is partitioned according to the current description of some individual, I. If I does not bijectively instantiate an MSN concept that is concrete, then by CTA, I will ultimately instantiate some optional concept. Therefore, the commonality among MGO concepts constitutes a set of implicit constraints imposed on I by the closed terminology. Recall from Section 2.1.8 that Cohen, et al., showed how to compute the *least common subsumer* (LCS) of a set of description logic concepts [Cohen *et al.*, 1992]. The LCS of the MGOs, written LCS(MGOs), is a concept representing just their commonality (we never use the LCS concept for any other purpose, so it is not permanently installed in the terminology). Any constraint on LCS(MGOs) not reflected in I should be added to I. The basic strategy is:

- Until arriving at a fixed point, repeatedly
    1. Compute LCS(MGOs)
    2. If LCS(MGOs) implies further constraints on I
        (a) Fold LCS(MGOs) into the description of I, and
        (b) Incrementally update the recognition state of I

Since each iteration of this process can add but never remove restrictions, the process must eventually terminate. Thus, K-REP may be able to infer constraints on I after it is first created and whenever it is updated. Similar reasoning applies recursively to fillers of I's roles.

Recall the second recognition state for COMPUTER-SYSTEM89 in Section 3.7:

$$\begin{aligned}
\text{MSNs} &= \{\text{COMPUTER-SYSTEM}\} \\
\text{MGOs} &= \{\text{UNIX-RISC-SYSTEM, RISC-MULTIPROCESSOR-SYSTEM}\} \\
\text{MSOs} &= \{\text{UNIX-RISC-SYSTEM, RISC-MULTIPROCESSOR-SYSTEM}\}
\end{aligned}$$

Note that the only MSN concept, COMPUTER-SYSTEM, is not a concrete concept. Hence, COMPUTER-SYSTEM89 will eventually instantiate at least one MGO concept. LCS({UNIX-RISC-SYSTEM, RISC-MULTIPROCESSOR-SYSTEM}) is:

```
(and COMPUTER-SYSTEM
     (all processor RISC-CPU))
```

Hence, K-REP discovers that COMPUTER-SYSTEM89's processors must be of type RISC-CPU. This conclusion can only come from closed terminology reasoning. As a result, future choices regarding the processors will be suitably constrained.

For an example where a primitive is inferred by CTA reasoning, refer to Figure 2.2 and note that the only immediate subsumees of SYSTEM are OS and COMPUTER-SYSTEM. Now consider an individual SYSTEM with a processor role:

```
(create-individual SYSTEM321
   (and SYSTEM
        (all processor IBM-CPU)))
```

As described, SYSTEM321 can not be classified as a COMPUTER-SYSTEM, because COMPUTER-SYSTEM is not among its primitives. However, for SYSTEM321, the only MGO is COMPUTER-SYSTEM (and no MSN concept is concrete). Trivially, LCS({COMPUTER-SYSTEM}) = COMPUTER-SYSTEM. Hence, SYSTEM321 is identified as a COMPUTER-SYSTEM, even though its description does not say so explicitly. In particular, COMPUTER-SYSTEM becomes a base concept of SYSTEM321 and in the resulting recognition state for SYSTEM321, COMPUTER-SYSTEM becomes an MSN concept.

The preceding discussion was simplified for clarity; we can sometimes do better. If no MSN concept is a concrete concept bijectively instantiated by I, then by Definition 11 (page 58), I will ultimately instantiate at least one optional *concrete* concept. Thus, constraints on I may be derived from the LCS of the most general optional concepts that are concrete, written LCS(MGO-CONCRETEs). These concepts could be identified by traversing downward from the MGOs. LCS(MGOs) must always subsume LCS(MGO-CONCRETEs). When the subsumption is proper, we obtain tighter constraints on the individual. Assuming, as we do in this thesis, that all concrete concepts are leaves, we can simply take the LCS of the MSOs.

At this point, we have presented our predictive concept recognition methodology for description logic. Since our methodology relies heavily on the closed terminology assumption, we proceed in the next section to discuss the impact of making a CTA.

3.9 Open vs. Closed Terminologies

Under CTA, the set of concepts that an individual can instantiate, perhaps after monotonic update, is limited by the requirement that ultimately it must bijectively instantiate an explicitly defined concrete concept. That requirement also imposes constraints on the description an individual may assume, e.g., the combination of roles it may restrict and the particulars of those role restrictions. Consider the following individual, which instantiates IBM-RISC-CPU (defined in Figure 2.2), even though IBM-RISC-CPU is not stated as a base concept:

(create-individual IBM-RISC-CPU86
  (and CPU
    (fills vendor IBM)
    (the technology RISC)))

Under OTA, IBM-RISC-CPU86 is in the extension of both IBM-CPU and RISC-CPU, whether or not IBM-RISC-CPU is explicitly defined in the terminology. Now suppose for the remainder of this paragraph that the terminology *does not* contain IBM-RISC-CPU. Then, under CTA, IBM-RISC-CPU86 is not allowable, and indeed no individual CPU can be both an IBM-CPU and a RISC-CPU.[26] Thus, CTA restricts

[26]Since restricted-roles(IBM-CPU) is neither a subset nor a superset of restricted-roles(RISC-CPU), no bijective instantiation of one can also instantiate the other. Also, no third concept

the extensions of IBM-CPU and RISC-CPU. In general, a concept's extension under CTA is a subset of its extension under OTA. Furthermore, all CTA-consistency relationships must hold under OTA, but the converse is not true. For example, IBM-PROCESSOR-DEVICE and UNIX-RISC-SYSTEM are OTA-consistent. Without the presence of IBM-RISC-CPU, they would not be CTA-consistent, because their processor roles would not have CTA-consistent value restrictions. The same reasoning applies to individuals under CTA. Without IBM-RISC-CPU, choosing an IBM-CPU for an individual system's processor role (either as a filler or as the value restriction), would be sufficient to render that system inconsistent with RISC-MULTIPROCESSOR-SYSTEM and UNIX-RISC-SYSTEM. On the other hand, choosing a RISC-CPU would be sufficient to rule out IBM-PROCESSOR-DEVICE and DUAL-IBM-PROCESSOR-SYSTEM. None of this could be concluded under OTA.

CTA also differs from OTA regarding primitives. CPU and DISK as defined in Figure 2.2 are not CTA-consistent, because each has a primitive (itself) that the other lacks, and no third concept licenses their consistency indirectly. Under OTA, CPU and DISK are presumed to be consistent by default, and their disjointness would have to be declared explicitly.

Finally, constraint derivation via the LCS inference depends crucially on the CTA. Without CTA, the "optional" concepts might include any number of concepts not explicitly defined in the terminology. Consequently, taking the LCS of optional *known* concepts in order to derive constraints on an individual would be entirely unjustified.

Some ideas on formalizing the semantics of description logic under the CTA are suggested in Chapter 7. Practical performance implications of our methodology are also of interest, so we study that issue next.

would license their consistency indirectly.

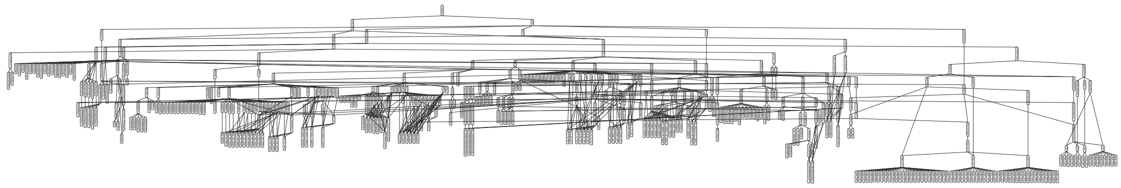


Figure 3.10: Configuration Concept Taxonomy

3.10 Performance Analysis

We now analyze the performance of our predictive recognition algorithm with two goals: to determine whether the algorithm is feasible for interactive use, and to get a sense of how it spends its time. Our experiment was conducted in the context of a small (but realistic) configuration terminology. This terminology was developed by an independent group, and was engineered specifically for demonstration and testing of a configuration engine that uses a K-REP knowledge base as illustrated in Figure 3.1 on page 46. Some impression of the concept taxonomy's layout may be gained from Figure 3.10, even if details are necessarily obscured at this scale. The taxonomy has 501 semantically distinct concepts, including 270 leaves. The number of roles restricted by leaf concepts ranges from 0 to 11, with a mean of 5.04. For the purpose of this experiment, all leaf concepts are presumed to be concrete, and all others abstract. Note that the uppermost concepts in the taxonomy are intended only for the internal use of the configurator, and do not represent meaningful choices for an end-user.[27]

To measure cost, we instrumented our code with the METERING system, a public domain Common Lisp utility described in [Kantrowitz, 1991]. METERING allows us to count basic operations and gather statistics on timing and storage utilization without modifying the existing code. Moreover, it adjusts timing statistics to discount overhead involved in the monitoring itself. We used METERING to count concept visits during the MSN, MSO, and MGO phases of our predictive recognition algorithm, as well as both kinds of description comparison: the instantiation tests during the MSN phase and the direct consistency tests during the

[27] They have names such as ITEM, UNIT, PART, etc.

MSO phase. Finally, we recorded the overall time for predictive recognition as concepts are incrementally instantiated.

Our experiment uses a program to incrementally instantiate each leaf concept in turn, as follows. First, the program collects a set of increasingly general subsumers of the leaf (inclusive) by traversing a path from the leaf towards the root. When a concept has multiple parents, one is chosen at random. The traversal stops upon encountering one of the general concepts intended only for internal configurator use. Then, we incrementally instantiate an individual by repeatedly adding the collected concepts to the individual's description, one at a time in reverse order, until it bijectively instantiates the leaf concept. During each *iteration* of this process, one concept is added to the individual's description and the individual's recognition state is updated. For example, we might incrementally instantiate COMPUTER-SYSTEM as defined in Figure 2.2 on page 20, by adding SYSTEM to the individual's description on the first iteration, and COMPUTER-SYSTEM itself on the second iteration. So as to focus on the predictive recognition algorithm, constraint derivation is disabled during this experiment.

For the configuration taxonomy of Figure 3.10, our implementation of the predictive recognition algorithm spends 86 percent of its time in the MSO phase, 13 percent in the MSN phase, and only 1 percent in the MGO phase. The "bushiness" of the concept taxonomy, together with the inherently bottom-up nature of the MSO search, explains why most time is devoted to finding the MSO concepts.

Our experimental findings are further detailed in Figures 3.11 through 3.22 on pages 88 through 93. Each pair of figures on a page contrasts the "cumulative" algorithm of Section 3.7 with a "non-cumulative" variant in which the MSN, MSO, and MGO sets are re-initialized between each iteration of the incremental instantiation process.[28] Of course, the non-cumulative variant still benefits from the subsumption-based organization of the taxonomy and the synergistic relationship among the MSN, MSO, and MGO phases. Thus, the contrast between the

[28]Remember that initially, the MSN set contains the root concept THING, the MGO set contains THING's children, and the MSO set contains the leaf concepts.

cumulative and non-cumulative variants shows the extra value of maintaining the recognition state between iterations. As we go on to review the results, some important caveats are in order. First, the reader should be cognizant of the different vertical scales in the different figures. Second, results are differentiated along the x-axis by the number of iterations in the incremental instantiation process. Of course, leaves at deeper levels of the taxonomy will tend to be instantiated over a greater number of iterations, which tends to be more time consuming. However, they *also* tend to have more complex descriptions, i.e., more primitives and restricted roles, which makes each iteration more expensive, as well. Consequently, the reader should not conclude that all iterations are equal.

Figures 3.11 and 3.12 show how many concepts are visited during MSN phases. For example, Figure 3.12 demonstrates that on average, the total number of concept visits[29] over seven iterations of incremental instantiation is still only about half the number of concepts in the taxonomy. These results highlight the organizational power of definitional taxonomies. Figures 3.13 and 3.14 show how many instantiation tests are made during this process. Observe that Figures 3.11 and 3.13 have strikingly similar shapes, as do Figures 3.12 and 3.14. Focusing on the latter pair, we see that Figure 3.14 shows noticeably lower counts than Figure 3.12. The difference reflects the positive impact of "backwards marking" in avoiding redundant instantiation tests.

Figures 3.15 and 3.16 show the number of concepts visited during MSO phases. The sharp contrast between the two illustrates the value of the incremental recognition state in limiting the number of concepts which must be examined when going from one iteration to the next. Figures 3.17 and 3.18 both show that the number of direct consistency tests is very well controlled by the pre-processing step which ensures that only those concepts that are subsumees of every MSN concept can possibly be MSO concepts (in an augmented terminology).

Figures 3.19 and 3.20 show how many concepts are visited during the MGO phases; no comparisons with the individual are required. Figure 3.20 in particular

[29]Concepts may be visited more than once, both within and across iterations.

indicates that dramatically few nodes are visited, thanks to the fact that all necessary and impossible concepts are already marked by the MSN and MSO phases of a given iteration.

Figures 3.21 and 3.22 show the total times consumed over all iterations during incremental, predictive concept recognition. These times are clearly adequate for interactive use.

The current implementation of our predictive recognition algorithm still permits ample opportunity for performance tuning. Furthermore, the implementation is for the Common Lisp version of K-REP. An initial port of K-REP from Common Lisp to C++ yielded an order of magnitude speedup in classification, even without any special effort directed towards efficiency.[30] Considering that predictive concept recognition, like classification, primarily involves traversing the concept taxonomy and comparing descriptions, and that the comparisons in each case involve primitives and role restrictions, it seems entirely reasonable to expect similar performance gains when porting our predictive recognition code to C++.

In conclusion, our predictive concept recognition algorithm works quite well by means of its a three-phase approach geared to the terminology's subsumption-based organization. Furthermore, the cumulative version significantly reduces the number of concepts visited in each of the MSN, MSO, and MGO phases, and also significantly reduces the number of instantiation tests. Most importantly, the overall time consumed is easily adequate for interactive applications. We anticipate that with increasingly difficult incremental recognition problems, the value of the three phase, cumulative approach will be further compounded.

3.11 Conclusion

This chapter introduces an explicit distinction between the knowledge engineering and problem solving phases of terminology usage in description logic, for

[30]The improvement is due to the use of arrays instead of linked lists, typed variables, etc.

those applications where the terminology remains fixed during problem solving. It presents a predictive concept recognition methodology for description logic which demonstrates, for the first time, the value of closed terminology reasoning over a description logic knowledge base. These ideas apply to tasks such as computer system configuration where all relevant concepts are known in advance. Our configuration application led us to devise inferences for characterizing the progress of incremental concept instantiation, i.e., bijective and finished instantiation. Those inferences take advantage of a distinction between abstract and concrete concepts, which we introduced to description logic. We take advantage of the closed terminology and its subsumption-based organization in several ways. As an individual system is incrementally specified, we efficiently track the types of systems that may result. We incrementally partition the terminology by categorizing concepts as necessary, optional, or impossible with respect to the current specification. The categories follow respectively from instantiation, potential instantiation, or lack thereof. This information is inherently useful for both user and configuration engine. We also exploit the current partition through novel use of the LCS inference to derive implicit constraints on the system from the knowledge base. This, in turn, may yield a more refined partition. Similar reasoning applies recursively to the system's components. Finally, we take advantage of the derived constraints to inform the user and the configuration engine, and to appropriately restrict future choices. This work is implemented in K-REP, with positive experimental results on performance. Although we focused on configuration, our methodology is domain-independent. Overall, predictive concept recognition constitutes a new and useful category of description logic service.

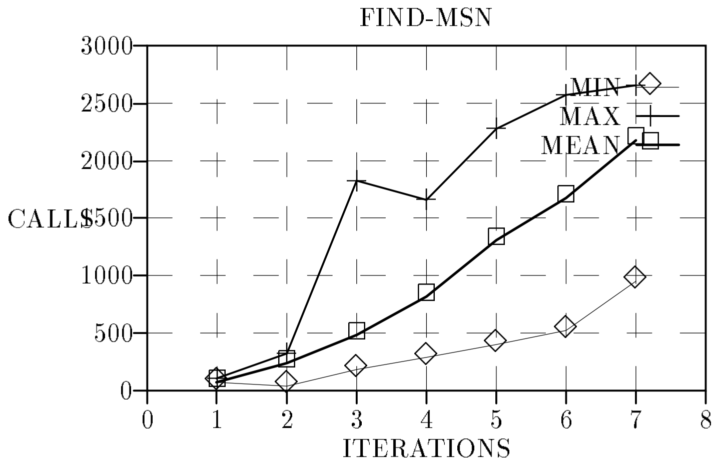


Figure 3.11: Concepts Visited During MSN Search (Non-Cumulative)

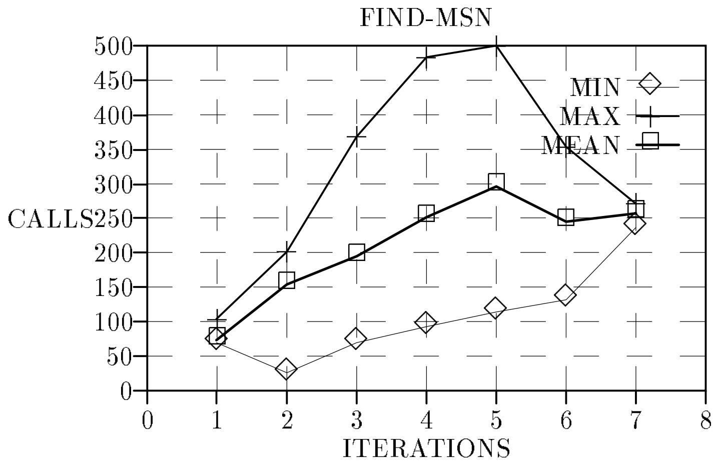


Figure 3.12: Concepts Visited During MSN Search (Cumulative)

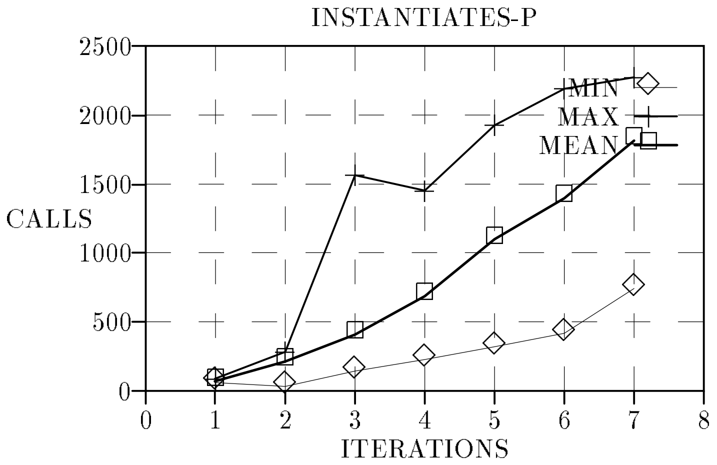


Figure 3.13: Instantiation Tests (Non-Cumulative)

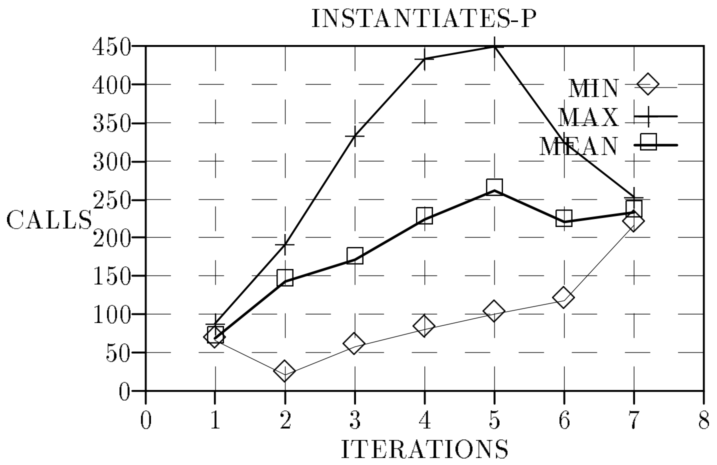


Figure 3.14: Instantiation Tests (Cumulative)

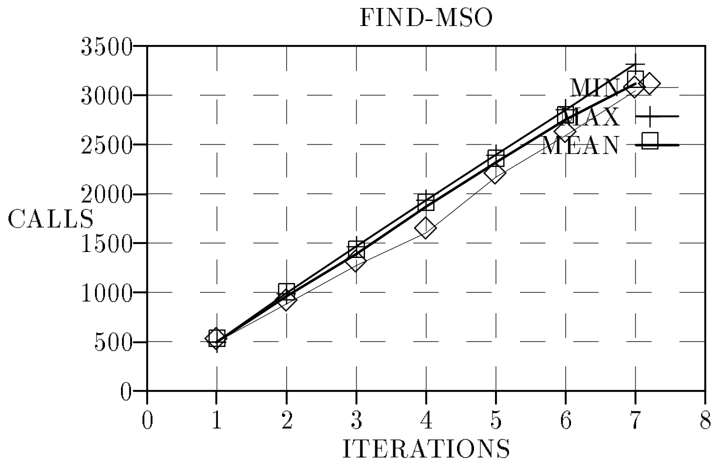


Figure 3.15: Concepts Visited During MSO Search (Non-Cumulative)

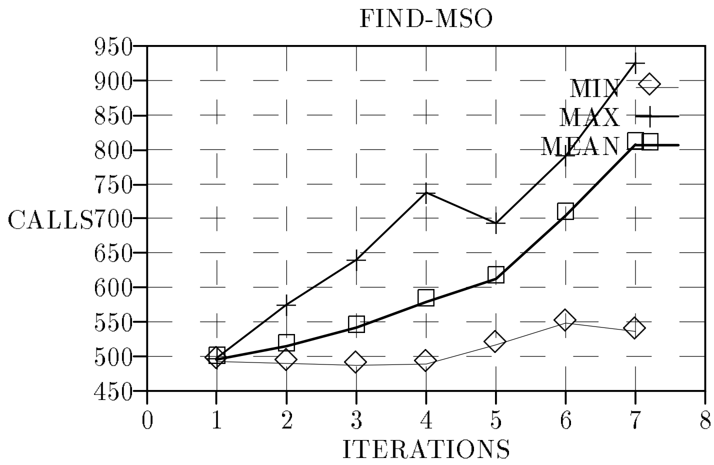


Figure 3.16: Concepts Visited During MSO Search (Cumulative)

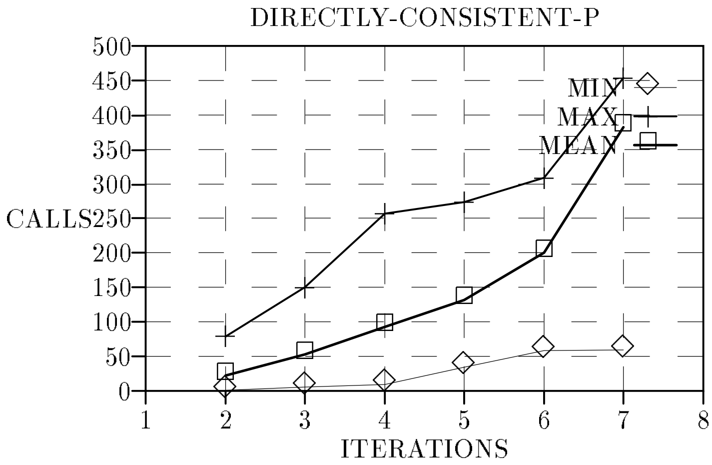


Figure 3.17: Direct Consistency Tests (Non-Cumulative)

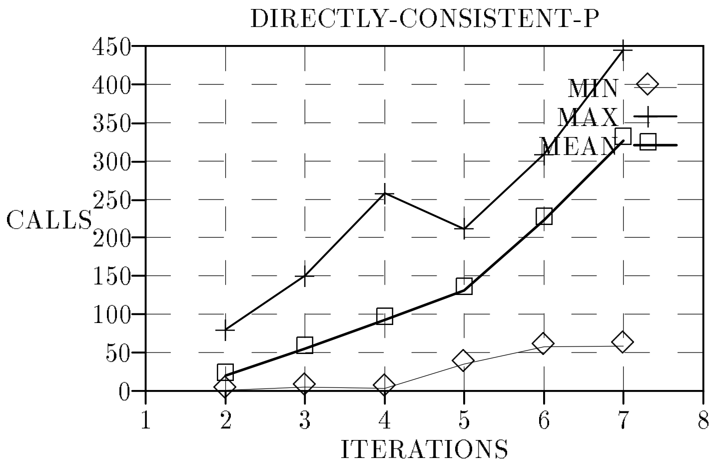


Figure 3.18: Direct Consistency Tests (Cumulative)

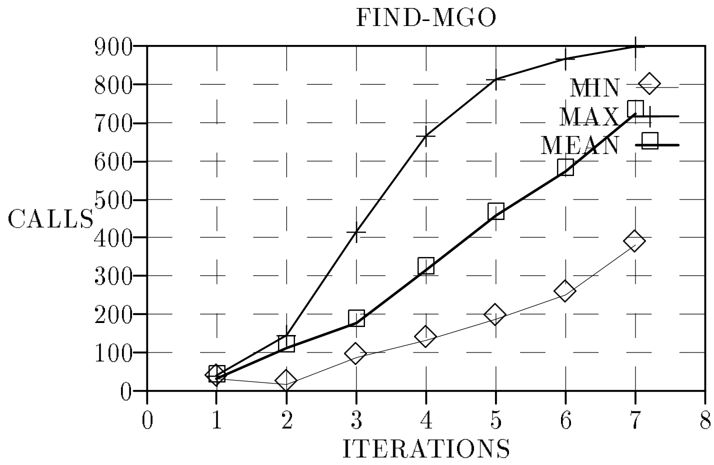


Figure 3.19: Concepts Visited During MGO Search (Non-Cumulative)

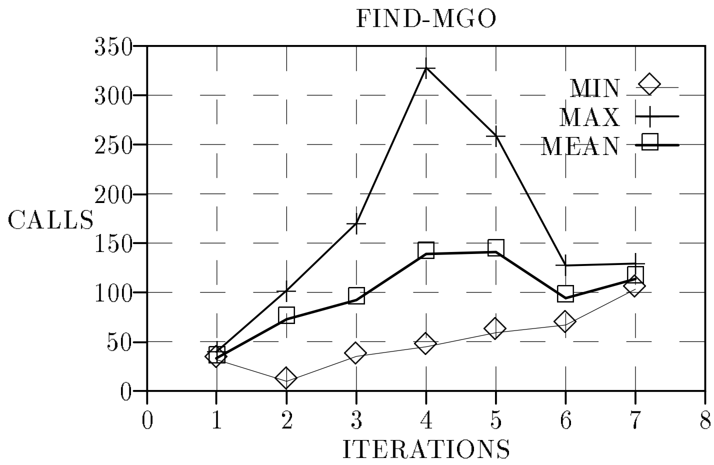


Figure 3.20: Concepts Visited During MGO Search (Cumulative)

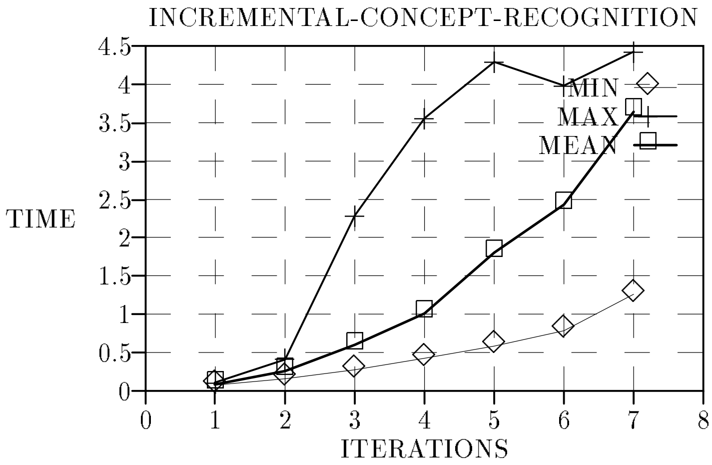


Figure 3.21: Predictive Recognition Time (Non-Cumulative)

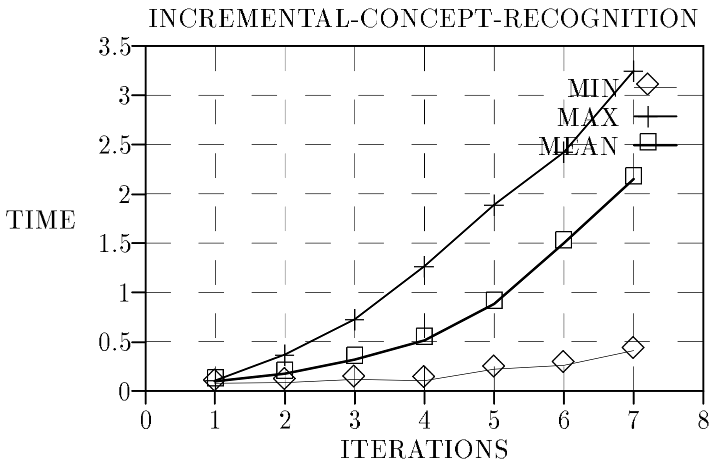


Figure 3.22: Predictive Recognition Time (Cumulative)

Chapter 4

Constraint Networks in Description Logic

4.1 Introduction

In traditional description logic, each description denotes a single concept or individual, and is expressed primarily in terms of primitives and role restrictions denoting its relationship to other concepts and individuals. There is no mechanism for describing a group of related concepts, or a group of related individuals, as peers. Thus, traditional description logic cannot handle some important classes of descriptions. One such class is temporal patterns of events, including plans composed of actions occurring over time. For example, a travel plan might consist of several actions: flying to Switzerland, skiing while there, and then flying home. Each of these actions has its own independent description. The plan forms a higher level description by relating the actions to one another temporally. Another such class is spatial patterns of objects, such as floor plans composed of rooms, furniture, and so on. Clearly, the scope and utility of description logic would be enhanced by an ability to handle composite descriptions like these. A definitional taxonomy of composite descriptions would enjoy all the benefits of standard description logic taxonomies cited in Section 2.1.1. Therefore, the work reported in this chapter builds upon description logic to encompass collections of peer concepts or individuals related by constraints. It is natural to view these descriptions as constraint networks, where the nodes are concepts or individuals and the arcs are binary constraints between them. This chapter also forms the basis for our work on predictive recognition of constraint network concepts, particularly for plan recognition, which follows in Chapter 5.

This chapter begins with a brief overview of the T-REX system and its architecture in the next section. T-REX demonstrates the ideas developed in this chapter and in Chapter 5. Next, Section 4.3 covers the specification, normalization, and completion of constraint network descriptions. Section 4.4 examines constraint network subsumption from both practical and theoretical standpoints. Our focus is mostly on plans conceived as temporal patterns of events, especially plans of action. Section 4.5 analyzes the performance of our subsumption algorithm. Classification of constraint network concepts is treated in Section 4.6, followed by recognition of constraint network individuals in Section 4.7. In Section 4.8 we present our conclusions.

4.2 T-REX System Overview

As a testbed for some of the ideas developed in this chapter and the next one, we have implemented a prototype plan recognition system named T-REX.[1] T-REX's modular architecture integrates and builds upon an existing description logic system for reasoning about concepts and individuals, as well as an existing temporal system. Specifically, it calls upon either K-REP [Mays *et al.*, 1991a] or CLASSIC [Borgida *et al.*, 1989] to manage a knowledge base of actions and related concepts and individuals, along with MATS [Kautz and Ladkin, 1991] to propagate qualitative and metric temporal constraints. T-REX contains its own

[1] The name derives as follows: Terminological RECognition System → T-RECS → T-REX (recall that description logics were formerly known as terminological logics).

equality constraint management facility. Thus, T-REX supports a rich and well-defined plan description language currently focused on plan bodies.[2] For plans, T-REX supports all the description logic inferences outlined in Section 2.1.3.

A T-REX system diagram appears in Figure 4.1. When a plan is defined, T-REX checks its syntax, normalizes the definition, completes it by deriving implicit information, and finally classifies it in a plan library by means of subsumption tests against previously defined plans. When observations are presented, T-REX determines the status of every plan with respect to the observations, by using a variant of the concept recognition methodology presented in Chapter 3. Hence the concept taxonomy is partitioned into necessary, optional, and impossible regions. Whenever the observations are updated, T-REX incrementally refines the partition accordingly. T-REX actively responds to recognition results through demons (procedures) conditioned to execute whenever a certain plan undergoes specified status changes, e.g., from optional to necessary.

The remainder of this chapter and the following chapter provide a detailed presentation of the ideas embodied in T-REX.

4.3 Constraint Network Descriptions

This section presents constraint network descriptions in detail. In general, a *constraint network concept* intensionally describes a class of *constraint network individuals*, by analogy to the concepts and individuals of standard description logic. A constraint network concept represents a general pattern in which collections of individuals may be arranged, while a constraint network individual represents a pattern in which a specific collection of individuals are arranged. A node of a constraint network concept (individual) is characterized by a standard description

[2]There are advantages to incorporating the plan language directly within the description logic as in [Artale and Franconi, 1993; Artale and Franconi, 1994]. K-REP could be extended to do this, largely by incorporating existing T-REX code in a modular way. We will return to this issue in Section 6.3.5.

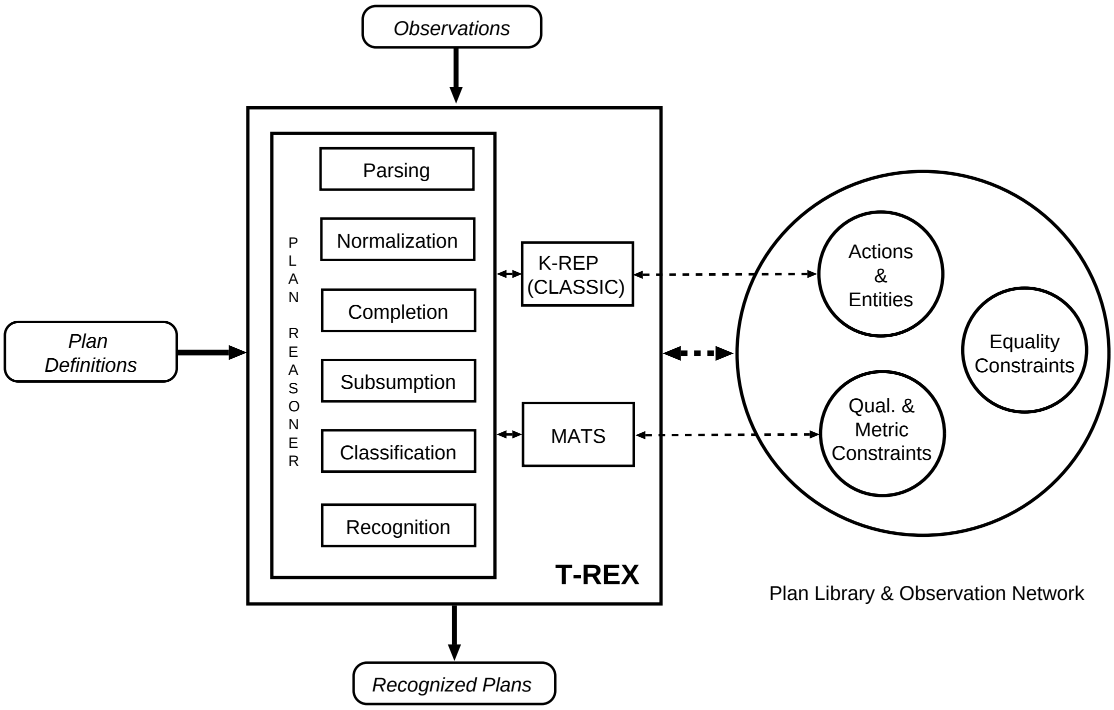


Figure 4.1: The T-REX System

logic concept (individual). An arc expresses a relationship between a pair of concepts (individuals) that is contingent on their joint appearance in the collection. In contrast, intrinsic properties of concepts and individuals are represented by their roles.

To ground our discussion, we will focus on network descriptions having temporal and equality constraints. In particular, we introduce *QME constraint networks* to represent patterns of events. The events are described by concepts drawn from a description logic taxonomy. They are related by qualitative temporal constraints, metric temporal constraints, and equality constraints.[3] For brevity, we will refer to them simply as *QME networks*.

[3]Thus, QME stands for Qualitative, Metric and Equality.

4.3.1 QME Constraint Network Concepts

QME network concepts denote possible patterns of events over time. We associate both a concept and a generic temporal interval with each node. The concepts denote classes of events and the intervals denote time periods over which they occur. In general, a concept may be associated with more than one node, so nodes are labeled with unique identifiers. These labels are arbitrary, though in practice they are presumably chosen for readability. We also associate a qualitative temporal constraint, metric temporal constraint, or equality constraint with each arc of a QME network. Hence, there are qualitative, metric, and equality arcs. Recall that qualitative and metric constraints were formally defined in Chapter 2. We now introduce binary equality constraints:

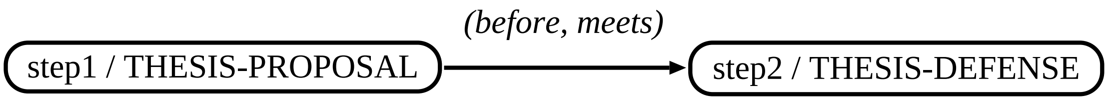
Definition 31 *A binary* equality constraint *between role* R *of concept* X *and role* S *of concept* Y*, written* $\mathrm{R_X} = \mathrm{S_Y}$*, requires the sets of fillers of those roles to be identical.*

We will say that R_X and S_Y are the *operands* of the constraint. We do not allow role composition in equality constraint operands, thus avoiding a potential source of undecidability when computing subsumption [Schmidt-Schauß, 1989].

Now we can formally specify the different types of arcs. In the case of QME networks, all arcs are directed. First, a *qualitative arc* denotes a relationship between the time intervals associated with a pair of nodes:

Definition 32 *A directed* qualitative arc *from node* a *to node* b *labeled* $(r_1, \ldots, r_n)$ *represents the qualitative constraint* a $(r_1 \vee \ldots \vee r_n)$ b.

When there is no ambiguity, we will use the labels of nodes to reference their temporal intervals, e.g., a and b in the preceding definition. To capture a pattern in which a thesis proposal precedes a thesis defense, we can create two nodes arbitrarily labeled *step1* and *step2*, associate suitably defined concepts THESIS-PROPOSAL and THESIS-DEFENSE with them, and connect the nodes with an appropriate qualitative arc to form a network as follows:



For clarity, we show both the label and the associated concept inside nodes of a network diagram. When it is possible to convey a network in linear text, we will sometimes do so, e.g., the preceding diagram is written:

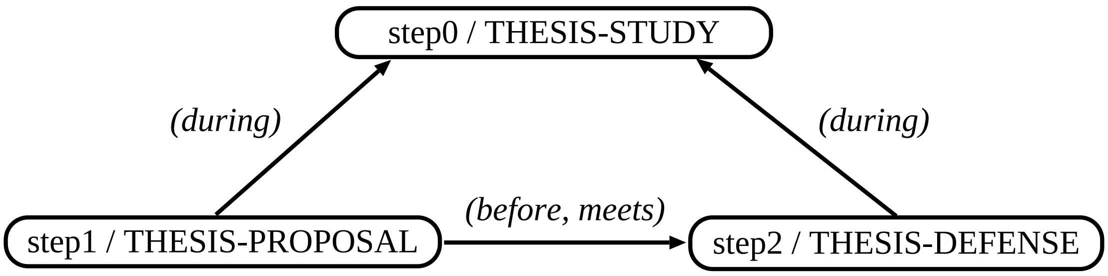
step1/THESIS-PROPOSAL *(before, meets)* step2/THESIS-DEFENSE

Not stated, but clearly implied, is the inverse relationship that *step2* is either after or is met by *step1*:

step2/THESIS-DEFENSE *(after, met-by)* step1/THESIS-PROPOSAL

Identification of implicit relationships through constraint propagation will be discussed in Section 4.3.4.

We could easily expand our pattern description with other events and qualitative constraints, e.g., both THESIS-PROPOSAL and THESIS-DEFENSE occur during THESIS-STUDY:

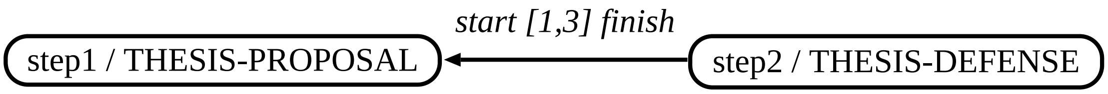


Second, a metric arc denotes a relationship between endpoints of the time intervals associated with a pair of nodes (recall Definition 5):

Definition 33 *A directed* metric arc *from node* a *to node* b *is labeled using standard interval notation, where* $F, G \in \{start, finish\}$*:*

1. *F (m, n) G represents the metric constraint* $m < a_F - b_G < n$.
2. *F [m, n) G represents the metric constraint* $m \leq a_F - b_G < n$.
3. *F (m, n] G represents the metric constraint* $m < a_F - b_G \leq n$.
4. *F [m, n] G represents the metric constraint* $m \leq a_F - b_G \leq n$.

For example, we might wish to promulgate a policy that a thesis defense follows a proposal by at least one year and at most three years. The nodes of the resulting network are the same as before. When the metric quanta are years, the following network results. Observe that this time, the relationship as stated applies from right to left:



To be precise, this metric arc indicates a time period of from one to three years (inclusive) from the start of the time interval during which a thesis is defended to the end of the time interval during which a thesis is proposed. The sign of the numbers that bound the magnitude of the difference, $step2_{start} - step1_{finish}$, follows from the intention that $step2_{start}$ occurs after $step1_{finish}$. This network entails a thesis proposal strictly before a thesis defense, which implies our first network with a qualitative arc[4], but the converse is not true, so this one carries strictly more temporal information. Again, where possible, we may write networks with metric constraints in linear fashion, e.g., the preceding diagram is written:

step2/THESIS-DEFENSE *start [1,3] finish* step1/THESIS-PROPOSAL

[4]Namely, step1/THESIS-PROPOSAL *(before, meets)* step2/THESIS-DEFENSE, which additionally allows the proposal immediately before the defense.

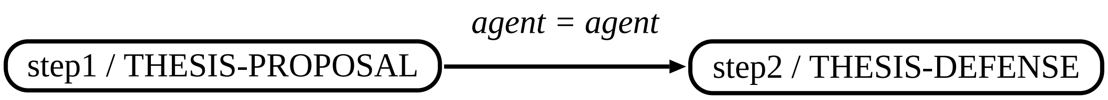
Again, there is an equivalent inverse relationship:

step1/THESIS-PROPOSAL *finish [-3,-1] start* step2/THESIS-DEFENSE

Finally, an equality arc denotes an identity relationship between the sets of fillers on roles of the concepts (individuals) associated with a pair of nodes. Then we have:

Definition 34 *A directed* equality arc *from node* a *to node* b *labeled* R = S *represents the binary equality constraint* $\mathrm{R_a} = \mathrm{S_b}$.

When there is no ambiguity, we will also use the labels of nodes to reference their associated concepts (individuals), e.g., *a* and *b* in the preceding definition. Returning to our example, the constraint that a thesis proposal occurs before a defense intuitively concerns the thesis of a single student. This requirement can be enforced with a suitable equality arc between the relevant nodes:



To be precise, this equality arc indicates that the agent of the THESIS-PROPOSAL action known as *step1* is identical to the agent of the THESIS-DEFENSE action known as *step2*. As usual, we may write this relationship inline:

step1/THESIS-PROPOSAL *agent = agent* step2/THESIS-DEFENSE

Every directed equality arc has an equivalent inverse, e.g., the inverse of this equality arc is:

step2/THESIS-DEFENSE *agent = agent* step1/THESIS-PROPOSAL

Although the label on these equality arcs is symmetric, not all of them are. For example, we may wish to indicate that the advisor of the thesis study is also the chair of the thesis defense:

$$\text{step0/THESIS-STUDY} \quad \mathit{advisor} = \mathit{chair} \quad \text{step2/THESIS-DEFENSE}$$

Here, the inverse arc has a different label:

$$\text{step2/THESIS-DEFENSE} \quad \mathit{chair} = \mathit{advisor} \quad \text{step0/THESIS-STUDY}$$

As a notational convenience, we permit more than two operands when writing an equality constraint:

Definition 35 *An* equality constraint *among any number of roles written* $\mathrm{R_X} = \cdots = \mathrm{S_Y}$*, requires the sets of fillers of those roles to be identical.*

For example, we might extend the preceding equality constraint concerning a thesis proposal and defense as follows:

$$agent_{\mathrm{THESIS-PROPOSAL}} = agent_{\mathrm{THESIS-DEFENSE}} = agent_{\mathrm{THESIS-STUDY}}$$

Since equality is transitive and symmetric, an N-ary equality constraint can be decomposed into a logically equivalent set of binary equality constraints. We thus derive binary equality arcs, as will be shown in Section 4.3.4.

The various constraints discussed in this section, omitting the redundant inverses, can be combined into a single constraint network as in Figure 4.2.

Only a limited set of explicit constraints are shown in this network. Other constraints are implicit and may be derived through constraint propagation as will be indicated in Section 4.3.4.

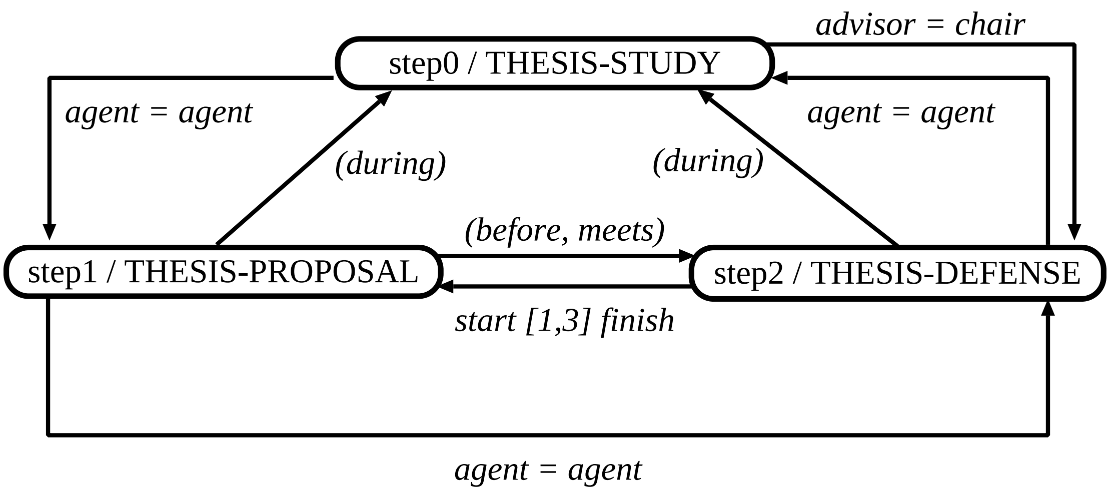


Figure 4.2: Network with Combined Constraints

In the remainder of this chapter and the next chapter, we will concentrate on QME network concepts as *plan concepts*, which are presumed to represent purposeful patterns of actions. We will also consider *plan individuals*, which are used to represent a singular pattern of individual actions. Since we speak about these entities at several levels, the correspondences among them are elucidated below:

	Constraint Network	**QME Network**	**Plan Concept**	**Plan Individual**
Nodes	unary constraints	concepts	action concepts	action individuals
Arcs	binary constraints	qualitative, metric, and equality constraints		

4.3.2 Plan Concepts

Let us now concentrate on using QME networks in the representation of plan concepts and plan individuals. Plans can be thought of as goal-directed behavior. Plan descriptions typically include preconditions, effects, a body composed of steps to carry out the plan, and some additional constraints. We define a *plan body* as a collection of steps, along with some constraints among them. Each step has an arbitrary label and a type of *action* associated with it. Action types, which can be thought of as atomic plans, are represented by standard description logic concepts. We call them *action concepts*. Together, the action concepts form an *action taxonomy* which is part of a standard description logic taxonomy. Action concepts reference related concepts through roles such as *agent* and *object*. We assume that the taxonomy includes every type of action that appears in a plan, following the CTA of Chapter 3. A *plan network* is a constraint network concept whose nodes correspond to steps of a plan's body. Hence, an action concept is associated with each node. These can be thought of as unary action constraints. As a simplifying assumption, we restrict plan networks to QME networks whose nodes are described by action concepts. A plan network denotes a set of possible plan individuals that satisfy its constraints.

Several examples of travel plan concepts were given in Figure 1.1. Here, we take a more comprehensive look at our plan notation. As with PLAN-A of Figure 1.1, we write plan descriptions using a Lisp-like syntax.[5] A plan is defined with the *define-plan* operator, whose first argument is the plan's name. Constraints, grouped by type, are then introduced by Common Lisp-style keywords[6]:

- **Steps:** A step consists of a label followed by an action concept specifier, both enclosed in parentheses, e.g., (step1 VISIT-CITY). The list of steps is itself enclosed in parentheses.

- **Qualitative constraints:** Each qualitative constraint is written as in Definition 4 and enclosed in parentheses. For example, (step1 (*before* $\vee$ *meets*) step2) conveys a temporal precedence relationship from *step1* to *step2*. The list of qualitative constraints is itself enclosed in parentheses.

- **Metric constraints:** Similarly, each metric constraint is written as in Definition 5 and enclosed in parentheses. For example, ($0 <$ step2$_{finish}$ − step1$_{start}$

[5] The notation corresponds directly to T-REX input, but is not identical.
[6] In Common Lisp, a keyword is a symbol whose name begins with a colon.

$\leq$ 2880) conveys a relationship between the finish of *step2* and the start of *step1*. The list of metric constraints is enclosed in parentheses as well.

- **Co-reference constraints:** Every equality constraint consists of two or more role designators of the form $<$ *role-name* $>$ ($<$ *label* $>$) where $<$ *label* $>$ identifies a step in the plan and $<$ *role-name* $>$ identifies a role of the concept associated with that step. Role designators are separated by "=" for readability. An example is (agent(step1) = agent(step2)). As usual, the list of equality constraints is enclosed in parentheses. While equality constraints are the only kind of co-reference constraints currently handled by T-REX, we use the *co-reference-constraints* keyword to allow for future enhancements such as inequality constraints. In Section 4.3.5, we will extend the equality constraint notation for macro *subplans*.

Each class of constraints is optional, although steps are required as referents if any other kind of constraint is defined. Note that a plan without steps can be considered well-formed.[7]

Several examples of travel plan concepts from Figure 1.1 are diagrammed as constraint networks in Figure 4.3 on page 106. Notice that there is a one-to-one correspondence between steps in the plan descriptions and nodes in the constraint networks.

We do not claim our representation itself as a substantial research result, but we do point out that it offers a unique combination of features. Only a few plan-based systems take advantage of the formal semantics and taxonomic inferences of description logic systems, as surveyed in Section 6.3. T-REX is the only plan recognition system to do so. Our interest in handling a rich variety of temporal information led us to integrate the full temporal expressiveness of MATS [Kautz and Ladkin, 1991] into our plan language. To our knowledge, no other plan reasoning system can handle such expressive temporal constraints.

[7] Indeed, under some circumstances, doing nothing may be the best strategy.

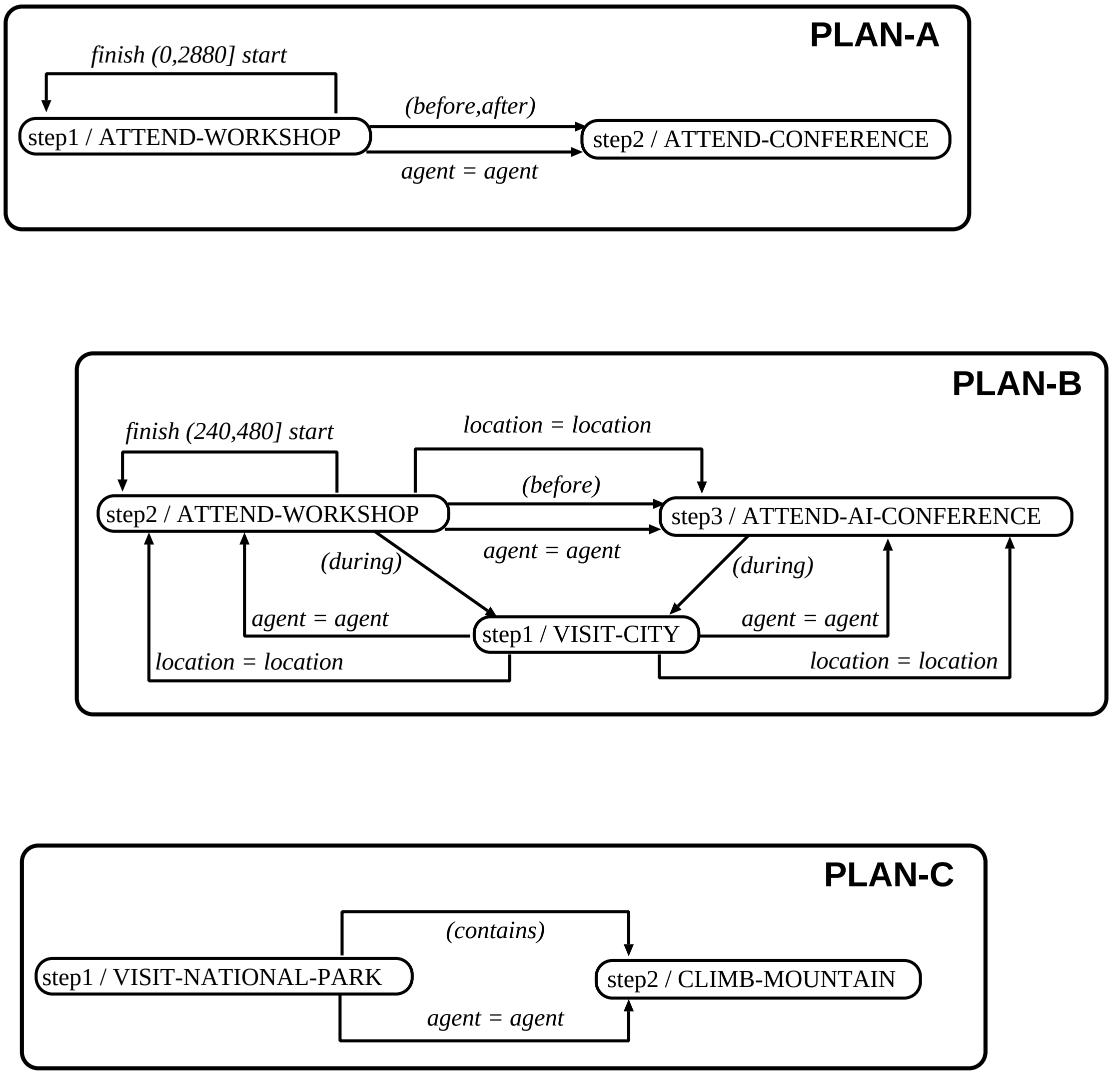


Figure 4.3: Sample Plan Networks

We now turn to the plan individuals which instantiate plan concepts.

4.3.3 Plan Individuals

For our purposes, a *plan individual* is essentially like a plan concept, except that its body is composed from individual actions rather than action concepts. Figure 4.4 defines OBS100, an instance of PLAN-A from Figure 1.1 on page 9, along with its individual constituents. OBS100 asserts the occurrence of the ATTEND-WORKSHOP98 action, followed after a non-zero time interval by the occurrence of the ATTEND-AI-CONFERENCE99 action, which takes up six hours (360 minutes). Although OBS100 does not specifically contain qualitative and co-reference constraints corresponding to those of PLAN-A, T-REX can nevertheless recognize that OBS100 satisfies all of PLAN-A's constraints. The first metric constraint on OBS100 implies that *step1* is qualitatively before *step2*[8], and the fact that A-RANDOM-RESEARCHER is the sole agent of both steps is sufficient to fulfill the equality constraint. Recognition will be defined precisely in Section 4.7.

In general, complete information about a plan individual may not be available, so constraints on the individual may be missing and/or may convey uncertainty. This leads directly to the topic of network completion.

4.3.4 Network Completion

When descriptions are presented to T-REX, they may be logically incomplete or perhaps even incoherent. In general, users will not find it practical and convenient to provide T-REX with logically complete input. Instead, T-REX accepts incomplete descriptions and employs constraint propagation methods to infer implicit constraints, make them explicit, and detect any inconsistencies.

[8] T-REX also supports absolute metric constraints on the extrema of intervals through its use of MATS. To make it easier for the reader to compare plans and their instances, we will generally express plan individuals in terms of relative metric constraints.

```
(create-individual A-RANDOM-RESEARCHER)

(create-individual MINOR-FIASCO)

(create-individual MAJOR-BOONDOGGLE)

(create-individual METROPOLIS)

(create-individual ATTEND-WORKSHOP98
    (and ATTEND-WORKSHOP
        (exactly 1 agent)
        (fills agent A-RANDOM-RESEARCHER)
        (exactly 1 workshop)
        (fills workshop MINOR-FIASCO)
        (exactly 1 location)
        (fills location METROPOLIS)))

(create-individual ATTEND-AI-CONFERENCE99
    (and ATTEND-AI-CONFERENCE
        (exactly 1 agent)
        (fills agent A-RANDOM-RESEARCHER)
        (exactly 1 conference)
        (fills conference MAJOR-BOONDOGGLE)
        (exactly 1 location)
        (fills location METROPOLIS)))
```

In OBS100, A-RANDOM-RESEARCHER *attends the* MINOR-FIASCO *workshop for six hours, followed by the* MAJOR-BOONDOGGLE *AI conference.*

(create-individual-plan OBS100
    :steps ((step1 ATTEND-WORKSHOP98)
            (step2 ATTEND-AI-CONFERENCE99))
    :metric-constraints (($0 < \text{step2}_{start} - \text{step1}_{finish} \leq \infty$)
                         ($360 \leq \text{step1}_{finish} - \text{step1}_{start} \leq 360$)))

Figure 4.4: Sample Plan Individual

T-REX uses K-REP/CLASSIC to infer implicit constraints on descriptions of concepts and individuals, e.g., via inheritance. It relies on MATS to derive implicit temporal information based on transitivity. In both cases, T-REX lives with the possibility of logically incomplete results. For details, refer to Sections 2.1.4 and 2.3.1, respectively.

It is assumed that temporal constraints are entirely independent of action descriptions. Of course, this is a simplification because world knowledge about actions can very well provide temporal information. For example, we know that the proposal of a particular thesis must, by its very nature, precede its defense. A simple mechanism to support this kind of domain-specific inference in T-REX will be given in Section 4.3.6.

Finally, T-REX must do completion with respect to equality constraints across concepts by itself. Here, completion must address the transitivity and symmetry of equality. Consider a plan with two equality constraints:

```
(define-plan PLAN-E
   :steps ((step1 ACTION-A)
           (step2 ACTION-B)
           (step3 ACTION-C))
   :co-reference-constraints ((agent(step1) = agent(step2))
                              (agent(step2) = agent(step3))))
```

Equality constraints with common operands must be correlated for completeness. It can be seen that PLAN-E entails another, unstated equality constraint:

agent(step1) = agent(step3)

Conceptually, each equality constraint gives rise to a pair of directed arcs which are inverses of each other. Thus PLAN-E would generate a total of six equality arcs labeled *agent* = *agent*, one from each step to both of the others. Of course, half of these equality constraints are redundant and could be dispensed with.

```
(define-primitive-concept HUMAN)

(define-primitive-concept FEMALE)

(create-individual JANE-DOE
    (and human female))

(create-individual JANET-ROE
    (and human female))

(define-primitive-concept ACTION-A
    (and (all agent HUMAN)
         (at-least 2 agent)
         (at-most 4 agent)
         (fills agent JANE-DOE)))

(define-primitive-concept ACTION-B
    (and (all agent FEMALE)
         (at-least 1 agent)
         (at-most 3 agent)
         (fills agent JANET-ROE)))

(define-plan PLAN-S
    :steps ((step1 ACTION-A)
            (step2 ACTION-B))
    :co-reference-constraints ((agent(step1) = agent(step2))))
```

Figure 4.5: Equality Constraint for Completion

T-REX must also propagate role restrictions across co-reference constraints. Let us focus on this process in terms of the self-contained example of Figure 4.5. The equality constraint on PLAN-S requires the agent roles of *step1* and *step2* to have identical fillers. T-REX therefore conjoins the restrictions on the agent roles of ACTION-A and ACTION-B, in the process creating two new concepts that correspond to them. We will refer to them here as ACTION-A′ and ACTION-B′, respectively:

```
(define-concept ACTION-A′
   (and ACTION-A
      (all agent HUMAN&FEMALE)
      (at-least 2 agent)
      (at-most 3 agent)
      (fills agent JANE-DOE)
      (fills agent JANET-ROE)))

(define-concept ACTION-B′
   (and ACTION-B
      (all agent HUMAN&FEMALE)
      (at-least 2 agent)
      (at-most 3 agent)
      (fills agent JANE-DOE)
      (fills agent JANET-ROE)))
```

The value restriction HUMAN&FEMALE was created by T-REX to represent the conjunction of humans and females, i.e., women. Both ACTION-A′ and ACTION-B′ have two known agents, JANE-DOE and JANET-ROE; they may or may not have another agent. Notice that in refining the agent roles, T-REX has:

- Conjoined the value restrictions
- Maximized the at-least restrictions
- Minimized the at-most restrictions
- Taken the union of the fillers

As part of the completion process, T-REX will substitute ACTION-A′ and ACTION-B′ for their counterparts in PLAN-S. The original concepts, ACTION-A and ACTION-B, are left undisturbed. In general, T-REX propagates role restrictions across any number of equality constraints having a common operand. T-REX will notify the user if the operands of a co-reference constraint are not mutually consistent.

Other issues for network completion are raised by the presence of macro nodes within the network. We address macros next.

4.3.5 Macros

It is often natural to think of complex plans as being composed from simpler plans. To support this kind of abstraction, and as a notational convenience, T-REX has a macro facility. Plans may be embedded as *macro actions* within other plans (but not recursively within themselves). This is convenient for specializing abstract plans in different ways. For example, Figure 4.6 shows how a TOUR-DU-MONT-BLANC plan is specialized into variants involving hiking and cross-country (XC) skiing.

The treatment of macros relies on the completion inferences from the preceding section. Figure 4.7 defines PLAN-D with two steps which are both *subplans*, namely PLAN-B and PLAN-C, repeated from Figure 1.1. When the steps of PLAN-D are expanded, the result is five base-level steps, three from PLAN-B and two from PLAN-C. The qualitative constraint in PLAN-D applies to implicit time intervals representing the macro steps in *toto*, i.e., *step-b* and *step-c*. It implies that each substep of *step-b* precedes every substep of *step-c*. The T-REX language also permits a plan description to specify qualitative, metric, and co-reference constraints on substeps within a macro step.[9] Substeps are referenced by an appropriate nested sequence of labels. For example, the equality constraint on PLAN-D relates the *agent* role of *step1* of *step-b* to the *agent* role of *step1* of *step-c*. Notice that the operands of this equality constraint also participate in the equality constraints defined in PLAN-B and PLAN-C. By transitivity, the agent roles of all five base-level steps are thus equated. T-REX will propagate restrictions across those roles as discussed in Section 4.3.4.

[9]Substeps can be referenced at any level of macro nesting.

TOUR-DU-MONT-BLANC involves round-trip travel through France, Italy, and Switzerland, stopping in towns along the way.

```
(define-plan TOUR-DU-MONT-BLANC
    :steps ((s1 VISIT-FRANCE)
            (s2 VISIT-ITALY)
            (s3 VISIT-SWITZERLAND)
            (s4 VISIT-FRANCE)
            (c1 VISIT-CHAMONIX)
            (c2 VISIT-COURMAYEUR)
            (c4 VISIT-CHAMONIX)
            (travel TRAVEL-ACT))
    :qualitative-constraints
        ((s1 (meets) s2) (s2 (meets) s3) (s3 (meets) s4)
         (c1 (during) s1) (c2 (during) s2) (c4 (during) s4)
         (c1 (overlaps) travel) (travel (overlaps) c4)))
```

In the next two examples, the travel action in the TOUR-DU-MONT-BLANC macro is refined to HIKE and XC-SKI, respectively. Notice that two plans have a single (macro) step. The labels associated with HIKE and XC-SKI identify the travel *substep within the macro step and thus do not introduce an additional step.*

```
(define-plan TOUR-DU-MONT-BLANC-HIKING
    :steps ((step TOUR-DU-MONT-BLANC)
            ((travel step) HIKE)))

(define-plan TOUR-DU-MONT-BLANC-SKIING
    :steps ((step TOUR-DU-MONT-BLANC)
            ((travel step) XC-SKI)))
```

Figure 4.6: Specializing a Macro Plan

In PLAN-B, an agent visits a city, during which time s/he attends a workshop for more than 240 and at most 480 minutes, then attends an AI conference, all in the same location.

(define-plan **PLAN-B**
:steps ((step1 VISIT-CITY)
(step2 ATTEND-WORKSHOP)
(step3 ATTEND-AI-CONFERENCE))
:qualitative-constraints ((step2 (*during*) step1)
(step3 (*during*) step1)
(step2 (*before*) step3))
:metric-constraints (($240 < \text{step2}_{finish} - \text{step2}_{start} \leq 480$))
:co-reference-constraints ((agent(step1) = agent(step2) = agent(step3)))
(location(step1) = location(step2) = location(step3)))))

In PLAN-C, an agent climbs a mountain while in a national park.

(define-plan **PLAN-C**
:steps ((step1 VISIT-NATIONAL-PARK)
(step2 CLIMB-MOUNTAIN))
:qualitative-constraints ((step1 (*contains*) step2))
:co-reference-constraints ((agent(step1) = agent(step2))))

In PLAN-D, an agent carries out PLAN-B followed by PLAN-C.

(define-plan **PLAN-D**
:steps ((step-b PLAN-B)
(step-c PLAN-C))
:qualitative-constraints ((step-b (*before*) step-c))
:co-reference-constraints ((agent(step1(step-b)) = agent(step1(step-c)))))

Figure 4.7: Plan with Macro Steps

Any temporal constraint on a step with a macro action can be propagated to each substep within that macro by appropriate use of constraint propagation algorithms such as those in [Kautz and Ladkin, 1991]. Song and Cohen have shown how to do this for qualitative constraints [Song, 1991; Song and Cohen, 1991]. Their algorithm is sound, but not complete. The incompleteness stems from their decision to avoid case reasoning with disjunctive temporal networks. T-REX uses Song and Cohen's algorithm as part of its plan completion process. An analogous algorithm could be devised to propagate metric constraints on a macro action to its substeps.[10]

So far, we have studied domain-independent aspects of network completion. We next present a means for T-REX to accommodate domain-specific reasoning.

4.3.6 Domain Theory

T-REX itself is a domain-independent plan reasoner. In the process of completing plan descriptions, T-REX checks their consistency by verifying that the stated temporal constraints are satisfiable (within the competence of the MATS propagation algorithm) and that the co-reference constraints are satisfiable. It cannot draw further inferences or further verify the plausibility of plans without some knowledge of the domain. Therefore, we have implemented an inference rule facility which allows users to represent domain-specific inferences over plan descriptions. One use of the rule facility is to enforce integrity constraints. The rule system builds on T-REX's constraint network matching capability.

Domain inference rules in T-REX have an antecedent and a consequent. The antecedent has the same form as a plan description, except that it is unnamed. A rule matches a plan just in case its antecedent subsumes the plan (constraint network subsumption will be detailed in Section 4.4). When a rule matches a plan, its consequent potentially refines the plan. Rule consequents express qualitative temporal, metric temporal, and/or co-reference constraints. These constraints are

[10]However, T-REX does not yet implement such an algorithm.

all added to the matching plan's description. Since rules can only add information, they update plans monotonically, i.e., whenever a rule is applied to a plan, the prior version subsumes the subsequent version. If the rule adds information, constraint propagation over the plan ensues, similar to when the plan was first defined.

Suppose that in our travel domain, we wish to enforce the constraint that an agent can only engage in one travel act at a time. The concept describing a generic travel act might be defined as follows:

```
(define-primitive-concept TRAVEL
    (and ACTION
        (all agent HUMAN)
        (the origin LOCATION)
        (the destination LOCATION)))
```

The following rule states the constraint we have in mind:

```
(define-rule
    :antecedent ((:steps ((s1 TRAVEL)
                          (s2 TRAVEL))
                 :co-reference-constraints ((agent(s1) = agent(s2)))))
    :qualitative-consequents ((s1 (before ∨ meets ∨ met-by ∨ after) s2)))
```

The antecedent of this rule matches all pairs of TRAVEL actions (using T-REX's existing constraint network subsumption code) while the consequent asserts temporal disjointness between any matched pair. In this manner, domain rules may further refine plan definitions. If an inconsistency results, the plan is ill-formed with respect to the domain theory.

Whenever a new plan is defined, the rules are repeatedly applied to monotonically refine constraints in the plan. This process continues until either no more rules are applicable or an inconsistency arises. Naturally, domain-specific rules that apply to plans must apply to their instances too. Therefore, T-REX also applies the set of domain-specific rules to plan individuals when they are first defined and whenever they are updated. New instantiation relationships may thus be discovered. Some standard description logic systems, such as CLASSIC [Borgida *et al.*, 1989], LOOM [MacGregor, 1991b], and MESON [Owsnicki-Klewe, 1988], apply forward-chaining rules to standard individuals in this fashion.

4.3.7 Conclusion

Section 4.3 has specified the representation of constraint network concepts and individuals, with particular emphasis on plan descriptions. We have also covered reasoning with respect to a single constraint network description, including both domain-independent constraint propagation and a rule-based facility to accommodate domain-specific constraints. This section has laid the groundwork for reasoning about the relationship between constraint network descriptions. Of central importance is constraint network subsumption, which we turn to next.

4.4 Constraint Network Subsumption

We now extend the notion of concept subsumption from standard description logic to constraint networks whose nodes are represented by concepts. Throughout this discussion, we will assume that constraint networks being compared for subsumption have been fully completed using the constraint propagation and domain-specific inference techniques of Sections 4.3.4 and 4.3.6. Then, we define constraint network subsumption in general terms as follows:

Definition 36 *One network concept* subsumes *another iff every possible instance of the second network concept is also an instance of the first.*

T-REX specializes this idea to compute plan (body) subsumption. The body of plan *P1 subsumes* that of *P2* iff the set of possible action patterns described by *P1* is a superset of those described by *P2*. Thus, plans can be automatically *classified* in a strict taxonomy where each plan subsumes its descendants and is subsumed by its ancestors. In this respect, plan subsumption in T-REX resembles previous work on plan subsumption [Devanbu and Litman, 1991; Wellman, 1990], but provides a far richer temporal representation language and adds co-reference constraints.

We will begin by considering *structural* constraint network subsumption, which follows directly from node subsumption and arc subsumption.

4.4.1 Node Subsumption

When we model the semantics of nodes with concepts, node subsumption is based directly on concept subsumption:

Definition 37 *Node* n1 subsumes *node* n2 *iff the concept associated with* n1 *subsumes the concept of* n2.

For example, if the ATTEND-CONFERENCE concept subsumes the ATTEND-AI-CONFERENCE concept, then a node modeled by ATTEND-CONFERENCE subsumes a node modeled by ATTEND-AI-CONFERENCE.

4.4.2 Arc Subsumption

Structurally speaking, arc subsumption follows from the associated constraints. In some applications, arc semantics might also be modeled with description logic concepts. For QME networks, we use a special-purpose representation for each type of constraint and accordingly we give special definitions to handle QME arc subsumption.

First, we consider qualitative constraints:

Definition 38 *Qualitative constraint* Q1 subsumes *qualitative constraint* Q2 *iff* Q1*'s disjuncts are a superset of* Q2*'s disjuncts.*

For example, *before* $\vee$ *after* subsumes *before*, *after*, and $before \vee after$.

Second, we consider metric constraints. Subsumption between metric constraints is based in part on the numeric intervals they embody. Since numeric intervals may be open or closed on either end, and we represent this in metric constraints with the inequality operators $\leq$ and $<$, we need to define subsumption between these inequality operators:

Definition 39 *Inequality relations* $\leq$ *and* $<$ *both* subsume *inequality relation* $<$.

Recall Definition 5 on page 43, the definition of metric constraints. Our formal definition of metric constraint subsumption requires both numeric interval containment and reference to the same interval extrema in the same order:

Definition 40 *Metric constraint* $m1\ Q1\ i1_F - j1_G\ R1\ n1$ subsumes *metric constraint* $m2\ Q2\ i2_F - j2_G\ R2\ n2$ *iff:*

1. $(m1 < m2) \vee ((m1 = m2) \wedge (Q1$ *subsumes* $Q2))$, *and*
2. $(n1 > n2) \vee ((n1 = n2) \wedge (R1$ *subsumes* $R2))$

Note that the presence of F and G in both metric constraints enforces the requirement that they reference the same interval extrema. For instance, consider the following metric constraint with magnitude (1,4] from the finish of *step1* to the start of *step2*:

$$1 < \text{step1}_{finish} - \text{step2}_{start} \leq 4$$

It subsumes the next constraint with magnitude [2,4) from the finish of step3 to the start of step4:

$$2 \leq \text{step3}_{finish} - \text{step4}_{start} < 4$$

Notice that the numeric interval of the first metric constraint contains that of the second. Also the same extrema are involved, and in the same order, i.e., the constraints relate the finish of one interval to the start of another. This metric constraint subsumption relationship would help support a network subsumption relationship if nodes *step*1 and *step*2 in the subsumer are matched with nodes *step*3 and *step*4 in the subsumee, respectively.

Finally, when it comes to binary equality constraints, subsumption is a matter of role identity:

Definition 41 *Binary equality constraint* $\mathrm{R}_a = \mathrm{S}_b$ subsumes $\mathrm{R}_c = \mathrm{S}_d$

For example:

$$agent_{step1} = recipient_{step2}$$

subsumes

$$agent_{step3} = recipient_{step4}$$

assuming only that the concepts associated with the steps have the indicated roles. In the context of constraint network subsumption, this equality constraint subsumption relationship could support a network subsumption relationship if nodes *step*1 and *step*2 in the subsumer are matched with nodes *step*3 and *step*4 in the subsumee, respectively.

In general, subsumption between a pair of arcs is determined by subsumption between the constraints that label them:

Definition 42 *Arc* a1 subsumes *arc* a2 *iff the constraint associated with* a1 subsumes *the constraint associated with* a2.

Having defined node and arc subsumption, we are ready to use them in the context of network subsumption.

4.4.3 Network Subsumption

Structural subsumption between constraint networks relies on establishing a suitable correspondence between their constituents. Based on node and arc subsumption, we can define a *structural subsumption mapping* between a pair of completed constraint network concepts:

Definition 43 *A* structural subsumption mapping *from network concept* N1 *to network concept* N2 *maps every node* n1 *of* N1 *to a distinct node* n2 *of* N2 *such that:*

1. n1 *subsumes* n2
2. *for all arc types T, every arc of type T between a pair of nodes in* N1 *subsumes the corresponding arc of type T in* N2.

In fact, only *N2* needs to be completed for this definition to hold.

We illustrate subsumption mapping in Figure 4.8, which includes two of the plan networks from Figure 4.3. Dashed arrows illustrate a mapping from nodes in the subsumer, PLAN-A, to nodes in the subsumee, PLAN-B. Notice that the two plans differ in the number and specificity of their actions, and likewise for the relevant constraints. This notion of structural subsumption is analogous to standard description logic, where concepts may specialize their parents by further restricting their roles and/or restricting additional roles.

Sometimes there may be more than one subsumption mapping between a pair of networks. Suppose ATTEND-MEETING subsumes both ATTEND-WORKSHOP and ATTEND-CONFERENCE. Then Figure 4.9 demonstrates two different subsumption mappings from the subsumer to the subsumee. One is shown with wide dashed arrows and the other with wide solid arrows. Either mapping by itself is sufficient to establish the subsumption relationship.

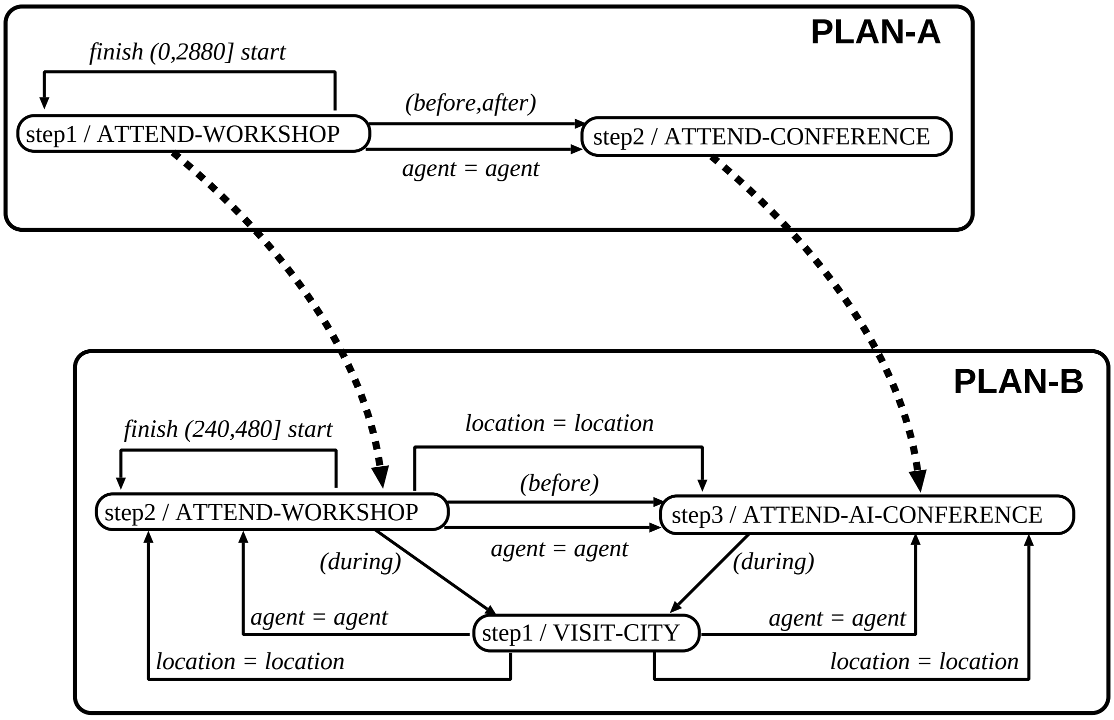


Figure 4.8: Subsumption Mapping

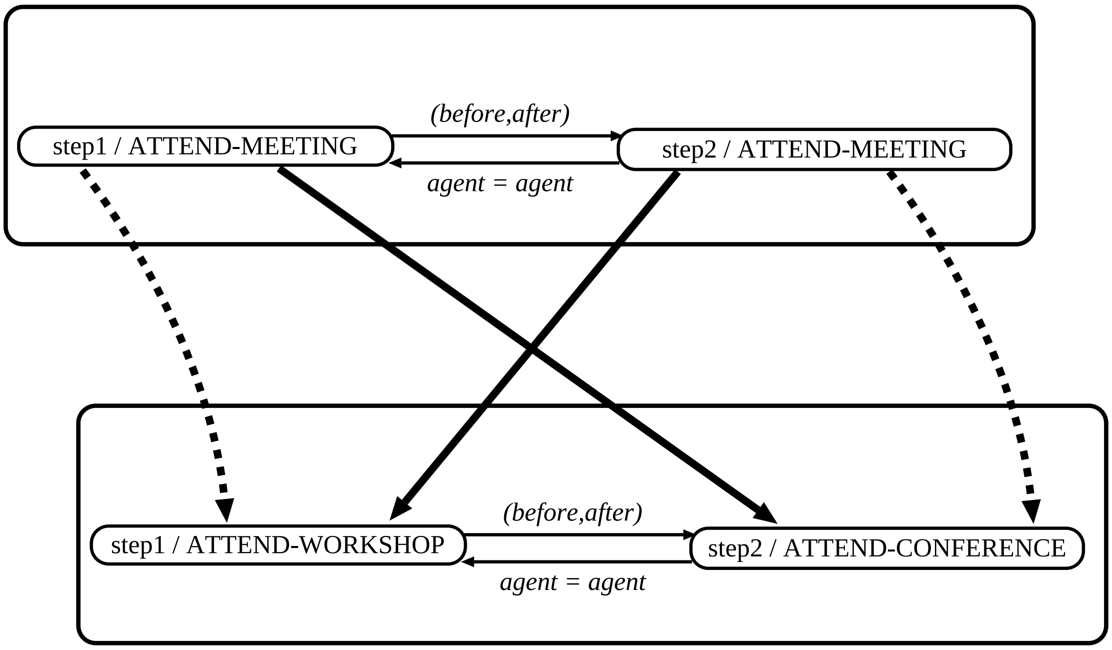


Figure 4.9: Multiple Subsumption Mappings

Existence of a structural subsumption mapping is necessary and sufficient to establish subsumption between constraint networks, given complete constraint propagation within the networks. Then we have the following result:

Theorem 5 *Network concept* N1 *subsumes network concept* N2 *iff there exists a structural subsumption mapping from* N1 *to* N2 *(assuming that closure of* N2 *is complete).*

Proof: See Appendix B.

As with standard description logic, this is written $N2 \Rightarrow N1$.

Constraint propagation in T-REX is not strictly complete, so certain caveats are in order with respect to our implementation:

- Complete node subsumption relies on the completeness of concept subsumption. For example, in CLASSIC concept subsumption is only complete under non-standard semantics [Borgida and Patel-Schneider, 1994]. K-REP is incomplete with respect to *some*, the existential role restriction.
- Subsumption with respect to qualitative arcs depends on complete propagation of qualitative constraints across the network. Qualitative constraint reasoning in MATS is incomplete [Kautz and Ladkin, 1991]. In the presence of macro steps, propagation of constraints to their substeps is done via Song and Cohen's incomplete algorithm.
- By default, T-REX assumes that nodes and arcs can be considered independently of one another. It relies on users to supply any required node-to-arc and arc-to-node inferences using the rule facility of Section 4.3.6.

Let us consider some examples of completion inferences that encompass both nodes and arcs. As an example of a node-to-arc inference, it may be possible to infer that two acts are temporally disjoint by virtue of their semantics. Examples include

C-FLY-PLANE and C-TAKE-TRAIN when the agent is identical. Alternatively, one might make an arc-to-node inference. If C-FLY-PLANE and C-TAKE-TRAIN do, in fact, coincide, then their agents must be different. For another general example of a node-to-arc inference, suppose that the action of *step1* produces something that is consumed by the action of *step2*. The exact implication for the temporal relationship between *step1* and *step2* depends on the nature of the producer-consumer relationship, but it is clear that *step2* cannot precede *step1*.[11]

As we have said, Theorem 5 assumes that nodes can be considered independently of arcs and *vice versa*. In fact, T-REX provides a rule-based system for expressing domain-specific constraints, including constraints involving both nodes and arcs. However, there is a *domain-independent* case for QME networks: an equality arc in the subsumer is fully satisfied if the corresponding roles in the subsumee (1) are filled by the same set of individuals, and (2) have explicit fillers equal in number to their at-most restrictions. We take this into account with a definition of subsumption mapping that is specialized for QME networks:

Definition 44 *A* subsumption mapping *from QME network concept* N1 *to QME network concept* N2 *maps every node* n1 *of* N1 *to a distinct node* n2 *of* N2 *such that:*

1. n1 *subsumes* n2
2. *every temporal (qualitative or metric) arc between a pair of nodes in* N1 *subsumes the corresponding temporal arc in* N2
3. *for every ordered pair of nodes* m1 *and* n1 *of* N1 *mapped to* m2 *and* n2 *of* N2*, respectively, and for every equality arc* R = S *from* m1 *to* n1*:*
   (a) *there is an equality arc* R = S *from* m2 *to* n2*, or*
   (b) *all of the following hold:*

[11] This was pointed out by a participant in the AAAI Fall Symposium, *Issues in Description Logics: Users Meet Developers*, Cambridge, MA, October 1992.

$$i.\ \mathit{fillers}(\mathrm{R_{m2}}) = \mathit{fillers}(\mathrm{S_{n2}})$$

$$ii.\ |\,\mathit{fillers}(\mathrm{R_{m2}})\,| = \textit{at-most}(\mathrm{R_{m2}})$$

$$iii.\ |\,\mathit{fillers}(\mathrm{S_{n2}})\,| = \textit{at-most}(\mathrm{S_{n2}})$$

For example, Figure 4.10 reproduces PLAN-A from Figure 1.1. It then introduces an individual named TONY and a pair of concepts which specialize ATTEND-WORKSHOP and ATTEND-CONFERENCE by restricting each of them to a single filler of the agent role, namely TONY. Finally, it introduces PLAN-T which is subsumed by PLAN-A according to Definition 44. Although PLAN-T has no equality constraint between the agent roles of its constituent actions, both actions are restricted to having the same filler by their own definitions, hence the equality constraint of PLAN-A is satisfied by PLAN-T.

Definition 44, implemented by T-REX, is not a purely structural definition. However, we could give a structural definition of QME network subsumption mapping, on the following assumption: the subsumee has been preprocessed during network completion such that an equality arc is added for every pair of closed roles (on any pair of concepts in the network) whose fillers are known to be identical.

Now that we have defined constraint network subsumption, the next section investigates its complexity.

4.4.4 Complexity (theory)

It is worthwhile to investigate the computational complexity of constraint network subsumption. Determining node subsumption amounts to computing subsumption between the associated concepts. Since we assume that the concept taxonomy is defined in advance, we can precompute subsumption relations between pairs of concepts. Thus we can retrieve the results in constant time. Qualitative temporal constraints can be encoded as bitstrings of length 13, so qualitative

In PLAN-A, an agent attends a workshop for at most 2880 minutes, either before or after attending a conference.

(define-plan **PLAN-A**
  :steps ((step1 ATTEND-WORKSHOP)
    (step2 ATTEND-CONFERENCE))
  :qualitative-constraints ((step1 (*before* $\vee$ *after*) step2))
  :metric-constraints ((0 $<$ step1$_{finish}$ $-$ step1$_{start}$ $\leq$ 2880))
  :co-reference-constraints ((agent(step1) = agent(step2))))

(create-individual TONY)

(define-concept ATTEND-WORKSHOP-T
  (and ATTEND-WORKSHOP
    (exactly 1 agent)
    (fills agent TONY)))

(define-concept ATTEND-CONFERENCE-T
  (and ATTEND-CONFERENCE
    (exactly 1 agent)
    (fills agent TONY)))

In PLAN-T, the agent named Tony attends a workshop for at most 2880 minutes, either before or after attending a conference.

(define-plan **PLAN-T**
  :steps ((step1 ATTEND-WORKSHOP-T)
    (step2 ATTEND-CONFERENCE-T))
  :qualitative-constraints ((step1 (*before* $\vee$ *after*) step2))
  :metric-constraints ((0 $<$ step1$_{finish}$ $-$ step1$_{start}$ $\leq$ 2880)))

Figure 4.10: Non-Structural Plan Subsumption

arc subsumption can also be computed in constant time. If positive and negative bits encode the presence and absence of qualitative disjuncts, respectively, then bitstring *B1* subsumes bitstring *B2* just in case the bitwise-and of *B1* with the bitwise-complement of *B2* is zero. Metric constraint subsumption can be computed directly from Definition 40. Obviously, this too is a constant-time operation.

We begin our further analysis with constraint networks having only qualitative and metric arcs, which we call QM networks. After completion, there are two qualitative arcs and four metric arcs between every pair of nodes in a QM network. Of these, one qualitative arc and two metric arcs are redundant inverses of the others. The crucial part of the QM network subsumption problem is to establish a suitable mapping from the nodes of the subsumer to the nodes of the subsumee. This problem is clearly in NP, as one can guess a subsumption mapping and check it with at most n node subsumption tests, n^2 qualitative arc subsumption tests, and $(2n)^2$ metric arc subsumption tests.[12] There exists a polynomial time transformation from directed subgraph isomorphism, which is NP-complete, to QM network subsumption. Thus we have:

Theorem 6 *Subsumption mapping between QM networks is NP-complete.*

Proof: See Appendix B.

A (directed) graph is *complete* in the graph-theoretic sense if there exists a (directed) edge from every vertex to every other vertex. In, MATS there are qualitative and metric arcs between every pair of nodes, making the QM graph complete. Although subgraph isomorphism is trivial when both graphs are complete, subsumption mapping between a pair of complete QM network concepts can nonetheless be reduced from the general subgraph isomorphism problem. The preceding result still holds:

Corollary 1 *Subsumption mapping between complete QM networks is NP-complete.*

[12] With absolute metric constraints in plans, it is $(2n+1)^2$ metric arc subsumption tests.

Proof: See Appendix B.

For QME networks, the problem size is characterized by the number of nodes plus the number of binary equality constraints. Since subsumption with respect to each binary equality constraint in the subsumer is trivial via part 3 of Definition 44, we have:

Theorem 7 *Subsumption mapping between complete QME networks is NP-complete.*

Proof: See Appendix B.

It has been remarked that all interesting problems in artificial intelligence are intractable in the computational complexity sense. The challenge is to design algorithms that make the best of this situation. Our algorithm for constraint network subsumption is presented next.

4.4.5 Computation (practice)

Although we have shown that constraint network subsumption is NP-complete, the networks generally contain a great deal of information that can be used for heuristic guidance. It is important to observe that in our instance of the subgraph isomorphism problem, both nodes and arcs are labeled, so powerful heuristics can be brought to bear. The labels promote quick results, both positive and negative: matching labels help guide the search to a successful conclusion when a mapping exists, and mismatching labels lead to early failure when there is no mapping. This section presents a strategy for computing constraint network subsumption in practice.

Constraint network subsumption can be cast in terms of the widely-studied constraint satisfaction problem (CSP), which itself is often formulated in terms of constraint networks. The constraint satisfaction problem underlies many important problems in AI and computer science in general. A useful survey is [Kumar, 1992]. A CSP consists of a set of variables (that correspond to nodes in the putative subsumer). A solution entails finding an instantiation for each variable from a set of values in a given domain (nodes in the putative subsumee) under certain constraints. For QME networks, there are unary constraints (concepts associated with the nodes) plus binary constraints (relationships described by the arcs). Because of the connection with CSP, constraint network subsumption is amenable to a variety of techniques developed by the CSP community, e.g., *arc consistency* [Mackworth, 1977], *forward checking*, etc. Choosing the optimal mix is domain-dependent and largely still a black art [Kumar, 1992].

We now introduce an algorithm to decide whether QME network concept *N1* subsumes QME network concept *N2*. This algorithm combines constraint satisfaction techniques with node and arc subsumption. It can easily be adapted to similar problems. Our strategy is to exploit the available information about partial orders, including the ordering among concepts induced by subsumption and the temporal precedence order induced by qualitative and metric constraints. We proceed in four phases, the last of which conducts the crucial search for a mapping from one network to the other:

1. **Macro Expansion:** Expand each macro node by replacing it with its constituent nodes (recursively). Propagate constraints on a macro node to each of its constituents. In practice this is done only once, when each network is defined. After macro expansion, if *N2* does not have at least as many nodes as *N1*, return *false*.

2. **Closure:** Close both networks via constraint propagation. If domain-specific inference rules have been supplied, they are factored in during this phase. In practice this too is done only once, when a network is defined.

3. **Preliminary Analysis:** First, topologically sort the nodes of *N1*, and likewise the nodes *N2*, by temporal precedence (in practice this done once, when each network is defined). For this partial order, we assert that node a precedes node b just in case a necessarily either starts earlier than b, or

starts concurrently with and ends earlier than b. This occurs when we have $a\ (before \vee meets \vee overlaps \vee starts)\ b$.

Second, for each node *n1* in *N1*, determine which nodes in *N2* are subsumed by *n1* according to the associated concepts and arcs from the nodes to themselves. Call those nodes the *potential images* of *n1*.[13] If the number of potential images for any node in *N1* is zero, return *false*. Otherwise, stably sort the nodes of *N1* in increasing order of potential image count to help guide the subsequent graph matching process.[14] As a result, wherever possible, ties in potential image count are broken by the temporal partial order.

4. **Matching by Backtracking:** Using the preliminary analysis for heuristic guidance, extend the mapping from *N1* to *N2* one step at a time. Each extension consists of selecting an additional node *p1* from *N1*, and associating with it an additional node *p2* from among its potential images in *N2*, such that the constraints on all nodes selected from *N2* continue to respect the constraints on the corresponding nodes from *N1*, i.e., for every previously selected node *q1* of *N1* mapped to node *q2* of *N2*, we must verify the following:

   - **Qualitative Arcs:** $p1(r_1 \ldots r_n)q1$ subsumes $p2(r_1 \ldots r_m)q2$. This guarantees subsumption for the redundant inverse arc, so the latter need not be checked separately.
   - **Metric Arcs:** The four metric arcs from *p1*'s extrema to *q1*'s extrema must be respected by the corresponding extrema of *p2* and *q2* under the mapping. In terms of the associated constraints, all of the following must hold:

[13]If *n2* is in *n1*'s set of potential images, $i1\ Q1\ n1_{finish} - n1_{start}\ R1\ j1$, denoting *n1*'s duration, must subsume $i2\ Q2\ n2_{finish} - n2_{start}\ R2\ j2$, denoting *n2*'s duration. (Since an interval's starting point is always before its finishing point, $n_{start} - n_{finish} < 0$. [Kautz and Ladkin, 1991]). The qualitative temporal relationship from a node's interval to itself is trivially *equal*. The metric temporal relationship from an endpoint of a node's interval to itself is trivially zero. Equality arcs from a node to itself are also handled at this stage.

[14]Since the sort is stable, temporal precedence is preserved among nodes with the same number of potential images.

(a) $p1_{start}(-m1, Q1, R1, n1)q1_{start}$
subsumes
$p2_{start}(-m2, Q2, R2, n2)q2_{start}$

(b) $p1_{start}(-m1, Q1, R1, n1)q1_{finish}$
subsumes
$p2_{start}(-m2, Q2, R2, n2)q2_{finish}$

(c) $p1_{finish}(-m1, Q1, R1, n1)q1_{start}$
subsumes
$p2_{finish}(-m2, Q2, R2, n2)q2_{start}$

(d) $p1_{finish}(-m1, Q1, R1, n1)q1_{finish}$
subsumes
$p2_{finish}(-m2, Q2, R2, n2)q2_{finish}$

In other words, for $F, G \in \{start, finish\}$ we have:

$p1_F(-m1, Q1, R1, n1)q1_G$
subsumes
$p2_F(-m2, Q2, R2, n2)q2_G$

The four redundant metric arcs from the extrema of *q1* to those of *p1* need not also be checked against their counterparts from *q2* to *p2*.

- **Equality Arcs:** Any equality arc $R = S$ from a role of *p1* to a role of *q1* must be respected by *p2* and *q2*. Either:

(a) There exists an identical equality constraint $R = S$ from *p2* to *q2*, or

(b) R_{p2} and S_{q2} are both closed, and fillers(R_{p2}) $\equiv$ fillers(S_{q2}).

When each node from *N1* has been mapped to a distinct node from *N2*, return the mapping, which constitutes a successful result. At any point, if there is a node from *N1* that cannot be mapped in the manner described, backtrack. If the backtracking process is exhausted without finding a suitable mapping, return *false*.

As a simple example, we will trace the derivation of the subsumption mapping shown in Figure 4.8 (page 122) from PLAN-A to PLAN-B as defined in Figure 1.1 (page 9). We will refer to nodes by the names of the corresponding steps. During preliminary analysis, the temporal precedence criterion does not affect node ordering in PLAN-A, but *step2* of PLAN-B is ordered before *step3* of PLAN-B. Next, we determine potential images for the nodes of PLAN-A. Notice that *step1* of PLAN-A and *step2* of PLAN-B have the same associated action, and the duration of the former subsumes the duration of the latter. Notice also that *step2* of PLAN-A has an associated action which subsumes the associated action of *step3* of PLAN-B, and both have unconstrained durations. Thus, each node of PLAN-A is found to have a single potential image in PLAN-B:

Node of PLAN-A	**Potential Image in PLAN-B**
step1 / ATTEND-WORKSHOP	step2 / ATTEND-WORKSHOP
step2 / ATTEND-CONFERENCE	step3 / ATTEND-AI-CONFERENCE

Then, the nodes of PLAN-A are stably by sorted by potential image count, which again leaves their order unchanged. Therefore, *step1* of PLAN-A is considered first during the matching by backtracking phase, and it is mapped to *step2* of PLAN-B. Next, the mapping is extended by mapping *step2* of PLAN-A to *step3* of PLAN-B. Each arc from *step3* to *step2* in PLAN-B which has a counterpart from *step2* to *step1* in PLAN-A is successfully verified against its counterpart[15]; note that these arcs happen to be the inverses of the explicit arcs shown in Figure 4.8 (they result from closure of the network via constraint propagation):

	PLAN-A *step2* **to** *step1*	**PLAN-B** *step3* **to** *step2*
Qualitative Arc	(before, after)	(after)
Co-reference Arc	agent = agent	agent = agent

[15]The co-reference constraint between the locations has no counterpart.

All the metric arcs involved are trivial defaults, so they are not shown. Since every node of PLAN-A has now been successfully mapped to a distinct node of PLAN-B, the algorithm has produced a successful result.

An alert reader will notice that there is considerable overlap in the expressive power of qualitative and metric temporal constraints, hence the qualitative and metric constraint networks underlying T-REX plans may contain substantial redundancy. For example, if the ending point of INTERVAL1 is 5 time units less than the starting point of INTERVAL2, this implies that INTERVAL1 is *before* INTERVAL2. Therefore, a plan subsumption algorithm that verified the first constraint need not also verify the second. Conversely, the current T-REX subsumption algorithm verifies qualitative constraints before metric constraints. Thus, when it comes to metric constraints, T-REX could ignore all those with linear inequalities whose numeric operand is zero or infinity. However, we must balance the cost of duplicated effort against the cost of identifying the duplication. Minimizing redundant tests is a desirable goal. Comparing metric constraints, however, is quite inexpensive so it is unlikely that this strategy would be very worthwhile.

The above algorithm puts all available information to effective use. It is similar to an existing algorithm for production rule subsumption due to Yen [Yen, 1990], whose work is reviewed in Section 6.5. Our optimism is reinforced by Yen's normal case analysis for subsumption of production rules composed of description logic concepts. That algorithm's complexity was found to be polynomial in "normal" cases [Yen, 1990]. Empirical performance analysis of our algorithm is reported in Section 4.5. Before turning to performance analysis, however, we probe more deeply into the nature of constraint network subsumption.

4.4.6 Comparison with Standard Description Logic

It is worth noting some fundamental distinctions between constraint network subsumption and standard concept subsumption. All comments in this section about subsumption apply equally to traditional recognition. If one standard concept is to subsume another, every role restriction on the first concept must subsume a corresponding role restriction on the second. The correspondence between the restrictions is "predetermined" by the names of the roles, i.e., corresponding restrictions always refer to the *same role*. For example, if the subsumer has an at-least restriction on role R, we need only check whether the subsumee has one too, and if so, whether the magnitude is at least as great. In contrast, the labels of nodes in our constraint networks are considered meaningless for subsumption, i.e., any node in the subsumer can potentially match any node in the subsumee, regardless of their labels. Consequently, constraint network subsumption requires combinatorial searching for a correspondence between nodes. In the case of plans, the effect is that a plan's steps are not treated as functional. Of course, steps of a plan *can* be seen as functional, but often the function is difficult to elucidate in a useful way, e.g., the function of the VISIT-ITALY action *in the context of* a TOUR-EUROPE plan is precisely to visit Italy. Moreover, when a plan entails several identical actions, e.g., visiting Italy twice, their functionality may be practically indistinguishable. If we matched nodes by name, users would be required to give steps names corresponding to their function and use the names consistently across plans. It appears that using such a system would be significantly more cumbersome.

A related distinction, pointed out in Section 4.4.3, is the possibility of multiple subsumption mappings from the subsumer to the subsumee. This doesn't matter for subsumption testing, because any one subsumption mapping is sufficient to decide the issue.

Another distinction concerns the T-REX implementation of constraint networks. All network concepts are considered *fully defined*, i.e., non-primitive. However, there is no apparent technical obstacle to implementing primitive constraint networks, should that seem useful.

4.4.7 Conclusion

Section 4.4 has examined the theory and practice of constraint network subsumption. The subsumption inference is crucial both to classifying libraries of constraint networks, which we cover in Section 4.6, and to our deductive recognition methodology for constraint networks, which we come to in Chapter 5. Empirical performance analysis of our subsumption algorithm is reported next.

4.5 Empirical Analysis

In order to validate our approach to constraint network subsumption, it seemed wise to test the performance of our algorithm on a reasonable variety of large and/or relatively difficult plan subsumption problems. Our goal is to show that the algorithm is feasible for interactive use when presented with such problems. This section describes the design of experiments with our constraint network subsumption algorithm, and then presents our experimental results. Our experiments are conducted through a *performance analysis workbench* program composed of a problem generator, an experiment administrator, and an experiment reporter. This is in keeping with a recent study on the performance of standard description logic systems [Baader *et al.*, 1992; Heinsohn *et al.*, 1992b], which used machine-generated knowledge bases (as well as several "real world" knowledge bases).

Design of Experiment

First we describe the design for an empirical study of the performance and scalability of our constraint network subsumption algorithm. For simplicity, we experimented on Q networks[16] with the algorithm currently implemented in T-REX and described in Section 4.4.5. Several factors influence the cost of computing subsumption, including:

[16] Q networks have only qualitative temporal constraints between nodes.

- Number of nodes in each network
- Variation among the concepts that label the nodes
- Relative generality of concepts associated with nodes (their level in the taxonomy)
- Uncertainty (disjunctiveness) of temporal constraints between nodes

To measure cost, we again instrumented our code with the METERING system [Kantrowitz, 1991]. Recall that METERING allows us to count basic operations and gather statistics on timing and storage utilization without modifying the existing T-REX code. Moreover, it adjusts timing statistics to discount overhead involved in the monitoring itself. We use a *problem generator* program to synthesize subsumption problems with differing characteristics based on input parameters, or *knobs*. Our experiment focuses on cases where a subsumption mapping exists. To construct such a subsumption problem, we first generate a subsumee and abstract from it to produce a subsumer. The *experiment administrator* explores the problem space by selecting combinations of *settings* for the knobs. For each combination of settings, the problem generator produces a representative sample of subsumption problems and then the experiment administrator solves them using T-REX to gather performance statistics.

Each node of a constraint network concept has an associated concept chosen from some terminology. While performance is indifferent to the semantic import of these concepts, it is influenced by the presence or absence of subsumption relations among them. We want to draw concepts from a terminology that offers sufficient flexibility without undue complexity. Thus, we use a terminology composed of primitive concepts in the form of a complete binary tree.[17] For instance, the terminology might have 255 concepts, including 128 leaves. Notice that because all concepts are primitive, the set of concepts on any level are mutually disjoint,

[17]The fact that the concepts are primitive makes it easy to generate the binary tree. The entire concept taxonomy is classified before constraint network subsumption is tested, so their primitiveness does not otherwise affect our experiment.

i.e., there are no subsumption relations among them. A synthesized terminology will have a somewhat unnatural "balance", but this should not impact particular constraint network subsumption problems. We now turn to the knobs that control constraint network generation.

Several knobs specify characteristics of the subsumee:

1. The *SUBSUMEE-NODE-COUNT* knob controls the number of nodes in the subsumee, n_2. Cost should naturally increase along with this number. We simply use powers of two, e.g., through 128. The last of these seems like an ample upper bound, considering the granularity at which plans are likely to be described.

2. *SUBSUMEE-DISTINCT-COUNT* controls the degree of variation among (concepts associated with) nodes in the subsumee. For simplicity, we stipulate that these concepts are chosen in equal numbers from a single level in the concept taxonomy. Then, variation is measured by the number of distinct concepts present, m_2, which we restrict to a power of two. We choose these concepts from the level in the taxonomy that contains exactly m_2 concepts. In general, the subsumption problem is most difficult when each node in the subsumee has the same associated concept, because this provides minimal information. It is easiest when each node has a different associated concept, providing maximal information. With respect to T-REX's subsumption algorithm, this minimizes the potential image sets of nodes in the subsumer. Current limitations of the problem generator prevent experiments which illustrate this phenomenon. In fact, this knob currently has no direct effect on performance because the concept taxonomy is a tree, all concepts associated with nodes in the subsumer are chosen from a single level of the tree, and all nodes in the subsumee are likewise chosen from a single level. As a result, this knob does not currently impact the size of potential image sets.

3. *SUBSUMEE-SEQUENCE-COUNT* varies the uncertainty of qualitative temporal constraints in the subsumee by controlling sequencing among nodes.

This knob breaks the subsumee network into a number of sequences of equal size. At one extreme, all nodes are arranged in a single sequence of length n_2, i.e., the nodes are totally ordered. At the other extreme, there are n_2 sequences containing a single node apiece, i.e., the nodes are totally unordered. Note that our aim here is simply to vary the temporal uncertainty of networks; we are not concerned with sequentiality, *per se*.

4. *SUBSUMEE-ORDER* controls the order of appearance of nodes in the subsumee network definition. The choices include topologically sorted in increasing or decreasing temporal order, and random order.

Taken together, these first four knobs control the generation of a subsumee network. The next three direct generation of its subsumer.

5. *SUBSUMER-NODE-COUNT* controls the number of nodes in the subsumer. While its setting, n_1, could range anywhere from one to n_2, we again restrict it to be a power of two. Again, cost should increase as this number does.

6. The *SUBSUMER-DISTINCT-COUNT* knob controls the degree of variation among nodes in the subsumer. Variation is again measured by the number of distinct concepts present in the network, m_1, which we restrict as usual to a power of two. For simplicity, we stipulate that concepts associated with nodes in the subsumer are chosen from the level in the concept taxonomy with exactly m_1 nodes, where m_1 is less than or equal to m_2. We randomly identify m_1 subsumee nodes, and for each one, include its (unique) subsumer from the concept taxonomy level having m_1 nodes. Therefore, this knob also controls the extent to which nodes in the subsumer generalize their counterparts in the subsumee. Cost should decrease as variation of nodes in the subsumer increases, as that decreases the size of the potential image sets of nodes in the subsumer.

- For simplicity, temporal constraints between nodes in the subsumer are identical to the constraints between their images in the subsumee.

7. *SUBSUMER-ORDER* is the counterpart of *SUBSUMEE-ORDER* for the subsumer. It controls the order of appearance of nodes in the subsumer network definition. Choices again include topologically sorted in increasing or decreasing temporal order, and random order.

Some observations about our experiment are in order:

- The space of possible experiments admitted by all combinations of all knobs is enormous. We therefore looked at a selection of examples that are difficult because they involve large numbers of nodes and/or large numbers of potential images for nodes in the subsumer.

- In the worst case, the subsumption problem amounts to subgraph isomorphism, which is NP-complete [Garey and Johnson, 1979]. This is shown in the proof of Theorem 6.

- When a network is defined, T-REX topologically sorts the nodes in ascending temporal order (for those pairs of nodes that are temporally ordered, as defined on page 129 for the preliminary analysis phase of the network subsumption algorithm).[18] In the case where both networks are totally ordered sequences of n nodes, and all nodes are labeled with the same concept, T-REX will find the subsumption mapping without backtracking by performing exactly n node mapping steps. If T-REX instead considered nodes in the order they were mentioned within the network definition, that would require $n!$ node mapping steps in the worst case.[19]

[18] Since the temporal sort occurs at plan definition time, it effectively renders knobs four and seven moot. It might however, be of some academic interest to disable the sort and determine by comparison when, and how much, temporal sorting contributes to the performance of our algorithm.

[19] Thanks to Sal Stolfo, who suggested experimenting with the order of nodes, e.g., by randomizing them, and thus prompted the idea of temporally sorting nodes.

Results of Experiment

This section reports on an empirical analysis of our constraint network subsumption algorithm. Detailed results are reported in a series of figures on the following pages. Each figure is produced entirely by our *experiment reporter*, a program which uses results of the experiment administrator program to generate LaTeX source files. As an example, consider Figure 4.11 on page 144. The upper left-hand portion of the figure reports the circumstances under which the experiment was conducted. The topology of the plan taxonomy is indicated by its depth and by the downward branching factor of its internal nodes, identified respectively as follows:

- *MAX-DEPTH*
- *BRANCHING-FACTOR*

Also shown are the settings of plan-related knobs that are held constant during the experiment. The knobs enumerated earlier in this section are identified as follows:

1. *SUBSUMEE-NODE-COUNT*
2. *SUBSUMEE-DISTINCT-COUNT*
3. *SUBSUMEE-SEQUENCE-COUNT*
4. *SUBSUMEE-ORDER*
5. *SUBSUMER-NODE-COUNT*
6. *SUBSUMER-DISTINCT-COUNT*
7. *SUBSUMER-ORDER*

In addition, *REPETITIONS* tells how many times the experiment was repeated. Following that, a "for" loop indicates how one of the plan-related knobs is varied during the experiment (the other six knobs are held constant). The syntax of the for loop is standard Common Lisp, as described in Section 26.6, *Iteration Control*, of [Steele, 1990].

In the case of Figure 4.11, the experiment generator produced twenty plan subsumption problems for each setting of *SUBSUMER-NODE-COUNT*, i.e., twenty problems where the subsumer has 8 nodes, twenty problems where the subsumer has 16 nodes, and so on. The graph in the lower left-hand portion of the figure reports the overall time required for computing plan subsumption (on the y-axis) versus the varying knob (on the x-axis). Minimum, maximum, and mean times at each setting are denoted by $\diamond$, $+$, and $\square$, respectively. It is important to note that all times should be considered as relative values. Specific times are dependent on numerous factors such as the particular (unoptimized) implementation and the particular platform (an IBM RISC System/6000 Model 390 running Allegro Common Lisp). The upper right-hand graph likewise reports the number of node subsumption tests performed for each setting. All node subsumption tests are carried out during the *preliminary analysis* phase of the subsumption algorithm as described on page 129. Therefore, the number of node subsumption tests is always the product of the number of nodes in the subsumer times the number of nodes in the subsumee. Finally, the lower right-hand graph reports the number of qualitative arc subsumption tests. In order for the subsumption algorithm to extend the subsumption mapping by mapping an additional node in the subsumer network to an additional node in the subsumee network, it must examine the arcs between those nodes and all previously mapped nodes to verify that arcs in the subsumer continue to subsume their counterparts in the subsumee under the mapping. Considering the implicit search tree explored by the subsumption algorithm during the *matching by backtracking* phase described on page 130, these numbers are bounded from above by the sum of the depths of the nodes in the search tree.

Figures 4.11 and 4.12 study the effect of varying the number of nodes in the subsumee. The experimental subsumption problems of Figure 4.12 are more difficult than their counterparts in Figure 4.11 (The settings of the *SUBSUMEE-

DISTINCT-COUNT*, *SUBSUMEE-SEQUENCE-COUNT*, and *SUBSUMER-NODE-COUNT* knobs are all multiplied by four in Figure 4.12). In both experiments, the overall time increases in proportion to the number of nodes in the subsumee. In Figure 4.11, performance is dominated by node subsumption tests. The graph of mean qualitative arc subsumption tests reflects the efficacy of temporally sorting nodes within a plan beforehand. (Since the sort is conducted once, when a plan is defined, there is no cost incurred for the temporal sort during plan subsumption testing.) In Figure 4.12, there are substantially more qualitative arc subsumption tests than in Figure 4.11, but they still total less than half the number of node subsumption tests, and indeed diminish, on average, as the subsumee node count increases.

Figures 4.13 and 4.14 examine the impact of varying the differentiation among nodes in the subsumee. We noted above that current limitations of the problem generator prevent this knob from having direct impact on performance. Given those limitations, the only real value of this experiment and the next one is to show that the subsumption algorithm is indeed well-behaved (predictable) at the settings indicated in the figure.

Figures 4.15 and 4.16 test the consequences of varying the extent to which nodes in the subsumee are sequenced (together with the corresponding nodes in the subsumer, as noted earlier in this section). Focusing on average case times, this variable has little effect in the experiments performed. This is because, as noted earlier, our problem generator simply makes temporal constraints between nodes in the subsumer identical to the constraints between their images in the subsumee.

Figures 4.17 and 4.18 study the effect of varying the number of nodes in the subsumer. The slight upward curve of the time and qualitative arc subsumption graphs might perhaps become an issue for subsumption testing with plans having substantially more nodes. However, we believe that larger scale applications, such as an extension of our travel plan application, will involve larger *numbers of* plans, as opposed to larger plans.

Figures 4.19 and 4.20 examine the impact of varying the differentiation among nodes in the subsumee. It is evident that increased differentiation speeds subsumption mapping because each node in the subsumer has fewer potential images in the subsumee. Particularly in Figure 4.19, the correlation between the number of qualitative arc subsumption tests and overall time is clear.

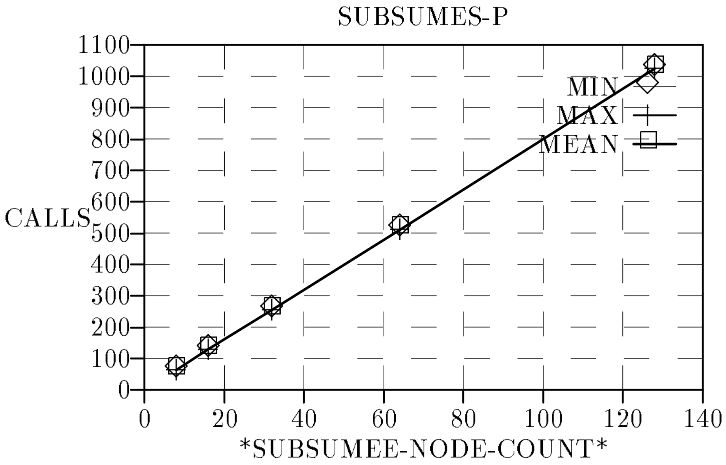
Our algorithm intuitively makes effective use of the available information, and our empirical results bear this out. While we make no claims as to our algorithm's optimality, the preceding experiments suggest that the algorithm is well-behaved when faced with a variety of "difficult" problems. Given our expectation that large scale applications may involve large numbers of plans, but not extremely large plans, our empirical results suggest that our approach is quite capable of supporting real-time, interactive applications.

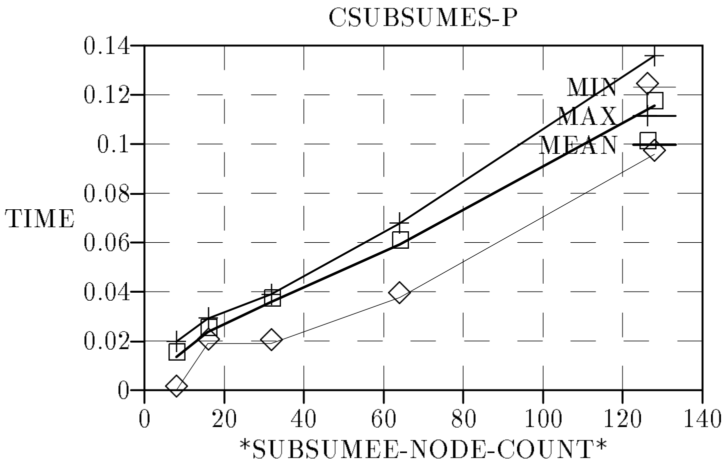
The following section goes into some more detail about organizing plan libraries via classification.

```
# Fixed knobs:
#
#    *MAX-DEPTH*                  7
#    *BRANCHING-FACTOR*           2
#    *SUBSUMEE-DISTINCT-COUNT*    4
#    *SUBSUMEE-SEQUENCE-COUNT*    2
#    *SUBSUMEE-ORDER*             :RANDOM
#    *SUBSUMER-NODE-COUNT*        8
#    *SUBSUMER-DISTINCT-COUNT*    4
#    *SUBSUMER-ORDER*             :RANDOM
#    *REPETITIONS*                20


Varying knob:

(FOR *SUBSUMEE-NODE-COUNT* = *SUBSUMER-NODE-COUNT* THEN
 (* 2 *SUBSUMEE-NODE-COUNT*) UNTIL (> *SUBSUMEE-NODE-COUNT* 128))
```





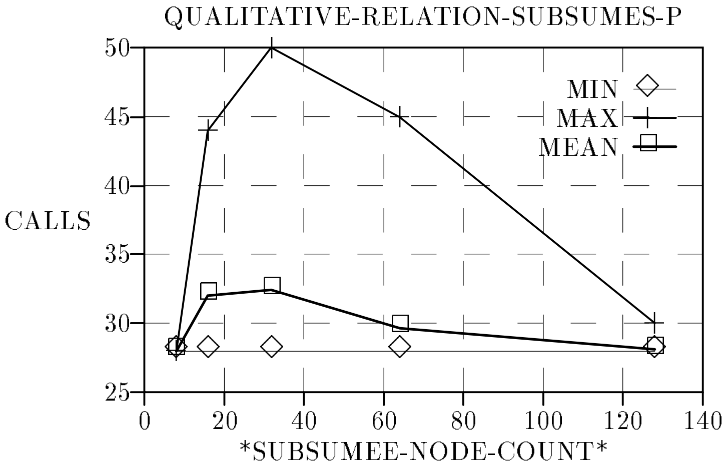
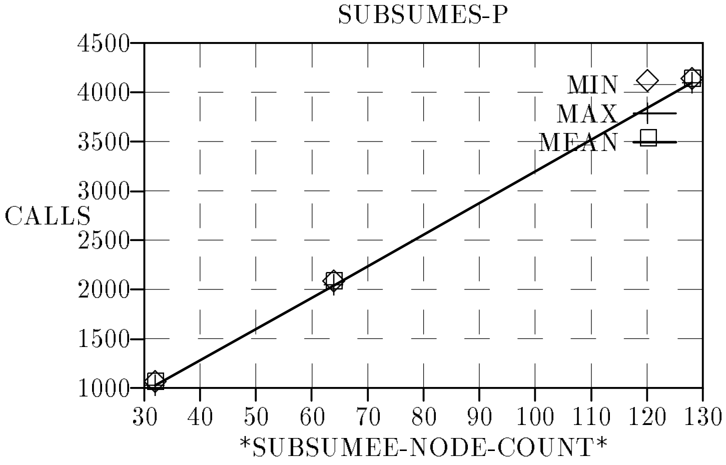


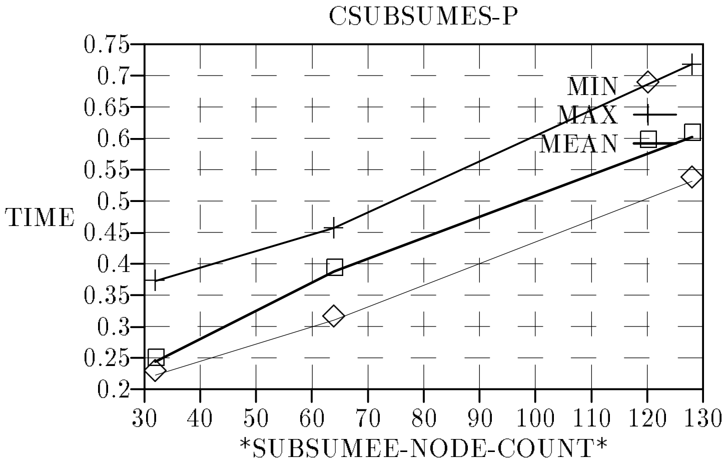
Figure 4.11: Subsumee Node Count Experiment A

```
# Fixed knobs:
#
#   *MAX-DEPTH*                 7
#   *BRANCHING-FACTOR*          2
#   *SUBSUMEE-DISTINCT-COUNT*  16
#   *SUBSUMEE-SEQUENCE-COUNT*  8
#   *SUBSUMEE-ORDER*           :RANDOM
#   *SUBSUMER-NODE-COUNT*      32
#   *SUBSUMER-DISTINCT-COUNT*  4
#   *SUBSUMER-ORDER*           :RANDOM
#   *REPETITIONS*              20


Varying knob:

(FOR *SUBSUMEE-NODE-COUNT* = *SUBSUMER-NODE-COUNT* THEN
 (* 2 *SUBSUMEE-NODE-COUNT*) UNTIL (> *SUBSUMEE-NODE-COUNT* 128))
```





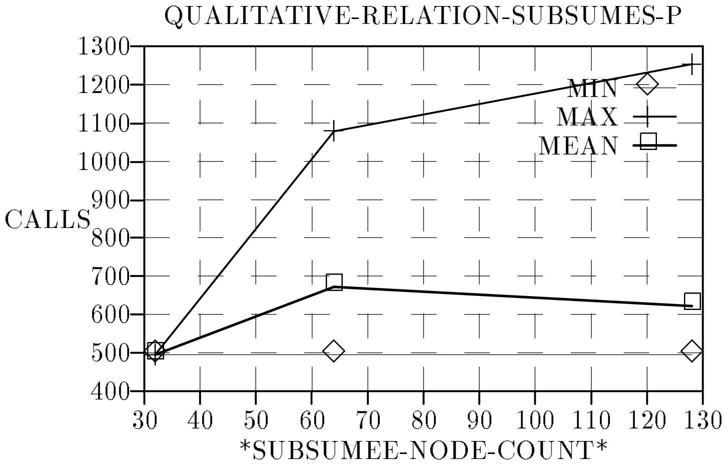
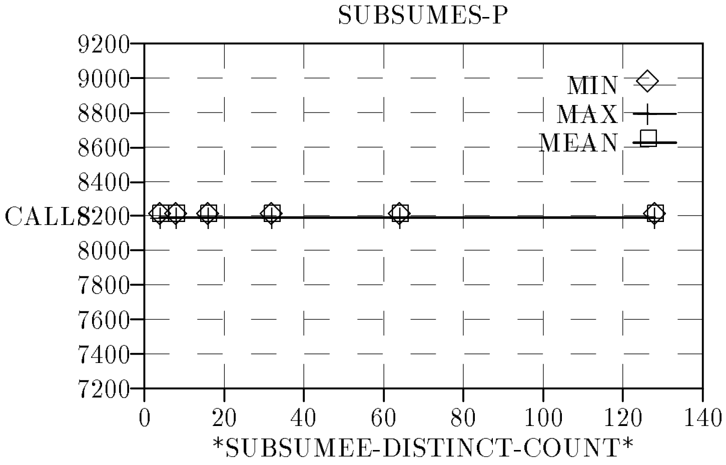


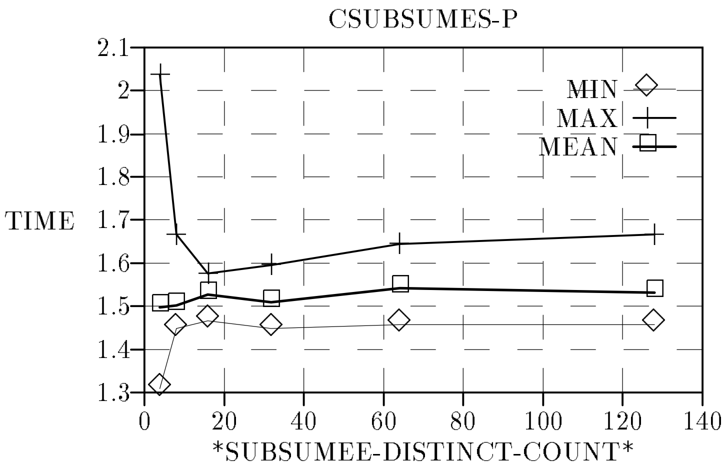
Figure 4.12: Subsumee Node Count Experiment B

```
# Fixed knobs:
#
#    *MAX-DEPTH*                 7
#    *BRANCHING-FACTOR*          2
#    *SUBSUMEE-NODE-COUNT*       128
#    *SUBSUMEE-SEQUENCE-COUNT*   2
#    *SUBSUMEE-ORDER*            :RANDOM
#    *SUBSUMER-NODE-COUNT*       64
#    *SUBSUMER-DISTINCT-COUNT*   4
#    *SUBSUMER-ORDER*            :RANDOM
#    *REPETITIONS*               20


Varying knob:

(FOR *SUBSUMEE-DISTINCT-COUNT* = *SUBSUMER-DISTINCT-COUNT* THEN
 (* 2 *SUBSUMEE-DISTINCT-COUNT*) UNTIL
 (> *SUBSUMEE-DISTINCT-COUNT* *SUBSUMEE-NODE-COUNT*))
```





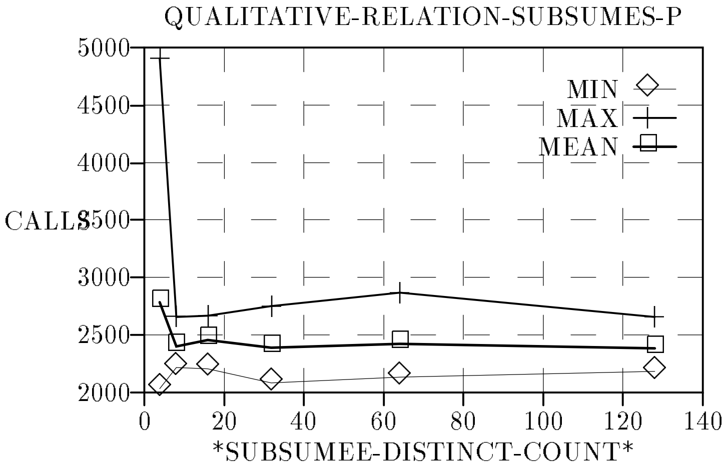
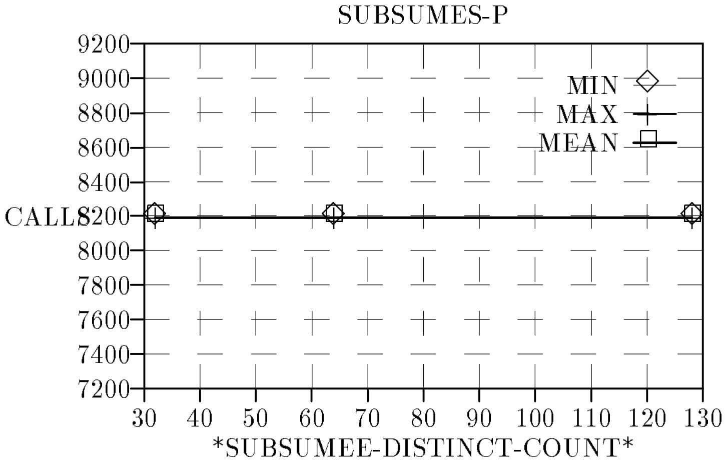


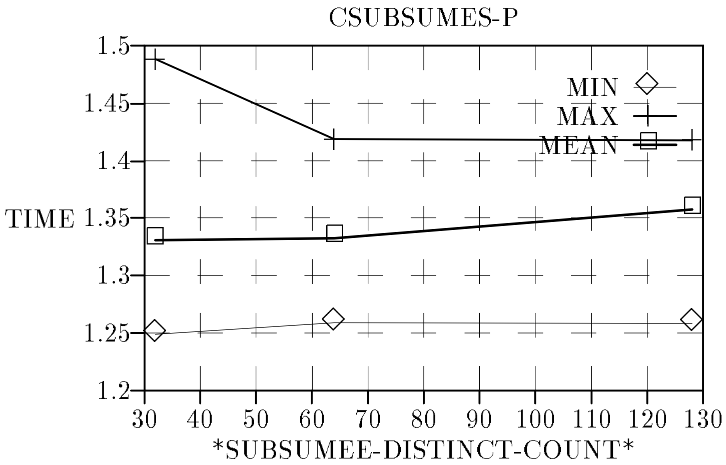
Figure 4.13: Subsumee Distinct Count Experiment A

```
# Fixed knobs:
#
#    *MAX-DEPTH*                  7
#    *BRANCHING-FACTOR*           2
#    *SUBSUMEE-NODE-COUNT*        128
#    *SUBSUMEE-SEQUENCE-COUNT*    2
#    *SUBSUMEE-ORDER*             :RANDOM
#    *SUBSUMER-NODE-COUNT*        64
#    *SUBSUMER-DISTINCT-COUNT*    32
#    *SUBSUMER-ORDER*             :RANDOM
#    *REPETITIONS*                20


Varying knob:

(FOR *SUBSUMEE-DISTINCT-COUNT* = *SUBSUMER-DISTINCT-COUNT* THEN
 (* 2 *SUBSUMEE-DISTINCT-COUNT*) UNTIL
 (> *SUBSUMEE-DISTINCT-COUNT* *SUBSUMEE-NODE-COUNT*))
```





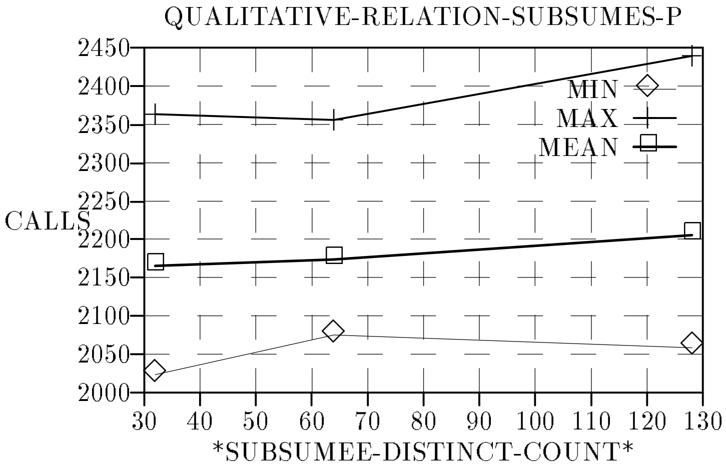
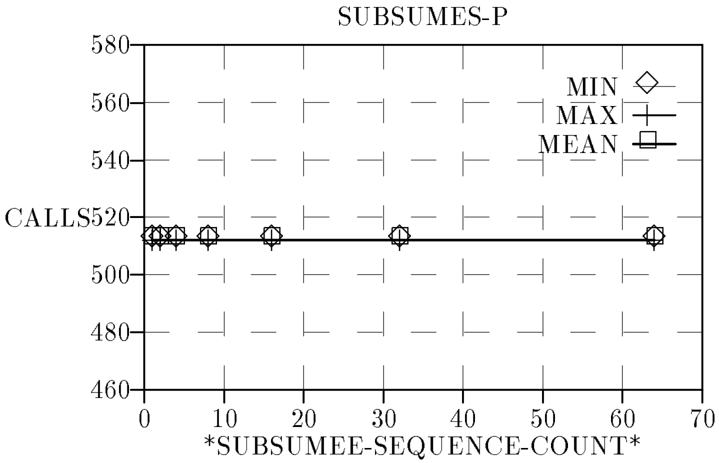


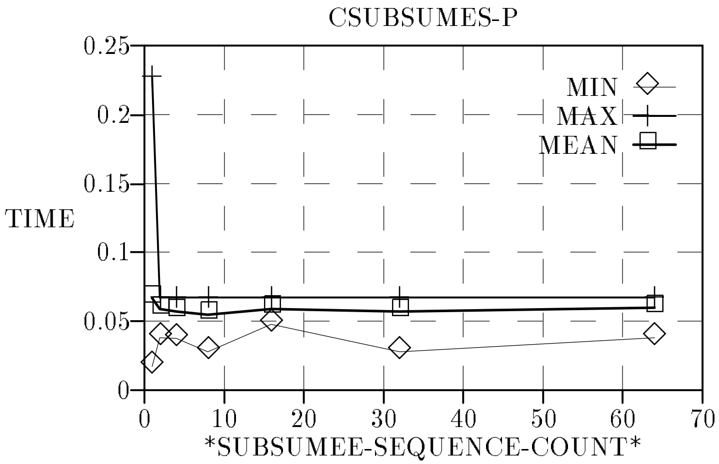
Figure 4.14: Subsumee Distinct Count Experiment B

```
# Fixed knobs:
#
#   *MAX-DEPTH*                 7
#   *BRANCHING-FACTOR*          2
#   *SUBSUMEE-NODE-COUNT*       64
#   *SUBSUMEE-DISTINCT-COUNT*   8
#   *SUBSUMEE-ORDER*            :RANDOM
#   *SUBSUMER-NODE-COUNT*       8
#   *SUBSUMER-DISTINCT-COUNT*   8
#   *SUBSUMER-ORDER*            :RANDOM
#   *REPETITIONS*               20


Varying knob:

(FOR *SUBSUMEE-SEQUENCE-COUNT* = 1 THEN (* 2 *SUBSUMEE-SEQUENCE-COUNT*)
 UNTIL (> *SUBSUMEE-SEQUENCE-COUNT* *SUBSUMEE-NODE-COUNT*))
```





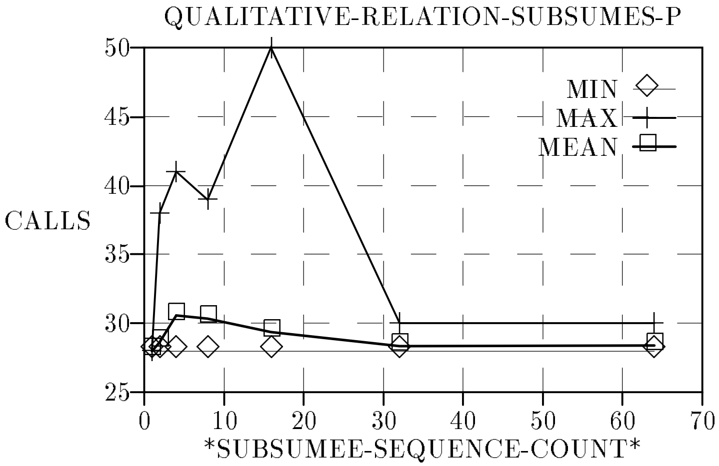
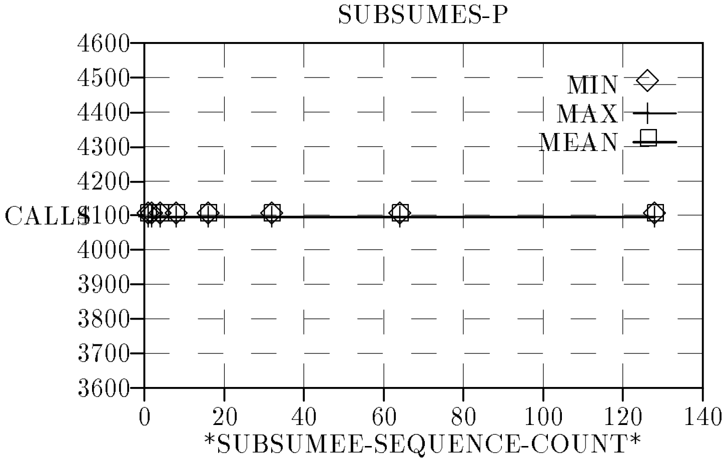


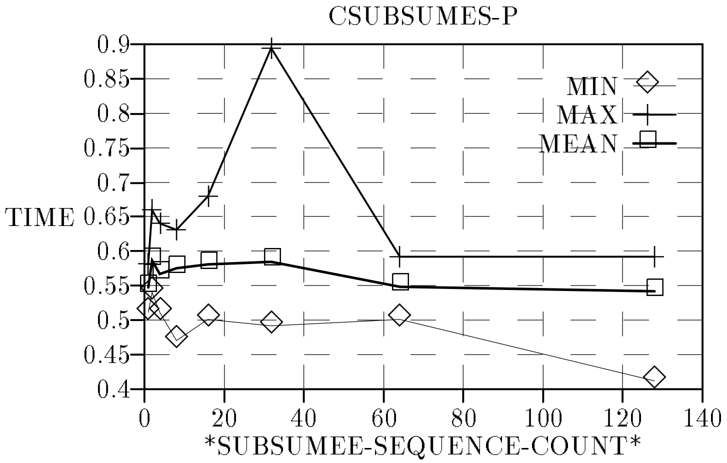
Figure 4.15: Subsumee Sequence Count Experiment A

```
# Fixed knobs:
#
#   *MAX-DEPTH*                 7
#   *BRANCHING-FACTOR*          2
#   *SUBSUMEE-NODE-COUNT*       128
#   *SUBSUMEE-DISTINCT-COUNT*   4
#   *SUBSUMEE-ORDER*            :RANDOM
#   *SUBSUMER-NODE-COUNT*       32
#   *SUBSUMER-DISTINCT-COUNT*   4
#   *SUBSUMER-ORDER*            :RANDOM
#   *REPETITIONS*               20


Varying knob:

(FOR *SUBSUMEE-SEQUENCE-COUNT* = 1 THEN (* 2 *SUBSUMEE-SEQUENCE-COUNT*)
 UNTIL (> *SUBSUMEE-SEQUENCE-COUNT* *SUBSUMEE-NODE-COUNT*))
```





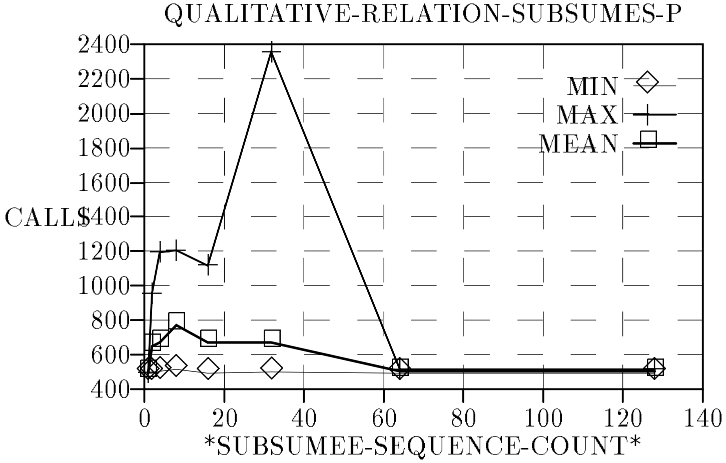
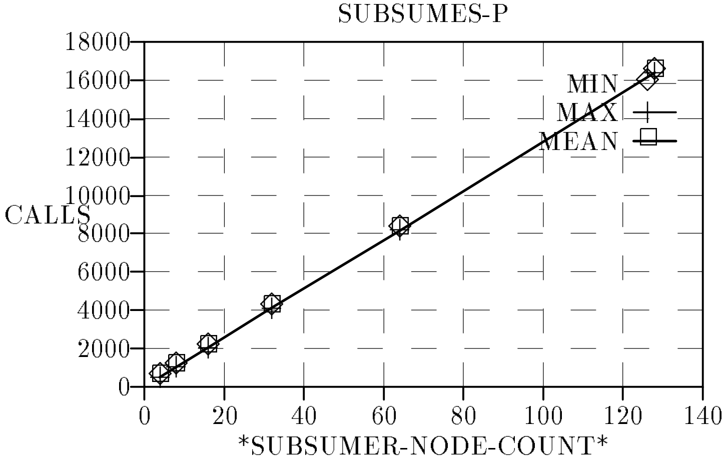


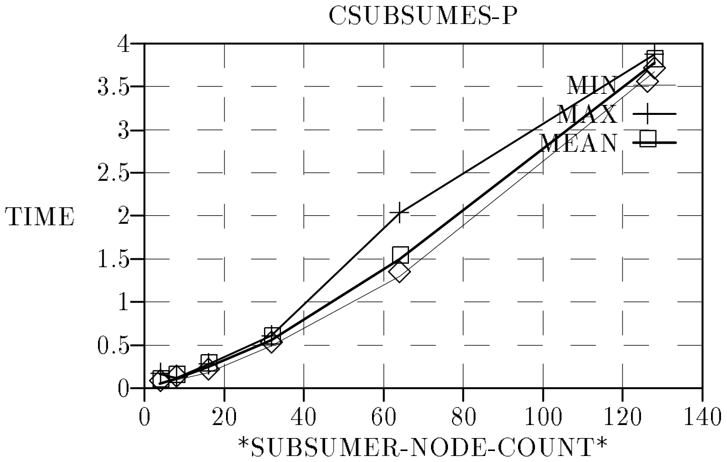
Figure 4.16: Subsumee Sequence Count Experiment B

```
# Fixed knobs:
#
#    *MAX-DEPTH*                  7
#    *BRANCHING-FACTOR*           2
#    *SUBSUMEE-NODE-COUNT*        128
#    *SUBSUMEE-DISTINCT-COUNT*    4
#    *SUBSUMEE-SEQUENCE-COUNT*    2
#    *SUBSUMEE-ORDER*             :RANDOM
#    *SUBSUMER-DISTINCT-COUNT*    4
#    *SUBSUMER-ORDER*             :RANDOM
#    *REPETITIONS*                20


Varying knob:

(FOR *SUBSUMER-NODE-COUNT* = 4 THEN (* 2 *SUBSUMER-NODE-COUNT*) UNTIL
 (> *SUBSUMER-NODE-COUNT* *SUBSUMEE-NODE-COUNT*))
```





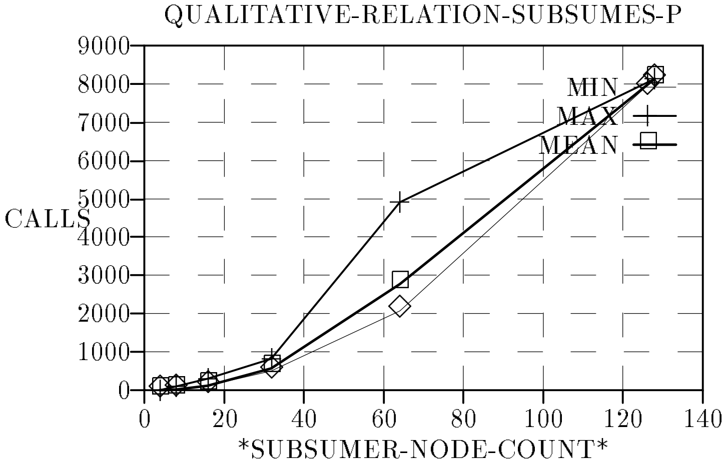
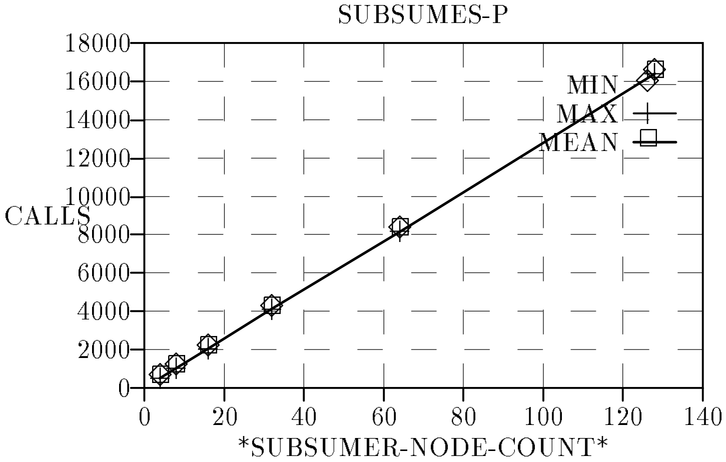


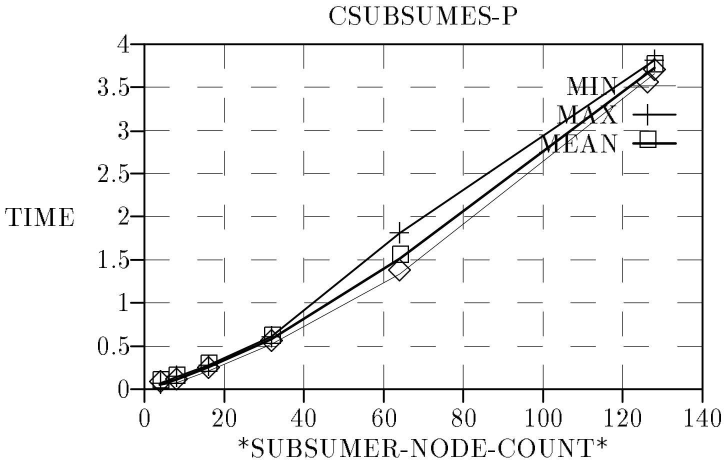
Figure 4.17: Subsumer Node Count Experiment A

```
# Fixed knobs:
#
#   *MAX-DEPTH*                 7
#   *BRANCHING-FACTOR*          2
#   *SUBSUMEE-NODE-COUNT*       128
#   *SUBSUMEE-DISTINCT-COUNT*   16
#   *SUBSUMEE-SEQUENCE-COUNT*   4
#   *SUBSUMEE-ORDER*            :RANDOM
#   *SUBSUMER-DISTINCT-COUNT*   8
#   *SUBSUMER-ORDER*            :RANDOM
#   *REPETITIONS*               20


Varying knob:

(FOR *SUBSUMER-NODE-COUNT* = 4 THEN (* 2 *SUBSUMER-NODE-COUNT*) UNTIL
 (> *SUBSUMER-NODE-COUNT* *SUBSUMEE-NODE-COUNT*))
```





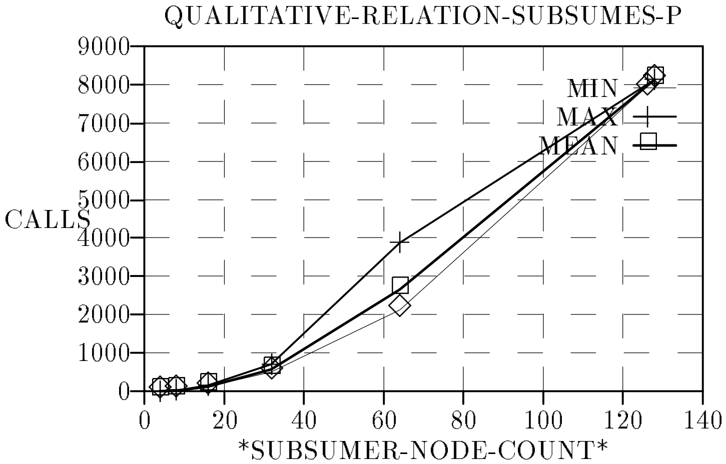
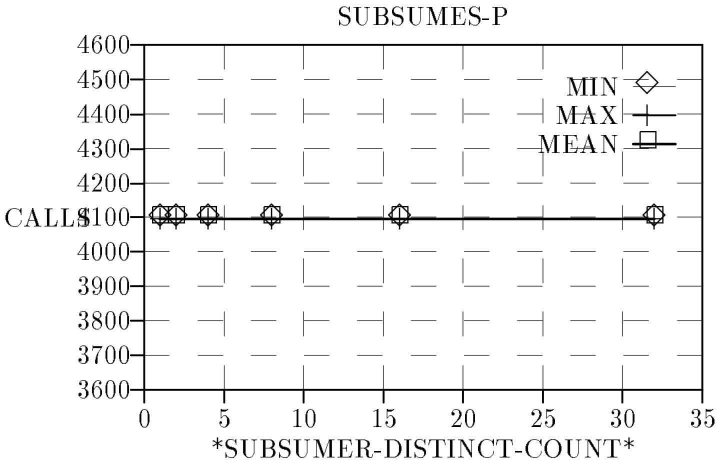


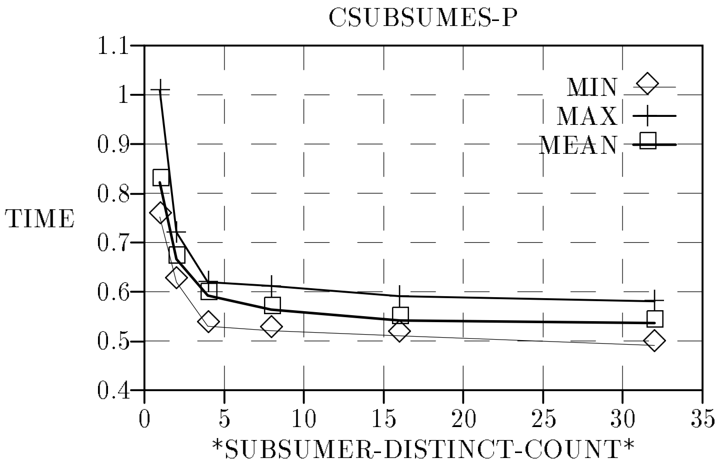
Figure 4.18: Subsumer Node Count Experiment B

```
# Fixed knobs:
#
#    *MAX-DEPTH*                  7
#    *BRANCHING-FACTOR*           2
#    *SUBSUMEE-NODE-COUNT*        128
#    *SUBSUMEE-DISTINCT-COUNT*    32
#    *SUBSUMEE-SEQUENCE-COUNT*    2
#    *SUBSUMEE-ORDER*             :RANDOM
#    *SUBSUMER-NODE-COUNT*        32
#    *SUBSUMER-ORDER*             :RANDOM
#    *REPETITIONS*                20


Varying knob:

(FOR *SUBSUMER-DISTINCT-COUNT* = 1 THEN (* 2 *SUBSUMER-DISTINCT-COUNT*)
 UNTIL (> *SUBSUMER-DISTINCT-COUNT* *SUBSUMEE-DISTINCT-COUNT*))
```





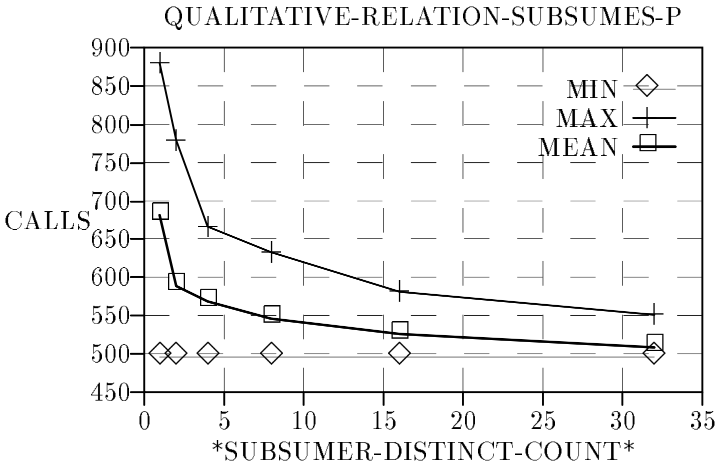
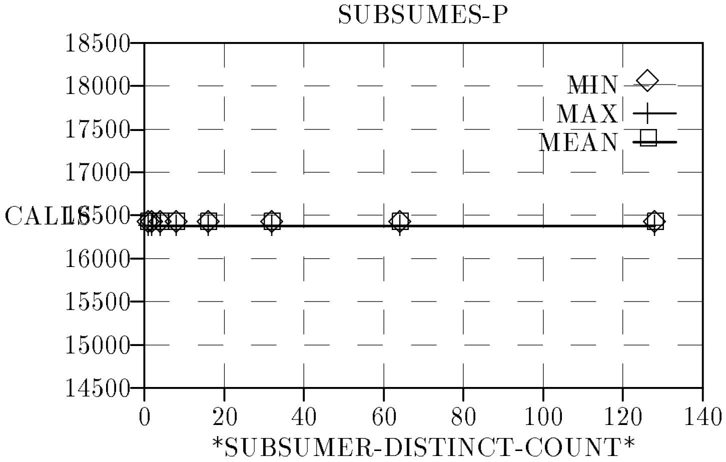


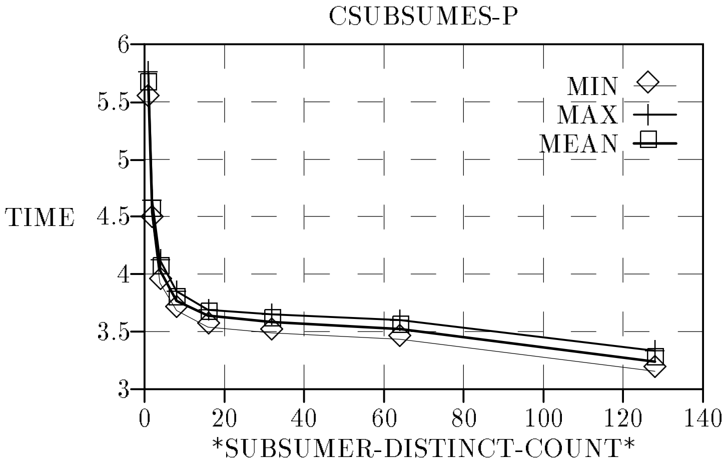
Figure 4.19: Subsumer Distinct Count Experiment A

```
# Fixed knobs:
#
#   *MAX-DEPTH*                 7
#   *BRANCHING-FACTOR*          2
#   *SUBSUMEE-NODE-COUNT*       128
#   *SUBSUMEE-DISTINCT-COUNT*   128
#   *SUBSUMEE-SEQUENCE-COUNT*   2
#   *SUBSUMEE-ORDER*            :RANDOM
#   *SUBSUMER-NODE-COUNT*       128
#   *SUBSUMER-ORDER*            :RANDOM
#   *REPETITIONS*               20


Varying knob:

(FOR *SUBSUMER-DISTINCT-COUNT* = 1 THEN (* 2 *SUBSUMER-DISTINCT-COUNT*)
 UNTIL (> *SUBSUMER-DISTINCT-COUNT* *SUBSUMEE-DISTINCT-COUNT*))
```





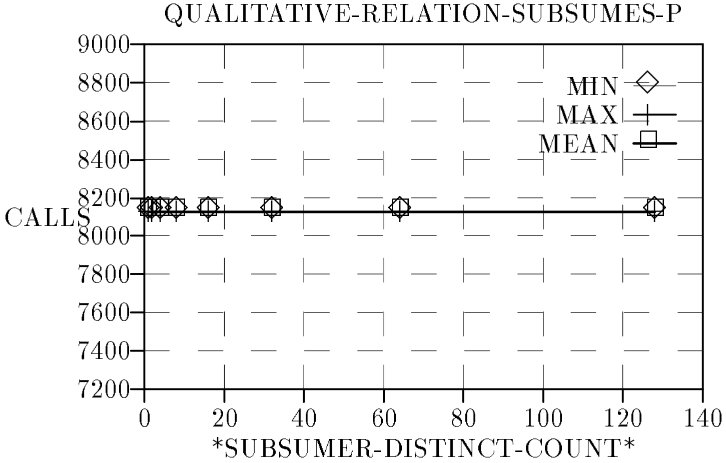


Figure 4.20: Subsumer Distinct Count Experiment B

4.6 Constraint Network Classification

Subsumption allows us to automatically classify a library of constraint network concepts, e.g., plans, into a taxonomy. By analogy to standard description logic taxonomies, the taxonomy derives all the benefits of subsumption-based organization noted in Section 2.1. Projecting into the future, as such libraries grow in size and scope, problems of organization and maintenance will become increasingly critical. Search procedures will be able to utilize the definitional nature of the taxonomy for fast and accurate results. Also, since most present day plan libraries are organized by hand, the clerical demands placed on the plan administrator may become burdensome. Our experience with knowledge engineering shows that when confronted with large quantities of information, the enforced semantics of the subsumption-based approach offers significant advantages [Mays *et al.*, 1991b].

The initial T-REX implementation of classification is entirely unremarkable, so we do not elaborate on it here. However, we will illustrate its results on a sample plan library in the travel domain. Figure 5.5 on page 182 defines a small travel plan library, and Figure 5.6 on page 183 shows the plan taxonomy constructed by T-REX using those plan definitions. The root of the plan taxonomy is the trivial plan, PLAN, which has no actions. Appendix A includes a somewhat larger travel plan library.

So far, this chapter has discussed subsumption and classification of constraint network concepts. Traditional recognition of constraint network individuals is covered next.

4.7 Recognition

This chapter has largely concentrated on representing and reasoning with constraint network concepts, as opposed to individuals. However, we are also concerned with discovering when a constraint network individual *instantiates* a constraint network concept. An example where a plan individual instantiates a plan concept was given in Section 4.3.3. This instance-checking problem, whose analogue in standard description logic is traditionally called *recognition*, is substantially similar to constraint network subsumption. Recognition that network individual I is an instance of network concept C requires a mapping from C to I similar to a subsumption mapping, except that nodes of C must be *instantiated* by their counterparts in I:

Definition 45 *Node* n1 instantiates *node* n2 *iff the individual associated with* n1 *instantiates the concept of* n2*.*

Individual QME constraints are no different from QME constraints at the conceptual level. Hence, QME constraint instantiation is identical to QME constraint subsumption; likewise QME arc instantiation is identical to QME arc subsumption. This immediately leads to a definition of *instantiation mapping*:

Definition 46 *A* structural instantiation mapping *from network concept* C *to network individual* I *maps every node* c *of* C *to a distinct node* i *of* I *such that:*

1. i *instantiates* c
2. *for all arc types T, every arc of type T between a pair of nodes in* C *is instantiated by the corresponding arc of type T in* I.

Existence of a structural instantiation mapping is necessary and sufficient to establish constraint network instantiation, given complete constraint propagation within the networks, so we have:

Theorem 8 *Network individual* I *instantiates network concept* C *iff there exists an instantiation mapping from* C *to* I.

Proof: See Appendix B.

As with standard description logic, this is written $I \epsilon C$. Caveats stated in Section 4.4.3 regarding the incompleteness of constraint propagation in T-REX apply equally here. Recall Definition 44 on page 124, which addressed the fact that a QME network with an equality constraint can be satisfied by a subsumee that lacks a corresponding equality constraint. An essentially similar definition can be given for QME network instantiation. As a special case of instantiation mapping, we have a *bijective instantiation mapping*:

Definition 47 *A* bijective instantiation mapping *is an instantiation mapping such that the nodes of the two networks are in one-to-one correspondence.*

Similar to our use of bijective instantiation in predictive concept recognition, we will make important use of *bijective instantiation mapping* when we consider plan recognition in the next chapter, e.g., to help define concrete plan concepts and finished individual plans. Definition 46 (and the extension for equality constraints just discussed) is essentially similar to subsumption mapping. Consequently, we will not go into greater detail about it. T-REX computes traditional recognition of constraint network individuals in this way. This traditional version of constraint network recognition also plays an important part in the *predictive* recognition methodology for constraint networks proposed in Chapter 5.

4.8 Conclusion

This chapter has shown how to create complex descriptions in the form of constraint networks based on a description logic framework, by associating standard description logic concepts or individuals with the nodes. Arcs express relationships between pairs of nodes that are contingent on their joint appearance in the description. Thus, constraint network concepts denote a class of possible constraint network individuals. For concreteness, we have emphasized QME constraint networks, where the nodes represent (classes of) events, and the arcs represent qualitative temporal, metric temporal, and/or equality relationships among them. In particular, we used QME networks for the bodies of plans, where the events are constrained to be actions. We have implemented the T-REX system to represent and reason with QME descriptions.

Proper comparison between constraint networks may depend upon implicit constraints within the networks. To complete QME networks, i.e., make implicit constraints manifest, we employ several application-independent constraint propagation techniques. Constraint propagation in T-REX takes advantage of pre-existing reasoners to propagate conceptual and temporal constraints. T-REX handles equality constraints itself. In addition, T-REX provides a rule-based reasoner which allows users to implement application-dependent constraints on network descriptions by writing suitable rules. This reasoner can be used to achieve application-dependent constraint propagation.

A hallmark of description logic is taxonomic reasoning based on subsumption. We defined subsumption over constraint network descriptions and showed the problem to be NP-complete. However, noting that QME networks often contain a wealth of information amenable to heuristics, we proposed a detailed subsumption algorithm for QME networks. QME network subsumption is based on standard concept subsumption, together with customized definitions of subsumption for qualitative, metric, and equality constraints. Instantiation is handled similarly. These inferences are also carried out by T-REX. Performance analysis shows that the algorithm performs well in practice on non-pathological networks.

On a different tack, Chapter 5 combines the constraint network representation and reasoning techniques developed in this chapter with the predictive concept recognition methodology of Chapter 3 to conduct predictive recognition of constraint network concepts.

Chapter 5

Predictive Recognition of Constraint Network Concepts

5.1 Introduction

Chapter 3 presented a predictive concept recognition methodology for standard description logic. Chapter 4 developed a methodology for subsumption-based reasoning with constraint networks. This chapter explores a synthesis of the two methodologies, which yields a predictive recognition methodology for constraint network concepts [Weida, 1995b].

In Chapter 3, a closed terminology assumption during problem solving permitted us to track the consistency of concepts in a terminology with incrementally specified individuals. A similar completeness assumption over a taxonomy of constraint network concepts enables us to track the status of constraint network individuals *vis à vis* a constraint network library. To make the material on constraint networks in Chapter 4 concrete, we studied QME constraint networks and applied them to the representation of plans. Following this precedent, we will now focus on predictive recognition of QME network concepts, particularly for the task of deductive plan recognition.

After briefly characterizing our plan recognition task in Section 5.2, we proceed in Section 5.3 to describe incremental instantiation of an *observation network*, i.e., a distinguished QME network individual which holds the observations from which we attempt to recognize plans. Section 5.4 covers our assumptions and goals. Section 5.5 goes on to adapt the notion of CTA-consistency from Chapter 3 to the realm of constraint network concepts and individuals. Here, constraint network consistency builds on concept consistency just as constraint network subsumption built on concept subsumption in the last chapter. Constraint network consistency also resembles constraint network subsumption because it entails searching for a suitable mapping from one network to another. At the same time, the requirement for search distinguishes it from standard concept consistency, where roles of concepts were matched directly by name. The preceding ideas come together in Section 5.6, where we detail the consistency inferences involving individuals which underly predictive constraint network recognition in the case of monotonic observation. Having thus laid the groundwork for plan recognition, in Section 5.7 we propose plan library augmentation along the general lines of terminology augmentation in Section 3.6. Our overall recognition methodology is introduced in terms of monotonic observation in Section 5.8. The more restricted case of *perfect observation* is addressed in Section 5.9, and the more general case of *unrestricted observation* is examined in Section 5.10. (T-REX is capable of operating in all three modes.) Subsequent material covers recognition of simultaneous plans, in Section 5.11, and active response during recognition, in Section 5.12. A comparison with predictive recognition in standard description logic is made in Section 5.13. Section 5.14 sums up this chapter.

5.2 Plan Recognition

Following [Kautz, 1991b], we will concentrate on plan recognition via plan bodies, where the bodies describe temporal patterns of actions. In our *plan recognition problem*, we accept incremental *observations* about individual actions and their relation to one another. An observation represents a determination that one or more actions have occurred and/or that one or more constraints hold between action occurrences. Given certain assumptions, we then deduce one or more plans that "explain" the observations in terms of known plans by partitioning the plan taxonomy into necessary, optional, and impossible plans.

The next section delves into the representation and incremental revision of observations during plan recognition.

5.3 Incremental Instantiation

During plan recognition, observations are recorded in a distinguished constraint network individual called the *observation network*. Observations may be supplied by a user and/or an application program. We expect that the observations reflect purposeful behavior and thus represent one or more *individual plans* composed of individual actions. An example is OBS100, which was shown in Figure 4.4 on page 108 and discussed in Section 4.3.3. In general, the observation network may be an inexact or incomplete model of actual events. As events unfold and observations are made, the observation network is updated, yielding successive versions. A monotonic update may entail *extension* and/or *refinement*. Extensions add new nodes and/or arcs, while refinements further constrain (specialize) existing ones. More generally, observations can be retracted or generalized. We will introduce updates to individual plan networks, such as observation networks, with the *redefine-individual-plan* operator. Figure 5.1 on page 161 shows a revision of OBS100 after monotonic update, preceded by a description of VISIT-CITY97, the individual action which has been added to it. We will further refine OBS100 later in this section.

We can recognize that the observations instantiate a particular plan based on the existence of an instantiation mapping (see Section 4.7) from the plan to the observations. When the mapping is bijective, we will say that the observations

```
(create-individual VISIT-CITY97
    (and VISIT-CITY
        (exactly 1 agent)
        (fills agent A-RANDOM-RESEARCHER)
        (exactly 1 location)
        (fills location METROPOLIS)))

(redefine-individual-plan OBS100
    :steps ((step1 ATTEND-WORKSHOP98)
            (step2 ATTEND-AI-CONFERENCE99)
            (step3 VISIT-CITY97))
    :metric-constraints ((0 < step2_start − step1_finish ≤ ∞)
                         (360 ≤ step1_finish − step1_start ≤ 360)))
```

Figure 5.1: Revised Observations

bijectively instantiate the plan (recall the definition of bijective instantiation mapping on page 156). This special case of instantiation will prove useful in identifying *concrete* plans. Just as we differentiated between concrete and abstract concepts in Chapter 3, we will now differentiate between concrete and abstract plans. Some constituents of the plan library describe a ground level course of action that can be carried out *per se*, presumably to achieve some goal. We will call these plans concrete:

Definition 48 *Plan* P *is* concrete *iff an individual plan which bijectively instantiates* P *might be both complete and sufficiently specific for the purposes of the intended application.*

Identifying a set of concrete plans for a particular application is a knowledge engineering choice, as is the level of detail at which they are described. As an example, a package tour operator may offer a number of concrete travel plans such as a particular ski tour to Chamonix, which we will refer to as SKI-CHAMONIX. However, the plan library administrator may also wish to introduce more general descriptions that should not be recognized as concrete plans *per se*, such as the class of ski tours to Europe. We will call these *abstract* plans:

Definition 49 *A plan is* abstract *iff it is not concrete.*

Abstract plans serve several purposes stemming from their ability to capture the commonality among a number of concrete plans. Abstract plans can be used to index more specific plans for plan retrieval or for browsing the plan library. They can provide notational convenience by serving as macro components of several more specific plans. They can serve as the basis for triggering some functionality whenever the abstract class of plans is recognized. Finally, they might prove useful for inheritance, although this last point is not developed in the present thesis.

Criteria for plan libraries that are well-formed with respect to the abstract / concrete distinction correspond to the criteria of Section 3.2 for concept terminologies. Ultimately, of course, the division between abstract and concrete plans is application-dependent and user-defined. We model the abstract / concrete distinction as a boolean status associated with each plan when it is defined. As an example, a plan library for a package tour operator may include abstract plans like EUROPEAN-SKI-VACATION, of which SKI-CHAMONIX is just one concrete example. More generally, we can compute the possibility and necessity of arbitrary patterns of events (other than plans, but presumably still meaningful) by classifying them in the plan taxonomy and treating them as abstract.

An important task is to determine when the observation network may reflect a *finished* plan. The entire set of concrete plans is instrumental in making this determination:

Definition 50 *Constraint network individual* I *is* finished *when*

1. I *bijectively instantiates a concrete constraint network concept*
2. *The individual associated with every node* i *of* I *is finished according to Definition 9*
3. *The constraints associated with each arc of* I *are exact.*

For QME networks, an exact qualitative constraint is non-disjunctive, and an exact metric constraint is (a numeric interval that degenerates to) a single number. An equality constraint is intrinsically exact, independent of any inexactness in the descriptions of its operands.[1] Definition 50 is rather strong, e.g., in some applications it might be rather difficult to observe temporal constraints exactly, and it may not be significant to do so. For some applications, e.g., the travel planning application exemplified in Section 5.6 and Appendix A, we could just as well dispense with the second and third clause of Definition 50. However, the first clause is crucial to the notion of a closed library assumption, which will be defined precisely in the next section.

Turning to an example, here is a second revision of OBS100:

(redefine-individual-plan OBS100
:steps ((step1 ATTEND-WORKSHOP98)
(step2 ATTEND-AI-CONFERENCE99)
(step3 VISIT-CITY97))
:metric-constraints (($1200 \leq \text{step1}_{start} - \text{step3}_{start} \leq 1200$)
($360 \leq \text{step1}_{finish} - \text{step1}_{start} \leq 360$)
($1440 \leq \text{step2}_{start} - \text{step1}_{finish} \leq 1440$)
($4320 \leq \text{step2}_{finish} - \text{step2}_{start} \leq 4320$)
($1200 \leq \text{step3}_{finish} - \text{step2}_{finish} \leq 1200$)))

[1]While the operands of an equality constraint can be modified, the equality constraint itself can not.

Through constraint propagation, T-REX infers several qualitative constraints on OBS100:

step1/ATTEND-WORKSHOP98	before	step2/ATTEND-AI-CONFERENCE99
	during	step3/VISIT-CITY97
step2/ATTEND-AI-CONFERENCE99	after	step1/ATTEND-WORKSHOP98
	during	step3/VISIT-CITY97
step3/VISIT-CITY97	contains	step1/ATTEND-WORKSHOP98
	contains	step2/ATTEND-AI-CONFERENCE99

Note that the metric constraints in this revision of OBS100 are all exact, e.g., the duration of *step1* is exactly 360 minutes. Now consider the plan library defined in Figure 1.1 on page 9 and diagrammed in Figure 4.3 on page 106. The concept taxonomy is not shown because relationships among concepts are quite intuitive based on their names. Assuming that PLAN-B is concrete and that the individual actions associated with the steps of OBS100 are all finished, the preceding revision of OBS100 is a finished instantiation of PLAN-B. A caveat: the T-REX implementation does not yet determine whether a plan is finished.

This section has characterized incremental instantiation of observation networks, which serve as input to the plan recognition process. The following section states our assumptions and goals for predictive plan recognition.

5.4 Assumptions and Goals

Initially, our plan recognition assumptions will correspond to those we made about concept recognition in Chapter 3. They are:

1. A closed library assumption
2. A monotonic update assumption

The second of these assumptions will be relaxed later on.

Our *closed library assumption* is defined as follows:

Definition 51 *Under the* closed library assumption, *it is assumed that no plans will be added to the library during problem solving*[2], *and that every individual plan will ultimately be finished according to Definition 50.*

This assumption implies that every finished individual plan bijectively instantiates an explicitly defined concrete plan concept, and every unfinished individual plan can be monotonically updated to do so.[3]

When the observation network is no longer subject to update, i.e., it fully reflects a plan that has been carried out, the CTA ensures that it will bijectively instantiate at least one concrete plan. Such plans are referred to as *ultimate plans*[4]:

Definition 52 *Given that an observation network, when finished, will bijectively instantiate one or more concrete plans, those plans are its* ultimate plans.

While an ultimate plan restricts every node and arc of an individual plan, it may do so at an abstract level, in the sense that PLAN-C of Figure 1.1 requires a VISIT-NATIONAL-PARK action which can be instantiated in many ways, including latitude as to the national park in question.

We also make a *monotonic update assumption* regarding the observation network. This assumption licenses conclusions based on the current state of the observation network. Monotonic update of an observation network may involve adding nodes and/or arcs. It may also involve further restricting existing nodes and/or arcs. Monotonic update of an existing node consists of refining the description of the associated individual as discussed in Section 3.4. Monotonic update of

[2] Although no plans may be added by the *application*, in Section 5.7 we will relax this assumption so the system can add plans strictly for its own internal use to improve efficiency.

[3] This definition does not imply that the updates *will* be monotonic, however.

[4] More generally, *ultimate network concepts.*

an existing arc results in a constraint that is properly subsumed by its predecessor. For QME networks, the possibilities are as follows:

1. A qualitative arc can be updated monotonically by eliminating some of its disjuncts. For example, *before* $\vee$ *after* can be monotonically updated either to *before* or to *after*.

2. Monotonic update of a metric arc results in numeric interval that is properly contained within the previous one, e.g., *start [3,5) finish* can become *start (3,4] finish*.

3. Equality arcs cannot be updated. Note that equality arc subsumption degenerates to equality arc identity.

Provided that an observation network is updated monotonically, the CTA guarantees that it will be continuously consistent with its ultimate plan(s). Then, the incremental recognition process can be seen as continually narrowing down the set of plans that may turn out to be ultimate plans. Note, however, that Section 5.10 will contemplate the case of nonmonotonic updates.

Our principal goal for plan recognition is to track the status of every plan *vis à vis* the observations. As with concept recognition, a plan is either necessary, optional, or impossible:

Definition 53 *Plan* P *is* necessary *with respect to observation network* O *iff* O *instantiates* P.

Definition 54 *Plan* P *is* optional *with respect to observation network* O *iff* O *does not instantiate* P *but can be monotonically updated to do so.*

Definition 55 *Plan* P *is* impossible *with respect to observation network* O *iff* O *neither instantiates* P *nor can be monotonically updated to do so.*

A secondary goal is the ability to respond to changes in the status of plans as required by a particular plan recognition application.

Having clarified our assumptions and goals, we next present consistency inferences for constraint networks which allow us to distinguish between optional and impossible plans under the CTA.

5.5 Constraint Network Consistency

This section covers consistency inferences for constraint networks in general and applies them to plan networks in particular. As with regular description logic concepts, both direct and indirect cases of consistency are treated. Significant differences are also addressed.

5.5.1 Introduction

It is possible to extend the notion of CTA-consistency from standard concepts to constraint networks whose nodes are represented by standard concepts. In the course of this discussion, we will assume that the constraint networks under consideration have already been completed using the constraint propagation and domain-specific inference techniques of Section 4.3.4 through Section 4.3.6. Section 4.4.3 pointed out that the current implementation of constraint propagation in T-REX is not entirely complete; the caveats stated there apply here as well. With that proviso, we can give a general definition of CTA-consistency for constraint network concepts:

Definition 56 *One network concept is* CTA-consistent *with another iff it is possible for an individual to instantiate them simultaneously under CTA.*

The T-REX system specializes this idea to compute plan (body) consistency. The body of plan *P1* is *CTA-consistent* with that of *P2* iff the set of possible action patterns described by *P1* intersects the set described by *P2*. Thus, plans can be compared for consistency relative to a closed plan library.

Nodes and arcs are the structural components of a constraint network. After examining node consistency in Section 5.5.2 and arc consistency in Section 5.5.3, we will come to *structural* constraint network consistency in Section 5.5.4.

5.5.2 Node Consistency

Consistency between nodes follows directly from their associated descriptions. Section 3.5 defined CTA-consistency between a pair of concepts in Definition 21, and between an individual and a concept in Definition 25. A definition for consistency of individuals will prove convenient. Given the unique name assumption, it is trivial:

Definition 57 *An individual is* consistent *with itself and no other individual.*

This permits a general-purpose definition of *node consistency* under CTA[5]:

Definition 58 *Node* n1 *is* CTA-consistent *with node* n2 *iff the concept or individual associated with* n1 *is CTA-consistent with the concept or individual associated with* n2*.*

To handle constraint network consistency, we will have to address arc consistency as well.

5.5.3 Arc Consistency

As with arc subsumption, arc consistency will be defined in terms of the associated constraints. Again, it might be appropriate for some applications to model

[5]When we consider QME network consistency involving equality constraints, Definitions 57 and 58 will allow uniform treatment of equality constraints in both the concept-to-concept and individual-to-concept network cases.

arc semantics with standard concepts, in which case Definition 58 would apply to arcs as well as nodes. For QME networks, however, we will give special definitions to handle consistency of QME arcs based on their special-purpose representations.

First, consistency between qualitative constraints requires at least one common disjunct:

Definition 59 *Qualitative constraints are* consistent *iff the intersection of their disjuncts is non-empty.*

For example, the qualitative constraints consistent with *before* $\vee$ *after* include its subsumees (*before*, *after*, and *before* $\vee$ *after*) along with *before* $\vee$ *meets*, *during* $\vee$ *after*, and numerous others. Counterexamples include *during*, and many others.

Second, metric constraint consistency requires two things: intersection between the numeric intervals they embody, together with reference to the same interval extrema in the same order:

Definition 60 *Metric constraint* $m1\ Q1\ i1_F - j1_G\ R1\ n1$ *(referred to as* M1*) is* consistent *with metric constraint* $m2\ Q2\ i2_F - j2_G\ R2\ n2$ *(referred to as* M2*) iff:*

1. M1 *subsumes* M2*, or*

2. M2 *subsumes* M1*, or*

3. $(m1 < m2)\ \wedge\ ((m2 < n1)\ \vee\ ((m2 = n1)\ \wedge\ (Q2 = R1 =' \leq')))$*, or*

4. $(m2 < m1)\ \wedge\ ((m1 < n2)\ \vee\ ((m1 = n2)\ \wedge\ (Q1 = R2 =' \leq')))$*.*

Notice that the presence of F and G in both *M1* and *M2* enforces the requirement that they reference the same interval extrema. For example, consider this metric constraint:

$$4 \leq interval1_{start} - interval2_{finish} \leq 7 \tag{5.1}$$

It denotes a time period of magnitude [4,7] between the start of one interval, *interval1*, and the end of another, *interval2*. The following metric constraint denotes a time period of magnitude (5,9] between the start of *interval3* and the end of *interval4*:

$$5 < interval3_{start} - interval4_{finish} \leq 9 \tag{5.2}$$

This constraint has no subsumption relation with the preceding one, but they are nonetheless mutually consistent, according to clause 3 of Definition 60. If either constraint is specialized to have a magnitude within (5,7], it will be subsumed by the other. An example of a metric constraint that is consistent with metric constraint 5.1 according to clause 4 of Definition 60 is:

$$1 < interval3_{start} - interval4_{finish} \leq 4 \tag{5.3}$$

On the other hand, metric constraint 5.1 is inconsistent with each of the following metric constraints, among others:

$$2 < interval3_{start} - interval4_{finish} < 3 \tag{5.4}$$

$$11 < interval3_{start} - interval4_{finish} < 12 \tag{5.5}$$

$$5 < interval3_{finish} - interval4_{start} < 7 \tag{5.6}$$

Metric constraints 5.4 and 5.5 have numeric intervals outside the numeric interval of metric constraint 5.1, and metric constraint 5.6 does not refer to the extrema of its intervals in the same order as does metric constraint 5.6.

Third, equality constraint consistency is somewhat different in nature, because equality constraints can not be specialized. Subsumption implies consistency in general, so two equality constraints in a subsumption relationship are also consistent (recall Definition 41 on page 120 and the subsequent discussion):

Definition 61 *Binary equality constraint* $\mathrm{R}_a = \mathrm{S}_b$ is consistent with $\mathrm{R}_c = \mathrm{S}_d$.

As a result, for one network to be consistent with respect to an equality arc in another network, it is sufficient for the first network to have a corresponding equality arc. However, it is not necessary, as we will explain in the next section.

In general, consistency between a pair of arcs follows from consistency between the constraints that label them:

Definition 62 *A pair of arcs are* consistent *iff the associated constraints are consistent.*

Having defined node and arc consistency, they can be employed in the service of network consistency.

5.5.4 Direct Network Consistency

Structural consistency between constraint networks depends on the existence of a suitable correspondence between their constituents. By using node and arc consistency, we can define a *structural CTA-consistency mapping* between a pair of completed constraint network concepts:

Definition 63 *A* structural CTA-consistency mapping *from network concept* N1 *to network concept* N2 *maps every node* n1 *of* N1 *to a distinct node* n2 *of* N2 *such that:*

1. n1 *is CTA-consistent with* n2
2. *for all arc types T, every arc of type T between a pair of nodes in* N1 *is CTA-consistent with the corresponding arc of type T in* N2*.*

A consistency mapping from *N1* to *N2* is written $N1 \mapsto N2$.

Figure 5.2 on page 172 demonstrates a structural consistency mapping between two plan networks, where dashed arrows map from nodes in PLAN-P to nodes in

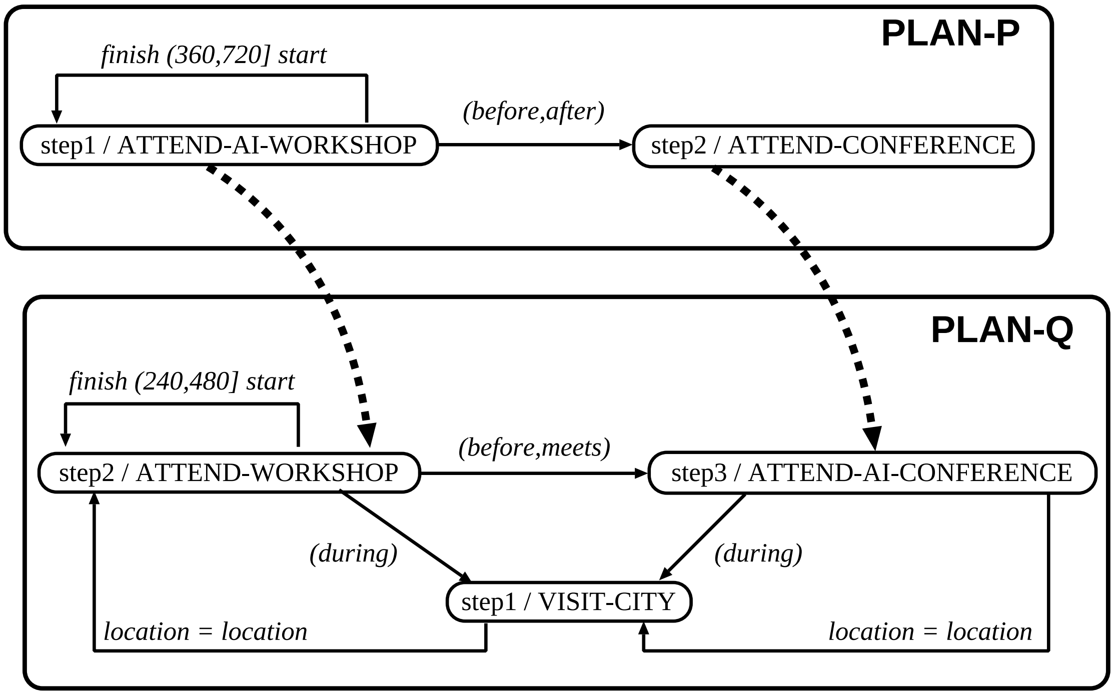


Figure 5.2: Structural Consistency Mapping

PLAN-Q. The intervening arcs are implicitly mapped accordingly.[6] This example illustrates a variety of consistency relationships. Notice that in one case, the node in PLAN-P is more specific than its counterpart in PLAN-Q, i.e., ATTEND-AI-WORKSHOP is subsumed by ATTEND-WORKSHOP. In the other case it is more general, i.e., ATTEND-CONFERENCE subsumes ATTEND-AI-CONFERENCE. The qualitative and metric arcs in PLAN-P neither subsume, nor are subsumed by, their counterparts in PLAN-Q. Thus, the two plans differ in the number and specificity of their actions, as well as the variety of their constraints. Although neither plan subsumes the other, it can be seen that a bijective instantiation of PLAN-Q can also instantiate PLAN-P. For example, consider a person who attends an AI workshop for 420 minutes, then attends an AI conference, all while visiting a single city. This notion of structural consistency is analogous to the case of standard description logic, where concepts are consistent by virtue of consistent roles and primitives.

[6]To avoid clutter, we omit arcs that would be added within each network by completion inferences.

Consistency mapping between QM networks is purely structural. As we remarked earlier, the situation changes for QME networks because of equality arcs. The presence of an equality arc in one network is not necessary to establish consistency with a corresponding equality arc in another network. Rather, consistency with respect to an equality arc is established by consistent roles corresponding to its operands. Let us revisit a simple example of a network with an equality constraint from Section 4.3.1:

PLAN1: step0/THESIS-STUDY   *advisor* = *chair*   step2/THESIS-DEFENSE

Let us also define two concepts that are subsumees of THESIS-STUDY and THESIS-DEFENSE, respectively:

```
(define-concept CS-THESIS-STUDY
  (and THESIS-STUDY
       (the agent CS-STUDENT)
       (the advisor CS-PROFESSOR)))

(define-concept CS-THESIS-DEFENSE
  (and THESIS-DEFENSE
       (the agent CS-STUDENT)
       (the chair CS-PROFESSOR)))
```

Now consider PLAN2, another network with two nodes, but no equality constraint between them:

PLAN2: step10/CS-THESIS-STUDY   (*before*)   step12/CS-THESIS-DEFENSE

PLAN2 is consistent with PLAN1, even though it has no equality constraint corresponding to the one in PLAN1. To see why, notice that CS-THESIS-STUDY and

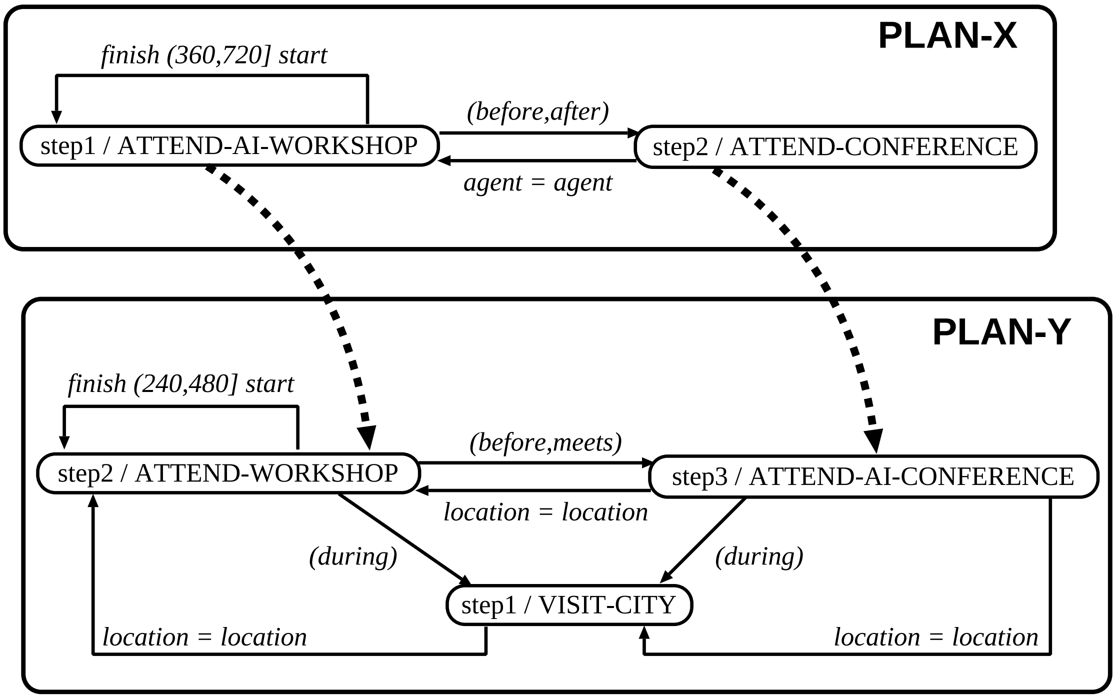


Figure 5.3: Non-Structural Consistency Mapping

CS-THESIS-DEFENSE allow for the possibility that the advisor of *step10* and the chair of *step12* are the same CS-PROFESSOR. Therefore, PLAN2 without any equality constraint is consistent with PLAN1. Of course, PLAN2 would also be subsumed by PLAN1 if it had an *advisor* = *chair* arc from *step10* to *step12*.

In general, T-REX may need to check the consistency of equality constraints in each network against role restrictions in the other. The need for bidirectional checking makes equality constraints different in kind from the other constraints. For example, consider the adaptation of Figure 5.2 shown in Figure 5.3, where PLAN-X has all the constraints of PLAN-P and PLAN-Y has all the constraints of PLAN-Q. PLAN-X additionally requires equality between the agents of its actions, but says nothing about their location. The converse is true in PLAN-Y. Nonetheless, it can be seen that a bijective instantiation of PLAN-Y can also instantiate PLAN-X. For example, again consider a person who attends an AI workshop for 420 minutes, then attends an AI conference, all while visiting a single city.

We take these factors into account with a definition of CTA-consistency mapping that is specialized for QME network concepts:

Definition 64 *A* CTA-consistency mapping *from QME network concept* N1 *to QME network concept* N2 *maps every node* n1 *of* N1 *to a distinct node* n2 *of* N2 *such that:*

1. n1 *is CTA-consistent with* n2
2. *every temporal (qualitative or metric) arc between a pair of nodes in* N1 *is consistent with the corresponding temporal arc in* N2
3. *for every ordered pair of nodes* m1 *and* n1 *of* N1 *mapped to* m2 *and* n2 *of* N2*, respectively,*
   (a) *for every equality arc* R = S *from* m1 *to* n1*, if roles* $\mathrm{R_{m2}}$ *and* $\mathrm{S_{n2}}$ *both exist*[7]*, they are CTA-restriction-consistent.*
   (b) *for every equality arc* R = S *from* m2 *to* n2*, if role* $\mathrm{R_{m1}}$ *and* $\mathrm{S_{n1}}$ *both exist, they are CTA-restriction-consistent.*

Such a CTA-consistency inference is written *N1* $\mapsto$ *N2*. As discussed informally earlier, it can be seen that PLAN-X $\mapsto$ PLAN-Y as in Figure 5.3.

A direct CTA-consistency inference between two network concepts can be made in either direction:

Definition 65 *QME network concepts* N1 *and* N2 *are* directly CTA-consistent *iff* N1 $\mapsto$ N2 *or* N2 $\mapsto$ N1*.*

Note that there may be multiple CTA-consistency mappings from one network to another. This will be significant for the plan library augmentation process to be described in Section 5.7.

[7]Although *m1* and *m2* are CTA-consistent, it can be seen from Definitions 16 through 21 that they need not restrict the same set of roles (similarly for *n1* and *n2*).

The algorithm that T-REX implements to decide direct constraint network consistency is substantially similar to the subsumption algorithm presented in Section 4.4.5. Therefore, we will not go into further detail about the network consistency algorithm. To supplement the direct consistency inferences described in this section, the next section shows how to make indirect consistency inferences between constraint network concepts, e.g., plans.

5.5.5 Indirect Consistency in a Closed Library

There is also an indirect case of CTA-consistency between two constraint network concepts. It can arise when each network has a node or arc for which the other lacks a suitable counterpart. Consider the simple plans in Figure 5.4 on page 177, and assume that the actions VISIT-ARGENTINA, VISIT-BRAZIL, and VISIT-CHILE are mutually disjoint by virtue of their locations. Both AB-TOUR and BC-TOUR subsume ABC-TOUR. They are not directly consistent, because AB-TOUR lacks an action consistent with VISIT-CHILE and BC-TOUR lacks an action consistent with VISIT-ARGENTINA. However, their consistency may still be established by the presence of a third user-defined plan. Notice that AB-TOUR and BC-TOUR can be simultaneously instantiated by a bijective instantiation of their common subsumee, ABC-TOUR, e.g.:

(create-individual-plan ABC-TOUR007
  :steps ((step1 VISIT-ARGENTINA1)
    (step2 VISIT-BRAZIL2)
    (step3 VISIT-CHILE3))
  :metric-constraints ((250 $\leq$ step2$_{finish}$ – step2$_{start}$ $\leq$ 250)
    (0 $\leq$ step2$_{start}$ – step1$_{finish}$ $\leq$ 0)
    (120 $\leq$ step3$_{start}$ – step2$_{finish}$ $\leq$ 120)))

The fact that ABC-TOUR is a common subsumee, is not essential. For example, *In AB-TOUR, a visit to Argentina precedes (perhaps immediately) a visit to Brazil of 100 to 300 time units.*

(define-plan AB-TOUR
  :steps ((step-a VISIT-ARGENTINA)
          (step-b VISIT-BRAZIL))
  :qualitative-constraints ((step-a ($before, meets$) step-b))
  :metric-constraints ((100 $\leq$ step-b$_{finish}$ – step-b$_{start}$ $\leq$ 300)))

In BC-TOUR, a visit to Brazil of 200 to 400 time units precedes by some time a visit to Chile.

(define-plan BC-TOUR
  :steps ((step-b VISIT-BRAZIL)
          (step-c VISIT-CHILE))
  :qualitative-constraints ((step-b (*before*) step-c))
  :metric-constraints ((200 $\leq$ step-b$_{finish}$ – step-b$_{start}$ $\leq$ 400)))

In ABC-TOUR, a visit to Argentina immediately precedes a visit to Brazil of 200 to 400 time units precedes, followed after some time by a visit to Chile.

(define-plan ABC-TOUR
  :steps ((step-a VISIT-ARGENTINA)
          (step-b VISIT-BRAZIL)
          (step-c VISIT-CHILE))
  :qualitative-constraints ((step-a (*meets*) step-b)
                            (step-b (*before*) step-c))
  :metric-constraints ((200 $\leq$ step-b$_{finish}$ – step-b$_{start}$ $\leq$ 300)))

Figure 5.4: Indirectly CTA-consistent Plans

suppose we relaxed the stated metric constraint in ABC-TOUR so that neither AB-TOUR nor BC-TOUR subsumes it, as follows:

$$(0 < \text{step-b}_{finish} - \text{step-b}_{start} \leq 500)$$

Nonetheless, the new definition of ABC-TOUR is more general than (subsumes) the previous one, so it still establishes the consistency of AB-TOUR and BC-TOUR. Taking this sort of case into account, we have:

Definition 66 *QME network concepts* N1 *and* N2 *are indirectly CTA-consistent iff there exists an explicitly defined network concept* N3 *such that*

1. N1 $\mapsto$ N3
2. N2 $\mapsto$ N3
3. *For every node* n3 *of* N3 *mapped from node* n1 *of* N1 *and also mapped from node* n2 *of* N2, n1 *and* n2 *are CTA-consistent*
4. *For every ordered pair of nodes* n3a *and* n3b *of* N3 *mapped from nodes* n1a *and* n1b *of* N1 *respectively and also mapped from nodes* n2a *and* n2b *of* N2 *respectively*
   (a) *every temporal (qualitative or metric) arc from* n1a *to* n1b *is consistent with the corresponding temporal arc from* n2a *to* n2b
   (b) *for every equality arc* R = S *from* n1a *to* n1b, *if role* $\mathrm{R_{n2a}}$ *and* $\mathrm{S_{n2b}}$ *both exist, they are CTA-restriction-consistent*
   (c) *for every equality arc* R = S *from* n2a *to* n2b, *if role* $\mathrm{R_{n1a}}$ *and* $\mathrm{S_{n1b}}$ *both exist, they are CTA-restriction-consistent*

The first two clauses of this definition ensure that networks *N1* and *N2* are consistent with *N3* by themselves. The last two clauses ensure that wherever the direct consistency mappings from *N1* and *N2* interact in *N3*, they do so in a mutually consistent way. For example, consider Figure 5.4 and let AB-TOUR, BC-TOUR, and ABC-TOUR play the parts of *N1*, *N2*, and *N3*, respectively. To detail the inference, we will simply refer to nodes by the corresponding steps, and take advantage of the fact that in this example, steps which are mapped together always happen to have the same label:

1. By mapping *step-a* and *step-b* of AB-TOUR to the identically named steps of ABC-TOUR, it is easily seen that AB-TOUR $\mapsto$ ABC-TOUR as in Definition 64.

2. By mapping *step-b* and *step-c* of BC-TOUR to the identically named steps of ABC-TOUR, it is likewise clear that BC-TOUR $\mapsto$ ABC-TOUR.

3. *step-b* of ABC-TOUR is mapped from the identically named steps of both AB-TOUR and BC-TOUR. Since the same action, VISIT-BRAZIL, is associated with each of them, their CTA-consistency is obvious.

4. For each pair of steps in ABC-TOUR, their counterparts in AB-TOUR and BC-TOUR must be checked against each other. We will just detail this for the explicit constraints given in the plan definitions; it is not hard to verify that the implicit constraints work out too. Noting that a step can be paired with itself, only *step-b* of ABC-TOUR is mapped from both AB-TOUR and BC-TOUR. Only the durations are at issue; together *step-b* of AB-TOUR and *step-b* of BC-TOUR admit a range of durations from 200 to 300, inclusive.

Putting the direct and indirect cases together, CTA-consistency identifies pairs of network concepts that can be instantiated simultaneously:

Definition 67 *QME network concepts* N1 *and* N2 *are* CTA-consistent *iff they are directly or indirectly CTA-consistent.*

This definition is justified as follows:

Theorem 9 *Under CTA, the extensions of QME network concepts* N1 *and* N2 *intersect iff they are CTA-consistent.*

Proof: See Appendix B.

Thus far, we have focused on consistency between constraint network concepts. When we make observations for the sake of plan recognition, the observations are at the individual level, so we must also consider when an individual constraint network is consistent with a constraint network concept.

5.6 Monotonic Observation

This section shows how to make direct and indirect CTA-consistency inferences between an individual constraint network, e.g., the observation network, and constraint network concepts, e.g., plans. As noted in Section 5.4, under an assumption of monotonic observation, the observations may be abstract and they may become available incrementally. Monotonic updates to the observation network may include extensions, i.e., adding new nodes and/or arcs, as well as refinements, i.e., further specialization of existing ones. Thus, the observation network may be updated to instantiate plans that it does not currently instantiate. Under the monotonic update assumption, however, it can only instantiate networks with which it is currently CTA-consistent.

We will illustrate the incremental plan recognition process using the small plan library defined in Figure 5.5 on page 182 and diagrammed in Figure 5.6 on page 183. The taxonomy of action concepts referenced by this plan library is shown in Figure 5.7 on page 183. This library represents several simplified travel plans of the type that might be offered by a package tour company dealing in cultural and recreational vacations.[8] A larger travel plan library will be presented in Appendix A. A travel consultation system might use T-REX to help a client select from among the vacation plans in its library. We assume that no other vacations are offered through the system. The client may incrementally describe desired aspects of a trip in any order and at any level of abstraction. The observation network would

[8]The metric unit in this library is one day.

be used to represent (hypothetical) individual travel acts representing part of the desired travel package. In this application, the requirement of the closed library assumption that every individual plan will ultimately be finished according to Definition 50 means that the client is presumably seeking to select a known package tour. Our goal is to incrementally narrow down the set of plans that are consistent with the client's specifications as the specifications are expressed. At any time during this process, the taxonomy of currently consistent plans can be used to inform the client of the possibilities and to guide future choices. When only one travel plan remains consistent, the process is complete. If none remain, the client's specifications cannot be met by any one of the current package tour offerings.

A basic task is to identify plans that are directly consistent with the trip that the user has described so far. This requires a suitable mapping from the client's observation network to a directly consistent travel plan. The mapping process is essentially the same as between a pair of network concepts in Definition 64. The following definition reflects this:

Definition 68 *A* CTA-consistency mapping *from QME network individual* I *to QME network concept* C *maps every node* i *of* I *to a distinct node* c *of* C *such that:*

1. i *is CTA-consistent with* c
2. *every temporal (qualitative or metric) arc between a pair of nodes in* I *is consistent with the corresponding temporal arc in* C
3. *for every ordered pair of nodes* i1 *and* i2 *of* I *mapped to* c1 *and* c2 *of* C*, respectively,*
   (a) *for every equality arc* R = S *from* i1 *to* i2*, if roles* $\mathrm{R_{c1}}$ *and* $\mathrm{S_{c2}}$ *both exist*[9]*, they are CTA-restriction-consistent.*
   (b) *for every equality arc* R = S *from* c1 *to* c2*, if role* $\mathrm{R_{i1}}$ *and* $\mathrm{S_{i2}}$ *both exist, they are CTA-restriction-consistent.*

[9]Although *i1* and *c1* are CTA-consistent, it can be seen from Definitions 22 through 25 that they need not restrict the same set of roles (similarly for *i2* and *c2*).

```
(define-plan EUROPEAN-TRIP
  :steps ((s1 VISIT-EUROPEAN-COUNTRY)))

(define-plan EUROPEAN-MUSEUM-TRIP
  :steps ((s1 VISIT-EUROPEAN-COUNTRY)
          (s2 VISIT-MUSEUM))
  :qualitative-constraints ((s1 (contains) s2)))

(define-plan EUROPEAN-CAPITOL-TRIP
  :steps ((s1 VISIT-EUROPEAN-COUNTRY)
          (s2 VISIT-CAPITOL-CITY))
  :qualitative-constraints ((s2 (during) s1)))

(define-plan CLIMBING-TRIP
  :steps ((s1 CLIMB-MOUNTAIN)))

(define-plan TOUR-ENGLAND
  :steps ((s1 VISIT-ENGLAND)
          (s2 VISIT-LONDON)
          (s3 VISIT-BRITISH-MUSEUM))
  :qualitative-constraints ((s1 (contains) s2) (s2 (contains) s3))
  :metric-constraints ((10 ≤ s1_finish − s1_start ≤ 10)
                       (3 ≤ s2_finish − s2_start ≤ 3)))

(define-plan TOUR-FRANCE
  :steps ((s1 VISIT-FRANCE)
          (s2 VISIT-PARIS)
          (s3 VISIT-LOUVRE))
  :qualitative-constraints ((s1 (contains) s2) (s2 (contains) s3))
  :metric-constraints ((21 ≤ s1_finish − s1_start ≤ 21)
                       (4 ≤ s2_finish − s2_start ≤ 4)))

(define-plan TOUR-USA
  :steps ((s1 VISIT-USA)
          (s2 VISIT-WASHINGTON)
          (s3 VISIT-SMITHSONIAN))
  :qualitative-constraints ((s2 (during) s1) (s3 (during) s2))
  :metric-constraints ((3 ≤ s1_finish − s1_start ≤ 3)))

(define-plan TOUR-SWITZERLAND
  :steps ((s1 VISIT-SWITZERLAND)
          (s2 VISIT-ZERMATT)
          (s3 CLIMB-MATTERHORN))
  :qualitative-constraints ((s2 (during) s1) (s3 (during) s2))
  :metric-constraints ((4 ≤ s1_finish − s1_start ≤ 4)
                       (1 ≤ s2_finish − s2_start ≤ 1)))
```

Figure 5.5: Small Travel Plan Library

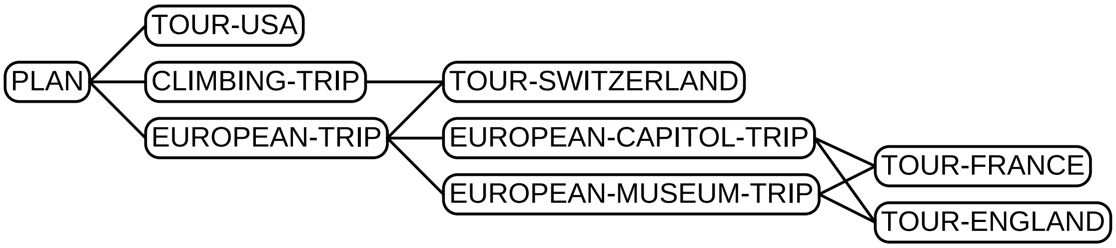


Figure 5.6: Small Travel Plan Taxonomy

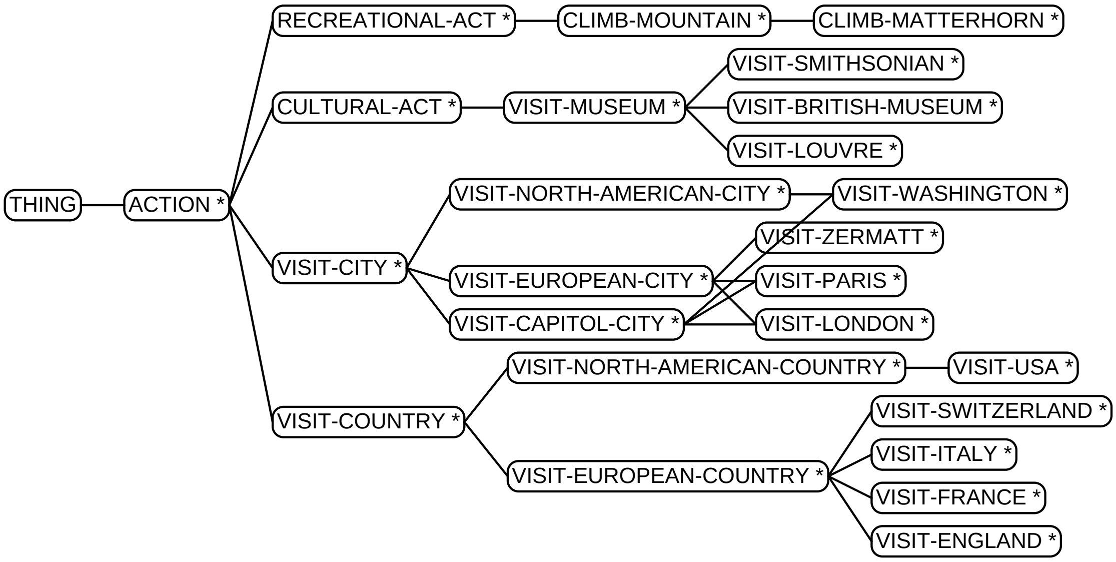


Figure 5.7: Small Travel Action Concept Taxonomy

As usual, a CTA-consistency mapping is written $I \mapsto C$. Direct CTA-consistency follows immediately from CTA-consistency mapping:

Definition 69 *Network individual* I *and network concept* C *are* directly CTA-consistent *iff* I $\mapsto$ C.

Suppose a client of our travel consultation service wishes to visit a European country. The initial observation network is as follows:

<u>OBS-A</u>: VISIT-EUROPEAN-COUNTRY1

These plans from our sample library are directly CTA-consistent with the observations:

- EUROPEAN-TRIP
- EUROPEAN-MUSEUM-TRIP
- EUROPEAN-CAPITOL-TRIP
- TOUR-ENGLAND
- TOUR-FRANCE
- TOUR-SWITZERLAND

Alternatively, suppose that the client expresses an interest in visiting a museum. Then, the initial observation network would be:

OBS-B: VISIT-MUSEUM1

As a result, the directly CTA-consistent plans would be:

- EUROPEAN-MUSEUM-TRIP
- TOUR-ENGLAND
- TOUR-FRANCE
- TOUR-USA

In the context of a plan library, the fact that an observation network is directly consistent with some plan concept may imply consistency with other plan concepts. For example, TOUR-ENGLAND is subsumed by EUROPEAN-TRIP, so the fact that OBS-B is directly consistent with TOUR-ENGLAND implies that it is also consistent with EUROPEAN-TRIP in the context of the library of Figure 5.5. More generally, if network individual I can potentially instantiate network concept C' such that it also instantiates network concept C, then I is *indirectly consistent* with C via C':

Definition 70 *Network individual* I *and network concept* C *are* indirectly CTA-consistent *iff there exists a network concept* C′ *such that:*

1. I $\mapsto$ C′
2. C $\mapsto$ C′
3. *For every node* c′ *of* C′ *mapped from node* i *of* I *and also mapped from node* c *of* C*,* i *and* c *are CTA-consistent*
4. *For every ordered pair of nodes* c1′ *and* c2′ *of* C′ *mapped from nodes* i1 *and* i2 *of* I *respectively and also mapped from nodes* c1 *and* c2 *of* C *respectively*
   (a) *every temporal (qualitative or metric) arc from* i1 *to* i2 *is consistent with the corresponding temporal arc from* c1 *to* c2
   (b) *for every equality arc* R = S *from* i1 *to* i2 *if role* $\mathrm{R_{c1}}$ *and* $\mathrm{S_{c2}}$ *both exist, they are CTA-restriction-consistent*
   (c) *for every equality arc* R = S *from* c1 *to* c2 *if role* $\mathrm{R_{i1}}$ *and* $\mathrm{S_{i2}}$ *both exist, they are CTA-restriction-consistent*

Let us again consider OBS-B: VISIT-MUSEUM1. It can be seen that both EUROPEAN-CAPITOL-TRIP and EUROPEAN-TRIP are indirectly CTA-consistent with OBS-B by way of TOUR-ENGLAND. As it happens, both are also indirectly CTA-consistent with OBS-B by way of TOUR-FRANCE. For a simple example of Definition 70, let OBS-B, EUROPEAN-CAPITOL-TRIP, and TOUR-ENGLAND play the parts of *I*, *C*, and *C′*, respectively (we will simply refer to nodes by the corresponding steps):

1. We have OBS-B $\mapsto$ TOUR-ENGLAND because its only (individual) action, VISIT-MUSEUM1, is CTA-consistent with the VISIT-BRITISH-MUSEUM action in TOUR-ENGLAND.

2. By mapping steps *s1* and *s2* of EUROPEAN-CAPITOL-TRIP to the identically named steps in TOUR-ENGLAND, it is clear that EUROPEAN-CAPITOL-TRIP $\mapsto$ TOUR-ENGLAND.

3. No step in TOUR-ENGLAND is mapped from both OBS-B and EUROPEAN-CAPITOL-TRIP.

4. No pair of steps in TOUR-ENGLAND is mapped from both OBS-B (which has only one step) and EUROPEAN-CAPITOL-TRIP.

CTA-consistency combines direct and indirect cases to identify the constraint network concepts (plans) that a constraint network individual (the observation network) might instantiate after being finished:

Definition 71 *Network individual* I *and network concept* C *are* CTA-consistent *under monotonic observation iff they are directly or indirectly CTA-consistent under monotonic observation.*

We have just seen several examples of direct and indirect consistency. As counterexamples, CLIMBING-TRIP and TOUR-SWITZERLAND are neither directly nor indirectly CTA-consistent with OBS-B. CTA-consistency is crucial to our constraint network recognition methodology, e.g., for plan recognition, which will be presented in Section 5.8. The correctness of CTA-consistency is established by the following:

Theorem 10 *Under CTA, network individual* I *can be monotonically updated to instantiate network concept* C *iff* I *and* C *are CTA-consistent.*

Proof: See Appendix B.

The consequences for plan recognition are as follows. With respect to observation network O, plan P is *necessary* if O instantiates P, else *optional* if O is CTA-consistent with P, else *impossible*. Observations are *CTA-consistent* with a plan if they can be monotonically updated to instantiate the plan under our closure assumptions about the action and plan taxonomies.

Given the plan library of Figure 5.5 and OBS-B: VISIT-MUSEUM1, T-REX assigns modalities to the plans as follows:

Necessary:	PLAN
Optional (directly):	EUROPEAN-MUSEUM-TRIP
	TOUR-ENGLAND
	TOUR-FRANCE
	TOUR-USA
Optional (indirectly):	EUROPEAN-TRIP
	EUROPEAN-CAPITOL-TRIP
Impossible:	CLIMBING-TRIP
	TOUR-SWITZERLAND

If the client expresses interest in visiting the museum while in a European country, we can update the OBS-B observation network to be:

OBS-B′: VISIT-MUSEUM1 (*during*) VISIT-EUROPEAN-COUNTRY2

Then the assignment of modalities to plans is revised as follows:

Necessary:	PLAN
	EUROPEAN-TRIP
	EUROPEAN-MUSEUM-TRIP
Optional (directly):	TOUR-ENGLAND
	TOUR-FRANCE
Optional (indirectly):	EUROPEAN-CAPITOL-TRIP
Impossible:	CLIMBING-TRIP
	TOUR-USA
	TOUR-SWITZERLAND

Notice that EUROPEAN-TRIP and EUROPEAN-MUSEUM-TRIP have become necessary, while TOUR-USA has become impossible. Next, suppose that the client constrains his or her European visit to be no more than fourteen days. The result is:

Necessary:	PLAN
	EUROPEAN-TRIP
	EUROPEAN-MUSEUM-TRIP
Optional (directly):	TOUR-ENGLAND
Optional (indirectly):	EUROPEAN-CAPITOL-TRIP
Impossible:	CLIMBING-TRIP
	TOUR-FRANCE
	TOUR-USA
	TOUR-SWITZERLAND

TOUR-FRANCE has now become impossible. Assuming that TOUR-ENGLAND, TOUR-FRANCE, TOUR-SWITZERLAND, and TOUR-USA are concrete plans, and that the remainder are abstract, there is only one concrete plan that is possible, namely TOUR-ENGLAND. Therefore, TOUR-ENGLAND must be necessary[10], and we are done. This last inference will be explained in Section 5.8.

As we have seen, optional plans are identified by means of direct or indirect consistency inferences. Finding indirect cases of consistency can be time consuming because of the search required, i.e., to identify a network concept C' as in Definition 70. To address this concern, the next section shows how to precompute certain aspects of indirect consistency inferences before plan recognition starts.

[10] So must its subsumers, including EUROPEAN-CAPITOL-TRIP.

5.7 Library Augmentation

We are motivated to speed the run-time recognition of indirectly consistent (plan) networks, just as we were for indirectly consistent concepts in Section 3.6. By Definition 70, a plan *P1* may be indirectly consistent with the observations through a directly consistent plan *P2* that it does not subsume. This occurs when some, but not all, instantiations of *P2* also instantiate *P1*. We will augment the plan library so that whenever this occurs, there exists a plan *P3* such that *P3* is directly consistent with the observations and *P1* subsumes *P3*. This general relationship is shown in Figure 5.8. After augmentation, every indirectly consistent plan such as *P1* can be identified by traversing explicit subsumption links upwards from directly consistent plans such as *P3*.

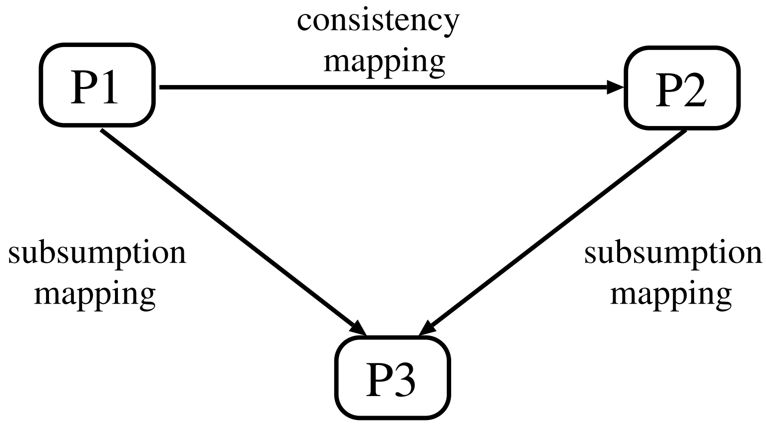


Figure 5.8: Library Augmentation

Figure 5.9 on page 190 reproduces PLAN-X and PLAN-Y from Figure 5.3. To illustrate the preceding discussion of plan library augmentation, PLAN-X and PLAN-Y play the parts of *P1* and *P2*, respectively. Figure 5.9 also shows PLAN-Z, which plays the part of *P3*. Wide broken arrows show the consistency mapping from PLAN-X to PLAN-Y and wide solid arrows show the subsumption mappings from each of those plans to PLAN-Z. Thus, it can be seen that every instantiation of PLAN-Y that also instantiates PLAN-X must instantiate PLAN-Z as well. Given the following observation network, PLAN-X is not directly consistent because it has no

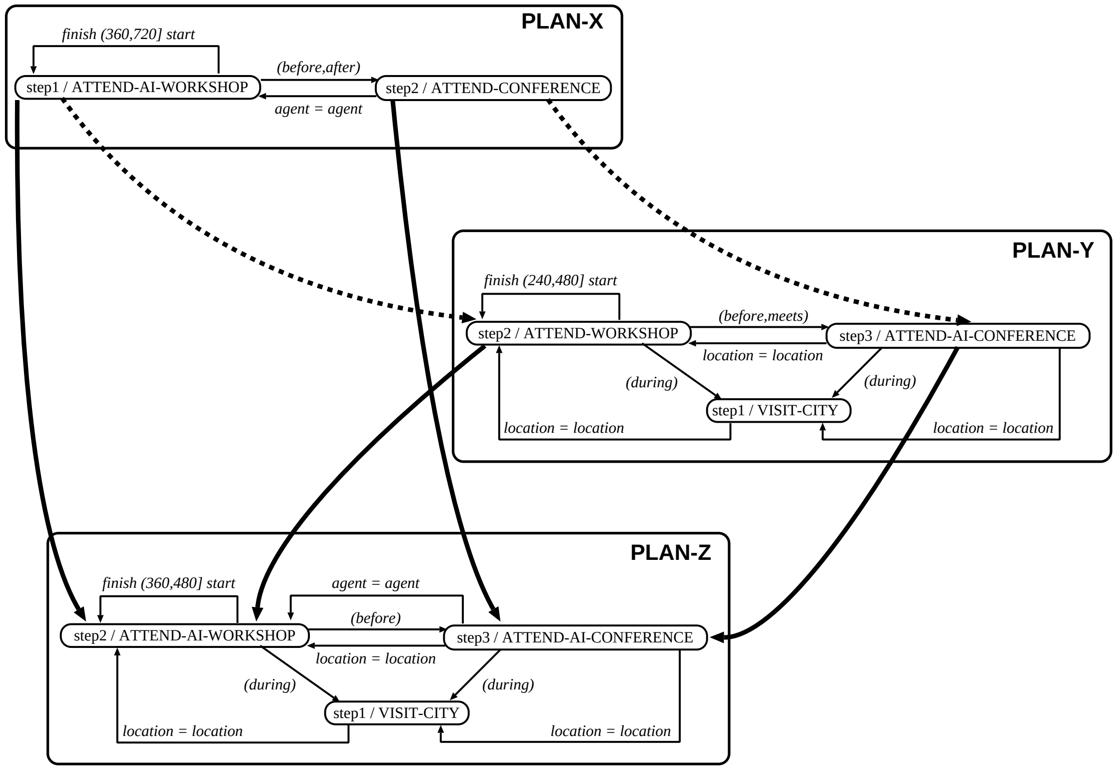


Figure 5.9: Plan Library Augmentation

counterpart for the observed VISIT-CITY1 action:

OBS: step2/ATTEND-AI-WORKSHOP2 *(during)* step1/VISIT-CITY1

Still, T-REX can readily infer that PLAN-X is indirectly optional because it subsumes PLAN-Z which is directly optional given these observations.

The augmentation process for network concepts is actually a bit more involved than it was for standard concepts because, considering Figure 5.8, plan *P1* might be consistent with plan *P2* in more than one way via multiple consistency mappings. Thus, we might need to add several plans based on the consistency mappings from *P1* to *P2*.[11]

[11]Considered disjunctively, that set of plans represents the conjunction of *P1* and *P2* under the CTA.

More precisely, we augment the (plan) network library as follows[12]: For every consistency mapping from a network *N1* to a concrete network *N2*, where *N1* neither subsumes nor is subsumed by *N2*, T-REX must ensure the existence of a network *N3* such that *N1* subsumes *N3* and *N3* is directly consistent with an observation network whenever that observation network can be monotonically updated to instantiate both *N1* and *N2*. Augmentation can be limited to cases where *N2* is concrete because of our assumption that the observation network will be continuously consistent with its ultimate plans, which are concrete. Note that *N2* always has at least as many nodes as *N1*. In case *N1* and *N2* have the same number of nodes, all consistency mappings between them are symmetric, so we need not consider mappings from *N2* to *N1* separately. *N3* is created by specializing *N2* according to a consistency mapping from *N1* so that each node and arc of *N2* mapped from a node or arc of *N1* is replaced by their conjunction. Also, any equality constraints between a pair of nodes in *N1* must be carried over to the images of those nodes in *N3*. As mentioned earlier, Figure 5.9 on page 190 illustrates the creation of a system-defined network, PLAN-Z, during the augmentation process.

As a result of the preceding discussion, the process of plan library augmentation can be formally specified:

Definition 72 *A library is* augmented *iff for all CTA-consistency mappings from network concept* N1 *to concrete network concept* N2*, there exists an explicitly defined network concept* N3 *such that:*

1. *For every node* n2 *of* N2 *with associated concept* C2 *mapped from node* n1 *of* N1 *with associated concept* C1*, there exists a distinct node* n3 *of* N3 *with associated concept* C3 *such that* $C3 \equiv C1 \wedge C2$

2. *For every (additional) node* n2 *of* N2 *with associated concept* C2 *not mapped from a node of* N1*, there exists a distinct node* n3 *of* N3 *with associated concept* C2

[12]Being for internal use, the added networks do not violate the CTA.

3. *For every ordered pair of nodes* m2 *and* n2 *of* N2 *mapped from nodes* m1 *and* n1 *of* N1*, respectively, and their counterparts* m3 *and* n3 *of* N3*, respectively,*

   (a) *For every temporal (qualitative or metric) arc from* m2 *to* n2*, there is a corresponding arc from* m3 *to* n3 *whose constraint is the conjunction of the constraints on the arc from* m2 *to* n2 *and the corresponding arc from* m1 *to* n1

   (b) *For every equality arc from* m1 *to* n1 *there exists an identical equality arc from* m3 *to* n3

   (c) *For every equality arc from* m2 *to* n2 *there exists an identical equality arc from* m3 *to* n3

4. *For every pair of nodes* m2 *and* n2 *of* N2 *not both mapped from nodes in* N1*, and for every arc between* m2 *and* n2*, there is a corresponding arc between* m3 *and* n3 *whose constraint is identical*

Following augmentation, all plans that are indirectly consistent with an individual can be identified through subsumption:

Definition 73 *Network individual* I *is* indirectly CTA-consistent *with network concept* C *in an augmented library iff there exists a network concept* C′ *such that* I $\mapsto$ C′ *and* C′ $\Rightarrow$ C.

Earlier, when we introduced the network individual labeled OBS on page 190, we explained informally that OBS $\mapsto$ PLAN-Z $\Rightarrow$ PLAN-X.

The following result in the context of an augmented library corresponds to Theorem 10 in the context of an unaugmented library:

Theorem 11 *Under the CTA and with an augmented library, network individual* I *can be monotonically updated to instantiate network concept* C *iff* I *and* C *are CTA-consistent using Definition 73 instead of Definition 70.*

Proof: See Appendix B.

This section has shown how to augment a library as needed to hasten the identification of plans that are indirectly consistent with an observation network. Augmentation is not always necessary. For example, augmentation does not add any new plans to the small plan library defined in Figure 5.5 and diagrammed in Figure 5.6.

Identification of indirectly consistent plans is a crucial part of identifying all the optional plans, which in turn, is central to predictive plan recognition. The next section explains how we conduct an ongoing plan recognition process by partitioning a plan library with respect to an observation network.

5.8 Recognition via Library Partitioning

The aim of library partitioning is to associate an appropriate modality, either necessary, optional, or impossible, with each plan as observations evolve. Figure 5.10 on page 194 illustrates the variety of generic relationships that may hold at any moment between an observation network and a set of plan networks in an augmented library. There is an instantiation mapping from plan *P1* to the observation network, so the observations instantiate *P1*, and *P1* is necessary with respect to those observations. While there is no instantiation mapping from plan *P2* to the observation network, there is a consistency mapping the other way. Consequently, *P2* is directly optional. The subsumption mapping from plan *P3* to plan *P2*, in combination with the consistency mapping from the observation network to *P2*, shows that *P3* is indirectly optional via *P2*. Assuming that the plan library has been augmented in accord with Section 5.7, all indirectly optional plans are discovered in this fashion via subsumption. Finally, plan *P4* does not engender any of the relationships to the observations exemplified by *P1*, *P2*, or *P3*. Since it is neither necessary nor optional, under the closed library assumption it must be impossible.

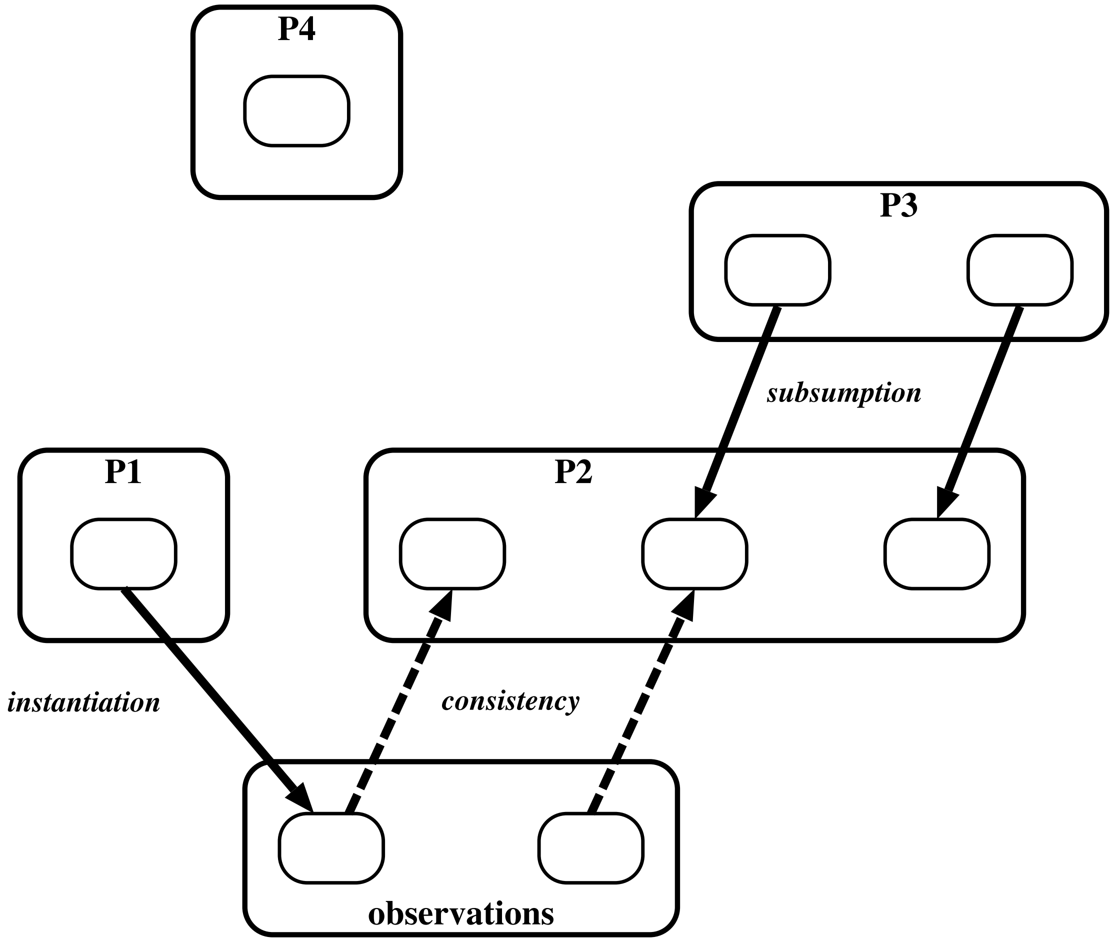


Figure 5.10: Computing Modalities in an Augmented Library

For incremental plan recognition, the process of partitioning a plan library by modality is essentially identical to the terminology partitioning scheme of Section 3.7. The only distinction here is that plans take the place of concepts. Definitions 28, 29, and 30 have obvious counterparts with respect to a plan library:

Definition 74 *A plan is an MSN plan iff it is necessary and none of its children are necessary.*

Definition 75 *A plan is an MGO plan iff it is optional and none of its parents are optional.*

Definition 76 *A plan is an MSO plan iff it is optional and none of its children are optional.*

It may happen that a single plan is both an MGO plan and an MSO plan.

The incremental recognition algorithm of Section 3.7 applies to our plan recognition problem, merely by substituting plan networks for standard concepts. This demonstrates the generality of our algorithm: it requires only an augmented subsumption-based taxonomy and the ability to make consistency inferences from an individual, e.g., either a standard description logic individual or an observation network, to the constituents of the taxonomy.

We will illustrate the process of incremental recognition by reprising a series of observations from Section 5.6. Given the plan library of Figure 5.5 on page 182 and OBS-B: VISIT-MUSEUM1, the ensuing recognition state is captured as:

$$\begin{aligned} \text{MSNs} &= \{\text{PLAN}\} \\ \text{MGOs} &= \{\text{EUROPEAN-TRIP, TOUR-USA}\} \\ \text{MSOs} &= \{\text{TOUR-FRANCE, TOUR-ENGLAND, TOUR-USA}\} \end{aligned}$$

Assuming that the client also expresses interest in visiting a museum while in a European country, the OBS-B observation network was monotonically updated to:

OBS-B′: VISIT-MUSEUM1 (*during*) VISIT-EUROPEAN-COUNTRY2

That generates the following recognition state:

$$\begin{aligned} \text{MSNs} &= \{\text{EUROPEAN-MUSEUM-TRIP}\} \\ \text{MGOs} &= \{\text{EUROPEAN-CAPITOL-TRIP}\} \\ \text{MSOs} &= \{\text{TOUR-ENGLAND, TOUR-FRANCE}\} \end{aligned}$$

Given the taxonomy, this partition clearly indicates the plans that are consistent with the observation network.

As with predictive recognition in standard description logic, we can sometimes obtain better results from our assumption that the taxonomy is closed. T-REX does not currently use LCS inferences to extract constraints on the observation network from the closed library and perhaps update the recognition state accordingly.[13] Instead, it could (but does not yet) adopt a sound but incomplete alternative which we call the *explicit common subsumer* (ECS) inference: if an explicitly defined plan P subsumes every MSO plan, then P is necessary. This condition is readily discovered by searching upward from each MSO plan. In the immediately preceding recognition state, the ECS inference determines that EUROPEAN-CAPITOL-TRIP is necessary[14], leading to a more informed recognition state:

$$\begin{aligned} \text{MSNs} &= \{\text{EUROPEAN-MUSEUM-TRIP, EUROPEAN-CAPITOL-TRIP}\} \\ \text{MGOs} &= \{\text{TOUR-ENGLAND, TOUR-FRANCE}\} \\ \text{MSOs} &= \{\text{TOUR-ENGLAND, TOUR-FRANCE}\} \end{aligned}$$

Next, supposing that the client's European visit is constrained to be no more than fourteen days, the result is:

$$\begin{aligned} \text{MSNs} &= \{\text{EUROPEAN-MUSEUM-TRIP, EUROPEAN-CAPITOL-TRIP}\} \\ \text{MGOs} &= \{\text{TOUR-ENGLAND}\} \\ \text{MSOs} &= \{\text{TOUR-ENGLAND}\} \end{aligned}$$

The ECS inference applies again in this state, leading to the conclusion that TOUR-ENGLAND uniquely satisfies the client's criteria:

$$\begin{aligned} \text{MSNs} &= \{\text{TOUR-ENGLAND}\} \\ \text{MGOs} &= \{\} \\ \text{MSOs} &= \{\} \end{aligned}$$

[13]The reason will be discussed in Section 5.13.

[14]This particular result could also be inferred from the following rule: if there is only one MGO plan, and no necessary plan is concrete, then the MGO plan must be necessary.

An extended example of incremental plan recognition under monotonic observation with a larger plan library will be presented in Appendix A. The next section concentrates on the more restrictive case of perfect observation.

5.9 Perfect Observation

So far, this chapter has focused on incremental plan recognition with respect to a monotonic observation assumption. We have permitted observations to be made in terms of abstract actions, disjunctive qualitative temporal constraints, etc. Consequently, we allowed any monotonic update to the observation network, including retroactive specialization of the action types associated with previously observed steps and constraints among them. In some cases, however, a stronger *perfect observation assumption* may be quite justified. For instance, we can flawlessly capture a user's interactions with software systems such as operating systems or graphical user interfaces. If observations are perfect, the types of observed actions are leaves[15] in the action concept taxonomy and the observed binary constraints are exact. That is, observed qualitative temporal relationships are non-disjunctive, and observed metric constraints denote intervals that reduce to time points, etc. Under perfect observation, the observation network may be extended with additional actions, as well as with temporal constraints between the additional actions or between an additional action and a previously observed action. Existing actions and temporal relationships may not be modified, i.e., neither refined nor retracted. By making the more stringent assumption of perfect observations, the recognition process is somewhat simpler.

Under perfect observation, the observation network can be seen as instantiating a portion of the ultimate plan(s), i.e., the observation network instantiates a network composed of a subset of the nodes and arcs in an ultimate plan. For example, consider PLAN-B in Figure 5.11 (repeated from Figure 1.1, whose plan networks were diagrammed in Figure 4.3 on page 106). The OBS101 observation network

[15]Strictly speaking, the types are conjunctions of one or more leaves.

In PLAN-B, an agent visits a city, during which time s/he attends a workshop for between 240 and 480 minutes, then attends an AI conference, all in the same location.

(define-plan **PLAN-B**
:steps ((step1 VISIT-CITY)
(step2 ATTEND-WORKSHOP)
(step3 ATTEND-AI-CONFERENCE))
:qualitative-constraints ((step2 (*during*) step1)
(step3 (*during*) step1)
(step2 (*before*) step3))
:metric-constraints (($240 < \text{step2}_{finish} - \text{step2}_{start} \leq 480$))
:co-reference-constraints ((agent(step1) = agent(step2) = agent(step3))
(location(step1) = location(step2) = location(step3))))

Figure 5.11: PLAN-B (repeated from Figure 1.1)

shown in Figure 5.12 on page 199 (with individuals borrowed from Figure 4.4) instantiates the portion of PLAN-B not including its *step1* (i.e., VISIT-CITY) or constraints involving that step.[16]

Recognition from perfect observations consists of detecting such *partial instantiation* relationships. This is done by searching for a *partial instantiation mapping* as follows:

Definition 77 *A* structural partial instantiation mapping *from network individual* I *to network concept* C *maps every node* i *of* I *to a distinct node* c *of* C *such that:*

1. i *instantiates* c
2. *for all arc types T, every arc of type T between a pair of nodes in* I *instantiates the corresponding arc of type T in* C.

[16] OBS101 can be updated with further perfect observations to have the same description as the finished description of OBS100 shown on page 163, thus demonstrating that OBS101 can be updated under perfect observation to fully instantiate PLAN-B.

```
(create-individual A-RANDOM-RESEARCHER)

(create-individual MINOR-FIASCO)

(create-individual MAJOR-BOONDOGGLE)

(create-individual METROPOLIS)

(create-individual ATTEND-WORKSHOP98
    (and ATTEND-WORKSHOP
         (exactly 1 agent)
         (fills agent A-RANDOM-RESEARCHER)
         (exactly 1 workshop)
         (fills workshop MINOR-FIASCO)
         (exactly 1 location)
         (fills location METROPOLIS)))

(create-individual ATTEND-AI-CONFERENCE99
    (and ATTEND-AI-CONFERENCE
         (exactly 1 agent)
         (fills agent A-RANDOM-RESEARCHER)
         (exactly 1 conference)
         (fills conference MAJOR-BOONDOGGLE)
         (exactly 1 location)
         (fills location METROPOLIS)))

(create-individual-plan OBS101
    :steps ((step1 ATTEND-WORKSHOP98)
            (step2 ATTEND-AI-CONFERENCE99))
    :metric-constraints ((360 ≤ step1_finish – step1_start ≤ 360)
                         (1440 ≤ step2_start – step1_finish ≤ 1440)
                         (4320 ≤ step2_finish – step2_start ≤ 4320)))
```

Figure 5.12: Perfectly Observed Plan Individual

As usual, we can tailor our definition for QME networks and handle the domain-independent aspect of equality constraints that is non-structural:

Definition 78 *A* partial instantiation mapping *from QME network individual* I *to QME network concept* C *maps every node* i *of* I *to a distinct node* c *of* C *such that:*

1. i *instantiates* c
2. *every temporal (qualitative or metric) arc between a pair of nodes in* I *instantiates the corresponding temporal arc in* C
3. *for every ordered pair of nodes* i1 *and* i2 *of* I *mapped to* c1 *and* c2 *of* C*, respectively, and for every equality arc* R = S *from* c1 *to* c2*:*
   (a) *there is an equality arc* R = S *from* i1 *to* i2*, or*
   (b) *all of the following hold:*
      i. $fillers(\mathrm{R}_{\mathrm{i1}}) = fillers(\mathrm{S}_{\mathrm{i2}})$
      ii. $|\, fillers(\mathrm{R}_{\mathrm{i1}})\,| = \textit{at-most}(\mathrm{R}_{\mathrm{i1}})$
      iii. $|\, fillers(\mathrm{S}_{\mathrm{i2}})\,| = \textit{at-most}(\mathrm{S}_{\mathrm{i2}})$

A partial instantiation mapping is a special case of a CTA-consistency mapping as in Definition 64 on page 175. To distinguish the two, we write a partial instantiation mapping from individual I to concept C as $I \hookrightarrow C$. In our example, OBS101 $\hookrightarrow$ PLAN-B is demonstrated by mapping *step1* and *step2* of OBS101 to *step2* and *step3* of PLAN-B, respectively. We now detail the application of Definition 78 to this example, simply referring to nodes by the associated actions:

1. ATTEND-WORKSHOP98 instantiates ATTEND-WORKSHOP and ATTEND-AI-CONFERENCE99 instantiates ATTEND-AI-CONFERENCE.
2. Each temporal constraint in OBS101 must be checked with regard to its counterpart in PLAN-B. We will just detail this for the explicit constraints; it is not hard to verify that the implicit constraints work out too.

(a) The 360 time unit duration of ATTEND-WORKSHOP98 instantiates the 240 to 480 time unit duration permitted for the ATTEND-WORKSHOP.

(b) The 1440 time unit gap between ATTEND-WORKSHOP98 and ATTEND-AI-CONFERENCE99 respects the constraint that ATTEND-WORKSHOP occurs before ATTEND-CONFERENCE.

(c) The 4320 time unit duration of ATTEND-CONFERENCE99 is fine, since the duration of ATTEND-CONFERENCE is unconstrained.

3. The equality constraints of PLAN-B pertaining to nodes mapped from OBS101 are satisfied by OBS101 because:

(a) Both ATTEND-WORKSHOP98 and ATTEND-AI-CONFERENCE99 have exactly one agent, namely A-RANDOM-RESEARCHER.

(b) Both ATTEND-WORKSHOP98 and ATTEND-AI-CONFERENCE99 have exactly one location, namely METROPOLIS.

When the perfect observation assumption is appropriate, partial instantiation mapping has several advantages over consistency mapping. Nodes are compared for instantiation relationships that would be computed by the description logic system anyway, so consistency inferences are not required. Comparing temporal arcs for instantiation is marginally faster than for consistency (which is still constant time). Equality arcs in the individual network are superfluous and can be ignored.[17] Also, strictly speaking, clause 3(a) of Definition 78 is superfluous under perfect observation. Finally, with the assumption of perfect observation, our algorithm for incrementally partitioning the plan library as observations become available need not consider re-expanding the optional region.

Indirect consistency under perfect observation differs from the case of monotonic observation only by using partial instantiation as a special case of direct consistency. For brevity, we will proceed directly to the results for augmented libraries:

[17] Under perfect observation, if an equality constraint is satisfied, its operands *must* be identical, closed sets of fillers.

Definition 79 *Under perfect observation, network individual* I *is* indirectly CTA-consistent *with network concept* C *in an augmented library iff there exists a network concept* C′ *such that* $\mathrm{I} \hookrightarrow \mathrm{C}'$ *and* $\mathrm{C}' \Rightarrow \mathrm{C}$.

As a simple example, consider the following observation network, which refers to VISIT-CITY97 introduced in Figure 5.1 on page 161:

```
(create-individual-plan OBS200
    :steps ((step1 VISIT-CITY97)))
```

Consider the library of Figure 1.1 on page 9, whose plan networks are diagrammed in Figure 4.3 on page 106. Since PLAN-A contains no VISIT-CITY step, it is not the case that OBS200 $\hookrightarrow$ PLAN-A. Nonetheless, there is an indirect consistency relationship because OBS200 $\hookrightarrow$ PLAN-B $\Rightarrow$ PLAN-A.

Again, CTA-consistency combines direct and indirect cases to identify the constraint network concepts (plans) that a constraint network individual (the observation network) might instantiate after being finished:

Definition 80 *Network individual* I *and network concept* C *are* CTA-consistent *under perfect observation iff they are directly or indirectly CTA-consistent under perfect observation.*

Then we have:

Theorem 12 *Under perfect observation, CTA, and with an augmented library, network individual* I *can be monotonically updated to instantiate network concept* C *iff*

1. $\mathrm{I} \hookrightarrow \mathrm{C}$, *or*

2. I *is indirectly CTA-consistent with* C *according to Definition 79.*

Proof: See Appendix B.

This section has addressed the case of perfect observation, which is more restrictive than monotonic observation. In contrast, the following section goes in the other direction to examine unrestricted observation.

5.10 Unrestricted Observation

T-REX actually provides support for arbitrary modification and retraction of observations.[18] To reach any useful conclusions, it is necessary to assume in advance that generalization and retraction will not happen. Thus our existing definition of potential instantiation under monotonic observation still applies. When allowing nonmonotonic observations, however, recognition results are reduced to contingent status, i.e., plans considered "necessary" given some observation network may revert to optional status later on. Indeed, seemingly "impossible" plans may later become possible. If an observed action instance is modified, it is automatically reclassified by K-REP (CLASSIC does not allow concept modification and reclassification). Nonmonotonic observation could have unfortunate performance consequences. We must effectively be able to undo any constraint propagation in the observation network, since the justification for the propagation may cease to exist. Retraction in the observation network is currently done by recomputation (recompleting the observation network starting from the explicitly stated constraints). Presumably it could also be supported via truth maintenance, but the cost of tracking dependencies may not be worthwhile.

To this point, we have relied on the assumption that a single plan is being observed. That assumption is fine for our travel application, where the goal is to select a single package tour. Of course, it is not realistic for all applications. Next we investigate the problem of recognizing multiple plans.

[18]But not yet in the incremental version of its library partitioning code.

5.11 Simultaneous Recognition

When no single plan can account for the observations, T-REX assumes that more than one plan is underway. First, it must be able to relate the observations to a group of plans. T-REX (conceptually) places the nodes from several plans into one plan network, preserving the original constraints on those nodes. Relationships between nodes taken from different plans are unconstrained. Thus, a *multiple plan network* allows its constituent plans to be interleaved in any way. As in [Kautz, 1991b], observed actions can be shared among plans.

A set of plans accounts for all observed actions iff there is a consistency mapping from the observation network to their multiple plan network. T-REX also needs a way to explore the set of possible plan combinations. For reasonable performance, it seems essential to make assumptions about the number of simultaneous plans, and thus constrain the combinations to be explored. Also, in the absence of cardinality assumptions, we would be forced to concede that all plans are always possible, since any given plan might commence in the future. Kautz's *minimum cardinality assumption* addresses this problem. Following the principle of Occam's razor, Kautz prefers to explain events by the smallest number of plans. His implementation simply considers plans pairwise when a single plan does not suffice to explain the observations, and failing that, three at a time and so on, *ad infinitum* [Kautz, 1991b]. T-REX also adopts the minimum cardinality assumption, which is a generalization of the single plan assumption that it uses as long as it can.[19] As a first cut at improvement, T-REX only considers those multiple plan networks that have a consistent action for every observed action.

As an example, consider observation network OBS-9 in the context of the plan library defined in Figure 1.1 on page 9:

- OBS-9: VISIT-CITY10 (*before*) VISIT-NATIONAL-PARK11

[19]The single plan assumption is embodied in the definition of the closed library assumption.

Since no single plan can account for the observations, T-REX considers pairs of plans that together cover both VISIT-CITY and VISIT-NATIONAL-PARK actions. Only the combination of PLAN-B and PLAN-C meets this criterion. Combinations involving PLAN-A are rejected at this point. T-REX goes on to verify that there is a consistency mapping from OBS-9 to a multiple plan network composed from PLAN-B and PLAN-C.

T-REX searches for combinations of *concrete plans* only; if T-REX were told that PLAN-A is an abstract plan, and hence not an end in itself, then even the preliminary test for coverage of all the observations would have been omitted.

In a more general approach, we might wish to find a minimum *cost* set of plans that potentially account for the observations, where cost need not be set cardinality. If we permit sharing of observed actions between plans, it is the *set covering* problem. Otherwise it is *set partitioning*. Integer programming techniques are applicable and could be considered for use in this context.

The present approach to sharing and interleaving steps between plans is undoubtedly inadequate, in general. There is a need for sound principles to control this process. Some positive work in this area was done in the context of multiple-trauma care, which involves both diagnostic and therapeutic plans for different injuries [Gertner *et al.*, 1995].

Useful plan recognition systems must be able to respond to recognized plans by taking appropriate action. Therefore, the next section looks at active recognition.

5.12 Active Recognition

So far, our discussion has dealt with a passive aspect of plan recognition, namely the ability to answer queries regarding the status (necessary, optional, or impossible) of a plan. A central concern of plan recognition work is to provide users with helpful responses, e.g., [Allen, 1983b]. T-REX also has a capacity for *active* *plan recognition*[20] which enables it to take the initiative and act when the modality of a plan changes. In particular, T-REX actively responds to recognition results through *demons* created to fire whenever a certain plan undergoes specified status changes, e.g., from optional to necessary. The action of a demon is implemented as an escape to the host language (currently Common Lisp). Such demons might perform a service for the user of a travel consultation system, e.g.,to point out relevant discount fares, etc. The following example defines a demon for T-REX which is activated when the modality of the TOUR-ENGLAND plan in Figure 5.5 changes to necessary:

```
(defdemon TOUR-ENGLAND
  (lambda (self)
    (format t
            "~%For the ~s package, a special airfare is available."
            (name self)))
  :to (:necessary))
```

Following the *defdemon* operator is the name of the plan (in this case TOUR-ENGLAND), a Common Lisp expression which is evaluated when the demon fires, and keyword argument(s) that specify the demon's applicability.

It should be noted that this style of active recognition could just as well be used in the concept recognition setting of Chapter 3. Not coincidentally, there is a great deal in common between our methods for predictive concept recognition in description logic and predictive plan recognition based on description logic. A comparison ensues.

[20] No relation to active vision or active learning.

5.13 Comparison with Predictive Recognition in Standard Description Logic

How does predictive concept recognition compare with predictive plan recognition? Consider the distinction between standard concept consistency and constraint network consistency. The comments in Section 4.4.6 about standard concept subsumption vs. constraint network subsumption via apply here too:

1. The labels of nodes in a network are not significant, in contrast to the names of roles in a concept.
2. Direct consistency is established by searching for a consistency mapping, rather than relying on the labels of nodes.
3. All network concepts are considered *fully defined*, i.e., non-primitive.
4. There may be multiple mappings from one network to another.

The last two points are particularly notable for plan recognition.

For point 3, recall that primitive concepts specify necessary but not sufficient conditions for membership in the class of interest. If we allowed primitive plans, the fact that a plan is primitive would mean that it has properties which do not follow from its definition. Naturally, T-REX could not reason about this primitiveness, so it could not justify a consistency inference from the observation network to a primitive plan. Thus it seems useless to permit primitive plans for the purpose of plan recognition.

For point 4, recall that in Section 3.8 we used the least common subsumer inference to derive additional constraints on a K-REP individual. T-REX does not currently compute the commonality among a set of plans in order to derive additional constraints on the observation network. For constraint networks, the task is greatly complicated by possibility of multiple mappings between pairs of networks. Details are not entirely worked out at this time.

5.14 Conclusion

This chapter has introduced a new view of plan recognition as a process that dynamically partitions the plan library into modalities, e.g., *necessary*, *optional*, and *impossible*, according to observations of the environment. We considered a range of scenarios regarding the mutability of observations, namely perfect, monotonic, and unrestricted observations. We designed an incremental, predictive recognition algorithm that leverages the plan taxonomy's enforced semantics to limit the number of plans that must be examined. Our approach unifies representation and reasoning work in plan recognition and description logic. This work is implemented in the T-REX system. We have presented examples from the domain of travel plans. Other domains can be addressed as well, including cooking plans (recipes) in [Weida and Litman, 1992] and patterns of error conditions in Automated Teller (ATM) machines [Weida and Litman, 1994]. There are many interesting and challenging directions along which this work can be further developed.

Chapter 6

Related Work

This chapter places our work in the context of related research. We compare and contrast our results with the most closely related contributions from the literature of description logic, temporal reasoning, configuration, and plan recognition. Our discussion of related work is ordered as follows:

1. Configuration via description logic
2. Recognition in description logic
3. Plans in description logic
4. Temporal concepts
5. Production rules
6. Deductive plan recognition
7. Candidate elimination
8. Intentional plan recognition

This sequence is not intended to reflect the importance of work in these areas.

6.1 Configuration via Description Logic

Many papers on configuration have appeared in the artificial intelligence literature, and we will not attempt to survey the field. [Kramer, 1991] perhaps exemplifies the state of the art. We have argued earlier that description logic offers unique advantages for the representation of conceptual knowledge required by configuration systems. This section surveys other research that has pursued this idea, namely BEACON [Searls and Norton, 1990], MESON [Owsnicki-Klewe, 1988], and PROSE [Wright *et al.*, 1993].

6.1.1 BEACON

The BEACON system developed at Burroughs[1] was an interactive, logic-based configurator implemented in Prolog. It was the first configurator based on a description logic system, KNET [Freeman, 1986; Searls and Norton, 1990]. BEACON's concepts are organized in a subsumption hierarchy, and also in an *aggregation hierarchy* by means of their roles. Typically, roles relate a component, such as a B25COMPUTER, to its components, such as a B25 MONITOR, a B25PROCESSOR-MODULE, and a B25GRAPHICS-MODULE. BEACON was strongly influenced by logic programming, and the aggregation hierarchy underlies a Prolog-style search for a suitable configuration. Note that while a concept's roles are not logically ordered, in practice they appear sequentially within the concept's description. An individual computer system description is built up while traversing the aggregation hierarchy in depth-first, left-to-right fashion, with backtracking as required. Concepts are thus instantiated incrementally, by analogy to Prolog's instantiation of logic variables. Concrete choices among leaves are made by querying the user, e.g., upon encountering the B25-MONITOR concept, BEACON might offer a choice between a BW-CRT[2] and a COLOR-CRT. Constraints in the form of Prolog procedures are housed in concepts. Thus, choice of a COLOR-CRT triggers forward

[1] Later, UNISYS.
[2] Black and White.

propagation which constrains the system to have a graphics module. Not surprisingly, BEACON tends to share the strengths and weaknesses of Prolog. Aspects of closed terminology reasoning are effectively achieved by the procedural traversal of the abstraction hierarchy. User-interaction in BEACON is liable to be cumbersome, because the order of choices is wired into the aggregation hierarchy. Constraint derivation is complete and consistent just to the extent that the set of constraints attached to concepts are complete and consistent.

6.1.2 MESON

An early use of description logic for configuration was reported in [Owsnicki-Klewe, 1988], which cast the entire configuration problem as a matter of maintaining internal knowledge base consistency, i.e., logical contradictions should follow from all invalid configurations but no valid ones. This goal is more ambitious than ours or that of [Wright *et al.*, 1993]. However, [Owsnicki-Klewe, 1988] did no closed terminology reasoning. Much of the reasoning in MESON is done through forward-chaining rules, which we will discuss in the context of the PROSE system.

6.1.3 PROSE

PROSE [Wright *et al.*, 1993] is a successfully deployed configurator featuring product knowledge bases written in CLASSIC [Borgida *et al.*, 1989]. Like PROSE, our work positions the description logic system as a product knowledge reasoner — a key module in a larger configurator architecture. In both cases, the description logic system maintains internal knowledge base consistency during configuration, but relies on configuration-specific modules for further reasoning. We share the views of the PROSE developers that (1) description logic fosters a reasonable, even natural approach to product knowledge representation, and (2) knowledge engineering efforts benefit from enforcing internal knowledge base consistency.

Closed world reasoning in CLASSIC is limited to the sets of fillers of particular roles on a particular concept or individual. Like BEACON and MESON, but in contrast with our work, PROSE does no explicit closed terminology reasoning. Recall that we exploit the closed terminology and its subsumption-based organization during interactive configuration to:

1. Efficiently track the types of systems and components that are consistent with the user's current choices.
2. Infer constraints on the system and components that follow from current choices and the taxonomy's specific contents.
3. Offer the user and the configuration engine guidance by suitably restricting future choices.
4. Characterize the most general choices available for refining the description of a particular individual.
5. Formally decide when an individual's description may be considered finished.

Little detail on the operation of PROSE is provided in [Wright *et al.*, 1993], but the configuration process is driven by a search which "appears to be related to iterative deepening." Unlike our work, it seems that PROSE does not envision an interactive search for a configuration where the user participates in making choices along the way. While PROSE presumably uses CLASSIC to help it address other aspects of the services we provide, there is no indication that PROSE is able to exploit a CTA to derive information. In any case, we believe that domain-independent services such as closed terminology reasoning are best supported within the description logic system, rather than implementing them anew in each application.

Unlike the current version of K-REP, MESON and CLASSIC support forward chaining rules that are useful for configuration. Such rules state that every possible individual instantiating some concept *C1* must also instantiate another concept *C2*. Constraints on *C2* may then be added to the individual, and the chaining process may continue.

6.2 Recognition in Description Logic

Prior work on recognition in description logic was only concerned with actual instantiation of concepts by instances. Thus, our predictive recognition methodology differs from traditional recognition in several key respects. First, we infer potential instantiations of *optional* concepts by drawing a correspondence from a partially described individual to part of a concept's description. Second, we use consistency inferences which exploit certain assumptions, namely the closed terminology and monotonic update assumptions.

For the most part, traditional recognition algorithms in description logic are part of the field's unpublished lore. One exception is [MacGregor, 1988], which caches the results of tests made during recognition to avoid unnecessary work while revising results in case the individual's definition changes. Like all other description logic recognition work prior to this thesis, it only seeks to identify concepts that the individual currently instantiates. Another exception is [Kindermann, 1992]: when an individual's description is updated non-monotonically, a data dependency network among individuals is used to reduce the number of dependent individuals which must be considered for reclassification.

6.3 Plans in Description Logic

Prior work on implemented plan subsumption systems considered atemporal plans in the context of plan generation [Wellman, 1990], as well as plans restricted to describing temporal sequences for the purpose of information retrieval [Devanbu and Litman, 1991]. There is also contemporaneous work on state-based reasoning with plans limited to simple sequences [Heinsohn *et al.*, 1992a]. After considering each of these systems in turn, we summarize their comparison in Table 6.1. We then discuss an unimplemented description logic of action and time that followed T-REX by proposing a formal semantics for a similar plan language [Artale and

Franconi, 1993; Artale and Franconi, 1994]. We conclude by comparing several different types of plan subsumption that have been studied.

We have not concentrated on detailed representation of the actions within description logic plans, but a number of authors have done so. CLASP [Devanbu and Litman, 1991] uses a STRIPS-like representation [Fikes and Nilsson, 1971], while Artale and Franconi use an Allen-style [Allen, 1991] representation [Artale and Franconi, 1993; Artale and Franconi, 1994]. There has also been significant work on action representation in description logic for the problem of understanding natural language instructions [DiEugenio and Webber, 1992; Di Eugenio, 1994].

6.3.1 SUDO-PLANNER

Wellman studied the formulation of tradeoffs in the context of planning medical therapy under uncertainty [Wellman, 1990]. He proposed an architecture for a constraint-posting planner named SUDO-PLANNER, which classifies a terminology of partial plan descriptions representing the explored portion of the search space. His proposal integrates a *dominance prover* which can prove that one class of plans characterized by a partial description dominates another, in the sense that some realization of the first class is at least as good as every realization of the second. Then his system is justified in pruning the entire dominated plan class from the search space. Wellman's plans are unorganized collections of actions classified in a subsumption-based hierarchy. Although his plans are entirely atemporal, he acknowledges the need for an explicit representation of time. In [Weida, 1993], we proposed some preliminary ideas for integrating Wellman's idea of a dominance proving planner with an Allen-style temporal planner [Allen, 1991].

6.3.2 CLASP (Devanbu and Litman)

CLASP (CLAssification of Scenarios and Plans)[3] was designed to aid in retrieval of software for telephone switching devices. It is the first description logic-based system to focus on plan library management [Devanbu and Litman, 1991; Devanbu and Litman, 1996]. Plans in CLASP are described as possible sequences of actions by means of regular expressions over action concepts. Scenarios, or plan instances, are sequences of action instances.

CLASP uses CLASSIC [Borgida *et al.*, 1989] to model action concepts in the style of STRIPS [Fikes and Nilsson, 1971]. Referring to Figure 6.1 on page 216, a generic `Action` can be specialized to a `System-Act` whose `ACTOR`s are restricted to `System-Agent`s, and further specialized to a `Connect-Dialtone-Act`. Devanbu and Litman note that "the system performs a Connect-Dialtone-Act and generates a dialtone after a user picks up a phone" [Devanbu and Litman, 1991]. CLASSIC is used in a similar manner to model individual actions.

Plan bodies are expressed via `PLAN-EXPRESSION`s which are extended regular expressions composed from action concepts (rather than mere symbols) using SEQUENCE, LOOP, and OR operators. Plans may optionally specify `INITIAL` and `GOAL` conditions by means of roles whose value restrictions are standard concepts. The generic CLASP plan specifies a loop with any number of generic `Actions`. It is defined at the top of Figure 6.2 on page 217.

CLASP's plan expressions add several convenient constructs to standard regular expressions. Plans may include conditional TEST expressions whose alternatives are guarded by state concepts. Macro expansion of subplans is provided via the `SUBPLAN` operator. REPEAT enables finite repetition of plan expressions.

Figure 6.2 also shows a POTS (Plain Old Telephone Service) plan and a subplan thereof. [Devanbu and Litman, 1991] paraphrases the POTS plan as: "Informally, `Pots-Plan` describes a plan in which the caller picks up a phone, gets a dialtone,

[3] Not to be confused with the homonymous CLASP system of [Yen *et al.*, 1991] to be discussed in Section 6.5.

```
(DEFINE-CONCEPT Action
    (PRIMITIVE
         (AND Classic-Thing
                (AT-LEAST 1 ACTOR)
                (ALL ACTOR Agent)
                (ALL PRECONDITION State)
                (ALL ADD-LIST State)
                (ALL DELETE-LIST State)
                (ALL GOAL State))))

(DEFINE-CONCEPT System-Act
    (AND Action
           (ALL ACTOR System-Agent)))

(DEFINE-CONCEPT Connect-Dialtone-Act
    (AND System-Action
           (EXACTLY 1 PRECONDITION)
           (ALL PRECONDITION
                 (AND Off-Hook-State
                        Idle-State))
           (EXACTLY 1 ADD-LIST)
           (ALL ADD-LIST Dialtone-State)
           (EXACTLY 1 DELETE-LIST)
           (ALL DELETE-LIST Idle-State)
           (EXACTLY 1 GOAL)
           (ALL GOAL
                 (AND Off-Hook-State
                        Dialtone-State)))))
```

Figure 6.1: CLASP Actions (from [Devanbu and Litman, 1991])

and dials a callee. If the callee's phone is on-hook, the call goes through; if the callee's phone is off-hook, the caller gets a busy signal, hangs up, and is disconnected."

An individual plan in CLASP, called a scenario, consists of a sequence of individual actions intended to achieve a goal state. CLASP uses the add and delete

```
(DEFINE-PLAN Plan
    (PRIMITIVE
        (AND Clasp-Thing
             (AT-LEAST 1 ACTOR)
             (ALL INITIAL State)
             (ALL GOAL State)
             (EXACTLY 1 PLAN-EXPRESSION)
             (ALL PLAN-EXPRESSION (LOOP Action)))))


(DEFINE-PLAN Pots-Plan
    (AND Plan
         (ALL PLAN-EXPRESSION
              (SEQUENCE
                 (SUBPLAN Originate-And-Dial-Plan)
                 (TEST
                     (Callee-On-Hook-State
                      (SUBPLAN Terminate-Plan))
                     (Callee-Off-Hook-State
                      (SEQUENCE
                          Non-Terminate-Act
                          Caller-On-Hook-Act
                          Disconnect-Act)))))))


(DEFINE-PLAN Originate-And-Dial-Plan
    (AND Plan
         (ALL PLAN-EXPRESSION
              (SEQUENCE
                 Caller-Off-Hook-Act
                 Connect-Dialtone-Act
                 Dial-Digits-Act))))))
```

Figure 6.2: CLASP Plans (from [Devanbu and Litman, 1991])

lists of action concepts to verify that a scenario is well-formed, i.e., the sequence of individual actions will transform the given initial state into the given goal state.

Internally, CLASP represents plans as extended finite automata. Results from automata theory are employed to determine plan subsumption and instantiation. One plan subsumes another if the automaton representing the first plan accepts whenever the automaton representing the second one does. Note that transitions are matched via subsumption and instantiation of action concepts. See [Devanbu and Litman, 1991; Devanbu and Litman, 1996] for further details. Regular expression subsumption is P-SPACE hard [Wellman, 1990]. A scenario is recognized as an instance of a plan if and only if the scenario's individual action sequence constitutes a string recognized by the plan's automaton. CLASP's recognition algorithm uses time proportional to the size of the plan's automaton multiplied by the length of the scenario's action sequence.

T-REX's plan subsumption is similar in spirit to that of CLASP [Devanbu and Litman, 1991]. However, by using Allen's temporal logic, T-REX supports concurrent actions that overlap in various ways. T-REX also captures finer sequential relations than CLASP, which, for example, makes no distinction between *before* and *meets*. CLASP has no metric temporal constraints, nor does it have co-reference constraints between actions. In [Devanbu and Litman, 1991], a plan instance with n steps can only instantiate plans with exactly n steps. That is, CLASP does not match an individual with a plan description when the individual adds steps not mentioned in the plan. Our system has no such restriction. Finally, T-REX plan networks can be composed nicely from binary constraints, making for a compact and facile notation. Regular expressions are comparatively unwieldy monolithic structures. Although partial orders may be captured in CLASP using disjunction, the expressions will not be concise in general. On the other hand, CLASP models preconditions and effects of actions and plans, and it fully supports disjunction and looping. Disjunction tends to be troublesome for matching in general, and indeed matching in CLASP is intractable. In practice, though, CLASP achieves considerable leverage from the compact representation afforded by finite state machines corresponding to the regular expressions.

Recently, PROTODL introduced a framework for extending description logic systems with customized language constructs [Borgida, 1992]. This methodology was demonstrated by reconstructing CLASP in PROTODL.

6.3.3 RAT

The RAT system [Heinsohn *et al.*, 1992a] was developed at the German Research Center for Artificial Intelligence (DFKI) as part of the WIP project on automatic generation of multimedia presentations. RAT was used to represent plans for assembling, using, maintaining, or repairing a physical device, namely an *espresso* machine. Plans in RAT are restricted to simple sequences of atomic, effectively instantaneous actions. However, RAT focuses on the representation of complex state descriptions that hold before and after each action in the sequence. RAT simulates the execution of a plan with a temporal projection algorithm that propagates the preconditions and postconditions of actions forwards and backwards along the action sequence. Thus, RAT can ensure a plan's internal consistency and also refine the intervening state descriptions insofar as possible. For the plan itself, RAT determines the *weakest precondition* and *strongest postcondition*. These could be used to classify plans by their executability or goals, respectively.

In RAT, actions are defined by triples consisting of (1) a conjunctive set of attribute restrictions that constitute formal parameters, as well as (2) preconditions and (3) postconditions. Both preconditions and postconditions are conjunctions of attribute restrictions, as well as agreements and disagreements (constraints across roles with equality and inequality operators). The following example is given in [Heinsohn *et al.*, 1992a], where periods denote role composition in role chains:

turn-switch-to-espresso =
<(agent: person $\sqcap$ machine: espresso-machine),
(machine.state: (off $\sqcap$ ready) $\sqcap$ machine.has-switch.position: off-position),
(machine.state: on $\sqcap$ machine.has-switch.position: espresso-position)>.

Heinsohn et al. give this gloss: "…the action TURN-SWITCH-TO-ESPRESSO has two action parameters AGENT and MACHINE, the precondition is that the switch is in the 'off' position and that the machine is off and ready, and the postcondition is that the switch is in the 'espresso' position and the machine is running."

Plans in RAT are defined by (1) a set of parameters, (2) an action sequence, and (3) equality constraints among the plan's parameters and constituent actions. These are illustrated in a plan for making espresso taken from [Heinsohn *et al.*, 1992a]:

make-espresso =
<(agent:person $\sqcap$ object1:cup $\sqcap$ object2:espresso-machine),
(…,
A5: put-cup-under-water-outlet,
A6: turn-switch-to-espresso,
…),
(object2 $\stackrel{\downarrow}{=}$ A5.machine $\sqcap$ object2 $\stackrel{\downarrow}{=}$ A6.machine $\sqcap$ …)>.

The mechanism for handling equality and inequality constraints in RAT is not discussed [Heinsohn *et al.*, 1992a].

In sum, RAT offers a detailed treatment of state information with respect to actions and plans, but only in the context of simple action sequences. T-REX by contrast, does not consider state information but offers a very rich temporal language for composing actions.

	SUDO-PLANNER	RAT	CLASP	T-REX
Application	plan synthesis	multimedia explanation	information retrieval	plan recognition
Temporal Language	none	simple sequences	regular expressions	qualitative/metric constraint networks
Concurrent Actions	n/a	no	no	yes
Disjunction	no	no	yes	no
Repetition	no	no	loop	(arbitrary, single action)
Subplans	no	no	yes	yes
Co-reference Constraints	no	equality, inequality	no	equality
Plan Instances	no	no	number of actions must agree w/ plan	unrestricted
Propagation of State Information	no	yes	yes	no

Table 6.1: Implemented Plan-based Systems in Description Logic

6.3.4 Implemented Plan Subsumption Systems

A concise comparison of the implemented systems for plan-based reasoning in description logic that we have described earlier in section 6.3 is found in Table 6.1.

6.3.5 Artale and Franconi

Recent theoretical work by Artale and Franconi provides a description logic for actions occurring over time [Artale and Franconi, 1993; Artale and Franconi, 1994]. Like T-REX, their language supports both elementary actions and complex actions (i.e., plans) at the concept and individual levels. Descriptions of complex actions may include qualitative temporal constraints over intervals of actions and equality constraints among attributes of constituent actions. Their language also has disjunction and plan parameters, but lacks metric constraints. While T-REX layers a temporal / equality plan language on top of a language for actions and related concepts, Artale and Franconi propose a single language that encompasses actions, states, and plans occurring over time. Their sound, complete, and decidable language has a formal syntax and semantics.

In Artale and Franconi's language, temporal concept expressions are formed by introducing explicit temporal variables with a temporal existential quantifier, $\diamond$. Temporal variables give rise to nodes in a temporal network, as in T-REX. Each temporal variable may be related, directly or indirectly, to a distinguished variable, $\sharp$, which represents the interval when the concept holds. Their view of when a complex action occurs differs from ours. Consider this simple cooking plan in T-REX [Weida and Litman, 1992]:

(defplan BOIL-SPAGHETTI
  :steps ((s1 C-MAKE-SPAGHETTI)
    (s2 C-BOIL))
  :qualitative-constraints ((s1 *before* s2)))

In our view, this plan occurs over the interval from the start of step *s1* through the end of step *s2*. Artale and Franconi permit an expression equivalent to the T-REX plan:

$$\diamond(x\ y)\ (x\ \mathtt{b}\ y).(\mathtt{Boil@}y\ \sqcap\ \mathtt{Make-Spaghetti@}x)$$

The temporal constraint $(x\ \mathtt{b}\ y)$ says that the temporal interval x is *before* the temporal interval y. However, they also write a similar plan thusly[4]:

$$\begin{aligned}&\mathtt{Boil-Spaghetti} \doteq\\ &\quad \diamond x\ \ (x\ \ \mathtt{b}\ \ \sharp).(\mathtt{Boil}\ \ \sqcap\ \ \mathtt{Make-Spaghetti@}x)\end{aligned}$$

According to the $(x\ \mathtt{b}\ \sharp)$ constraint, the interval when `Make-Spaghetti` occurs, x, is before the single interval when both `Boil` and `Boil-Spaghetti` occur, $\sharp$. Thus,

[4]Although Artale and Franconi use the same name, this plan is not equivalent to the T-REX BOIL-SPAGHETTI plan discussed earlier in this section.

for Artale and Franconi, the time interval of a complex action need not include the time intervals of the constituent actions.

Artale and Franconi's effort to fully integrate complex actions within a standard description logic is laudable. To accomplish this, they must define conjunction of arbitrary complex actions. Their treatment of action conjunction presumes that the conjuncts are always entirely unrelated. For example, consider conjoining a pair of identical descriptions like `Boil-Spaghetti` above. One reasonable outcome would contain a single `Boil-Spaghetti` action. However, because their language does not allow one to specify the unification of temporal variables, the only possible outcome is two unrelated `Boil-Spaghetti` actions which entail two distinct `Boil` actions and two distinct `Make-Spaghetti` actions.

While the language of [Artale and Franconi, 1994] allows for disjunction, their subsumption algorithm requires normalized descriptions, and their normalization process may cause the size of a description to grow exponentially. Briefly, their *disjunctive normal form* amounts to a disjunction of non-disjunctive temporal constraint networks, where arcs denote non-disjunctive temporal constraints and nodes are atemporal, non-disjunctive concepts.

Artale and Franconi's process of temporal completion has a significant restriction. They rely on an "= collapsing" step: whenever two nodes are temporally related by Allen's "=" relation[5], the associated concepts are conjoined. Thus, they presume that only one (possibly conjoined) action takes place over a given interval. This seems artificial, particularly since they allow two distinct actions on nearly equal overlapping intervals.

The description of a complex action may include state descriptions associated with temporally related intervals. (T-REX can do this much, too.) Artale and Franconi do not address the more difficult part of reasoning about states in plans, namely figuring out how to correlate state descriptions across intervals and in relation to the semantics of actions. For example, when two state descriptions

[5]Written *equals* in Figure 2.5 on page 39.

overlap, are they consistent? Limited, yet computationally daunting ideas have been proposed in temporal planning work, e.g., [Allen, 1984; Allen, 1991]. State information can be handled more effectively when plans are limited to description of action sequences [Devanbu and Litman, 1991; Heinsohn *et al.*, 1992a].

6.3.6 Varieties of Plan Subsumption

In our work, we have applied constraint network subsumption to plan subsumption by means of plan bodies. However, there are other ways to view plan subsumption. Several different types of plan subsumption relations are identified in [Heinsohn *et al.*, 1992a]:

Abstraction-subsumption: This is defined by [Heinsohn *et al.*, 1992a] as "plan *P1* subsumes plan *P2* iff for each action in *P1* there is an action in *P2* that is more specialized." It is the type of subsumption computed by T-REX, with the understanding that *P1*'s temporal and equality constraints are also respected in *P2*. Abstraction subsumption is also used in CLASP [Devanbu and Litman, 1991] and SUDO-PLANNER [Wellman, 1990]. Heinsohn, et al., note that abstraction-subsumption is appropriate for maintaining a database of plans at varying levels of abstraction and for conveying operating instructions for devices at varying levels of detail.

Goal-subsumption: Here, plans are ordered by subsumption relations among their goal descriptions. Goal subsumption is appropriate for plan retrieval: to achieve certain conditions, one can seek to execute some plan whose goal is subsumed by those conditions. For example, in [Swartout and Neches, 1986], plans are classified and retrieved according to goals that are represented as standard description logic concepts.

Applicability-subsumption: In this case, plan *P1* subsumes plan *P2* just in case *P1* is applicable whenever *P2* is. Heinsohn, et al. observe that the notion of applicability-subsumption is approximated when *P2* has both stronger preconditions and weaker postconditions than *P1*. This criterion is inexact because it ignores the possibility of "protected formulae" such that an inconsistent state occurs in *P1* even though *P2* can be executed successfully.

Another type of subsumption based on goals, called *g-subsumption*, was identified in [Devanbu and Litman, 1991]: the initial and goal conditions of the subsumee must satisfy those of the subsumer.

Not surprisingly, the appropriate criterion for plan subsumption depends on the application.

6.4 Temporal Concepts

Several authors have proposed concept languages with temporal concepts and individuals. We will review the work of Schmiedel [Schmiedel, 1990], Bettini [Bettini, 1994], and Lambrix and Rönnquist [Lambrix and Rönnquist, 1993].

6.4.1 Schmiedel

Schmiedel [Schmiedel, 1990] has described an ambitious attempt to extend description logic to temporal concepts by integrating both Allen's temporal logic and Shoham's [Shoham, 1987]. Standard, atemporal concepts are viewed as functions from intervals to sets of individuals, e.g., the extension of the following concept varies over time[6]:

car-owner := (at-least 1 (and own (range car)))

Schmiedel's interval-based temporal description logic adds several temporal constructs to a standard description logic language. In his examples, he uses a variety

[6] We cite Schmiedel's examples verbatim. Although his language differs slightly from ours, the examples should be clear enough in context.

of temporal constants enclosed by single quotes. The *at* construct qualifies the time interval during which an expression holds, e.g., if 'August 1990' is an absolute temporal interval constant, one can write:

(at 'August 1990' car-owner)

The distinguished interval NOW represents the time at which an expression is evaluated. Schmiedel also introduces temporal variables through an existential temporal quantifier *sometime*[7] and a universal temporal quantifier *alltime*. Using qualitative temporal constraints over intervals, it is possible to define concepts such as:

former-car-owner := (sometime (x)
(before x NOW)
(at x car-owner))

The language also supports duration constraints on intervals, e.g., owning for 35 days:

(sometime (x)
(> x '35 days')
(at x own))

Granularity predicates impose calendar restrictions on intervals, e.g., x and y are consecutive calendar days in the following:

(and (day x) (=x '24h')
(day y) (=y '24h')
(meets x y))

[7] Equivalent to the $\diamond$ quantifier of Artale and Franconi.

Schmiedel does not develop the use of Shoham's work very far, so we will not elaborate on that portion of his work. Schmiedel's logic is very expressive, and in fact undecidable [Artale and Franconi, 1994]. We believe that undecidable languages run counter to the spirit of description logic, but to be sure, Schmiedel's goal was merely to "demonstrate the expressive potential of the formalism." Although he offers no algorithm, Schmiedel does suggest a few "preliminary hints" (his words), including a definition of subsumption that corresponds to ours. His work did not consider temporal constraint networks as first class entities to be reasoned with in their own right. Schmiedel did not handle general metric constraints, nor did he address either recognition or the notion of potential instantiation.

6.4.2 Bettini

Following Schmiedel's work, Bettini has described a family of variable-free description logics with existential and universal temporal quantifiers, $\Diamond$ and $\Box$ [Bettini, 1994]. Since there are no explicit variables, Bettini's temporal expressions can only relate pairs of implicit temporal variables: a reference interval and a "current" interval.

For example, the following expression taken from [Bettini, 1994] "denotes individuals that will be ENGINEER in some future interval and that will remain ENGINEER forever," assuming that NOW is its reference interval:

$$\Diamond(after)\ (\text{ENGINEER}\ \sqcap\ \Box(met-by, after).\text{ENGINEER})$$

The future interval introduced by the outer $\Diamond$ becomes the reference interval for the interval of the nested expression introduced by the $\Box$.

Bettini claims that the variable-free approach is a more natural way of extending description logics than Schmiedel's. Indeed, it is apparent from discussions at the DL'95 workshop[8] that some members of the description logic community

[8] Held in Rome, Italy, June 1995.

believe variable-free languages to be an essential characteristic of description logics. Bettini's contributions include formalizing his interval-based temporal concept descriptions by translating them into first order logic.

6.4.3 T-LITE

Lambrix and Rönnquist integrated a novel temporal logic called LITE (Logic Involving Time and Evolution) [Rönnquist, 1990] with a terminological logic in a framework they named T-LITE [Lambrix and Rönnquist, 1993]. LITE supports time-dependent versions of individuals, and allows concepts to be defined according to the progression of individuals over versions, which are partially ordered. For example, one can describe "a particular version of (an individual named) PETER who owns a red car that changes to green and he owns no other cars." There are also plan-forming operators, corresponding to a portion of CLASP [Devanbu and Litman, 1991], that can be used to reason about recurrence. These operators can be used to describe traffic lights which continuously cycle from green to yellow to red and back to green, as well as safe intersections where no two crossing lanes are green at the same time. Lambrix and Rönnquist give a formal semantics for T-LITE, but do not address inferences such as subsumption.

6.5 Production Rules in Description Logics

The CLASP system of [Yen *et al.*, 1991][9] reasons with production rules whose antecedents are expressed using LOOM [MacGregor, 1991b]. In particular, the antecedents of CLASP rules are patterns composed of unary predicates (corresponding to concepts) and binary predicates (corresponding to roles). CLASP is concerned in part with computing subsumption relationships among the antecedents of a set of production rules and classifying the rules accordingly. Besides being valuable from

[9]Not to be confused with the homonymous CLASP system of [Devanbu and Litman, 1991] that was discussed in Section 6.3.2.

a knowledge engineering perspective, the resulting rule taxonomy provides a principled basis for selecting rules to fire under the commonly used specificity criterion. One rule is more specific than another just in case its antecedent is subsumed by the other rule's antecedent. This method compares favorably with ad-hoc specificity measures used in popular production systems such as OPS5 [Brownston *et al.*, 1985]. We observe that the constraint network subsumption task we face is rather like the one described in [Yen *et al.*, 1991]. Antecedents of CLASP rules can be viewed as constraint networks, where the unary predicates are nodes and the binary predicates are arcs. Our algorithm for QME constraint network subsumption is similar in some important respects to the existing algorithm of [Yen *et al.*, 1991].

6.6 Deductive Plan Recognition

This section recounts work in deductive plan recognition, an area of research originated by Kautz [Kautz, 1991b]. In an effort to identify tractable aspects of deductive plan recognition, Vilain later cast portions of Kautz's work in a parsing framework [Vilain, 1990]. Song and Cohen studied the use of deductive plan recognition to extract temporal information from natural language discourse [Song, 1991; Song and Cohen, 1991]. We will consider each of these efforts, in turn.

6.6.1 Kautz

Kautz's landmark work produced a formal, deductive theory of plan recognition based on circumscription [McCarthy, 1980]. His work boasts an expressive plan representation centered on a hierarchy of event types which are arranged according to *abstraction* and *decomposition* relations. Kautz's events include atomic actions as well as composite plans. A sample hierarchy from [Kautz, 1991b] is diagrammed in Figure 6.3 on page 230, where wide grey arrows indicate abstraction and thin black arrows indicate decomposition. For example, *PrepareMeal* is an ab-

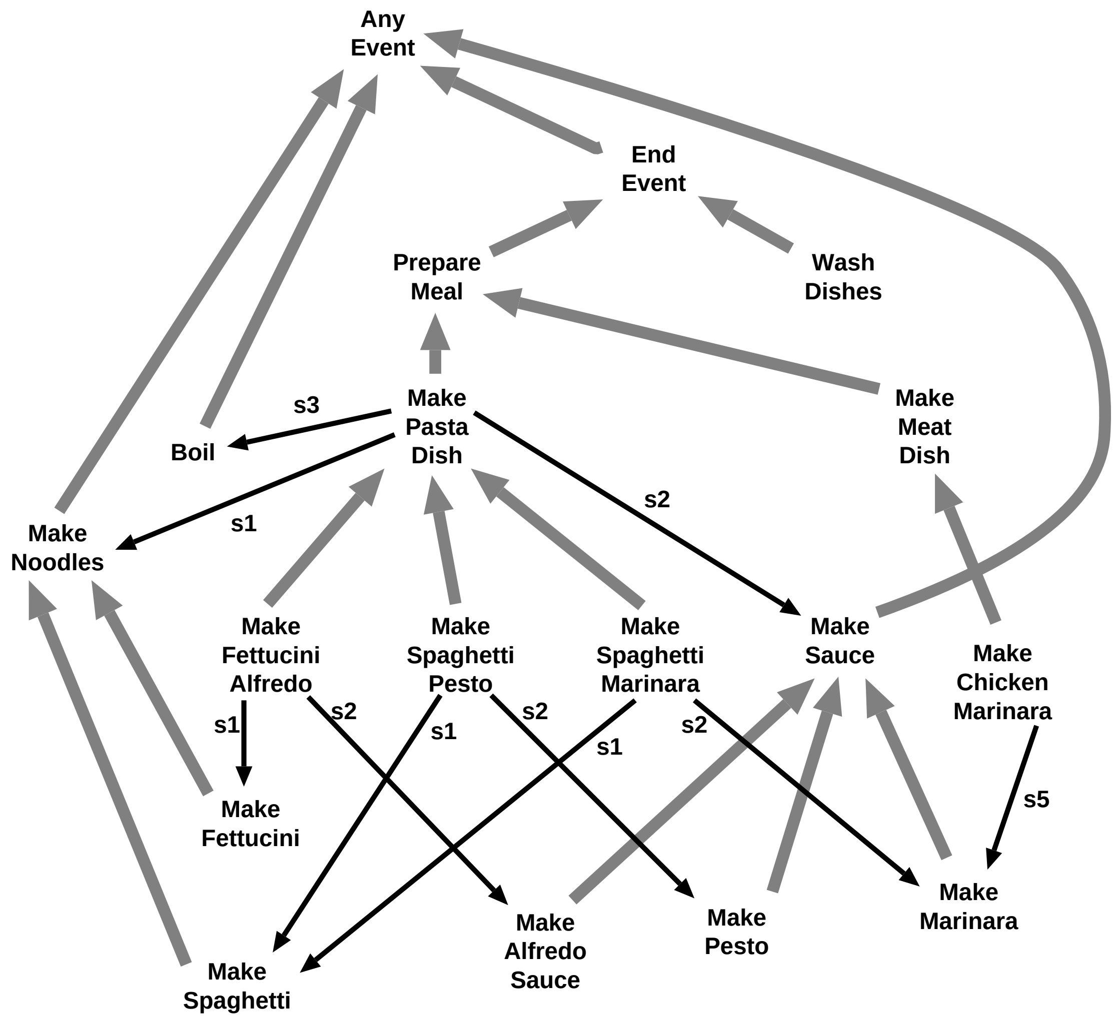


Figure 6.3: Plan Hierarchy (after [Kautz, 1991b])

straction of *MakePastaDish* and *MakePastaDish* is decomposed into *MakeNoodles*, *MakeSauce*, and *Boil* steps arbitrarily labeled *s1*, *s2*, and *s3*, respectively. Steps may be shared via inheritance, e.g., *MakeFettuciniAlfredo* inherits the *Boil* step of *MakePastaDish*.

Although not shown in Figure 6.3, Kautz's plans have parameters, equality constraints, temporal constraints, preconditions, and effects. A definition for *MakePastaDish* taken from [Kautz, 1991b] appears in Figure 6.4. Abstraction relations are stated via universally quantified implication, e.g.:

$$\forall x. MakeFettuciniAlfredo(x) \supset\ MakePastaDish(x)$$

$$\forall x. MakePastaDish(x) \supset\ PrepareMeal(x)$$

A distinguished member of the hierarchy named *EndEvent* explicitly abstracts those plans that represent independently meaningful courses of action. It is assumed that non-end plans only occur as steps of other plans. For example, Kautz's MAKE-SPAGHETTI-PESTO plan is an end plan which contains two (atomic) non-end steps: MAKE-SPAGHETTI and MAKE-PESTO.

Kautz formally characterized plan recognition as a deductive process which relies entirely on the event hierarchy, the observations, and several simplifying assumptions. These assumptions are as follows:

- **Exhaustiveness Assumptions** hold that all (relevant) types of events are known and all ways of specializing an event type are known. In Kautz's sample hierarchy, a *MakeNoodles* observation implies either *MakeSpaghetti* or *MakeFettucini.*

- **Disjointness Assumptions** hold that two types of events are *compatible* just in case one event abstracts the other, or they both abstract a common type of event. For example, *MakePastaDish* is compatible with *MakeSpaghettiPesto* but not *MakeMeatDish.*

- **Component Use Assumptions** maintain that the occurrence of one event implies the occurrence of some other event that contains the first as a component. For example, *MakeSpaghetti* implies either *MakeSpaghettiPesto* or *MakeSpaghettiMarinara.* Likewise, *MakeMarinara* implies *MakeSpaghetti-Marinara* or *MakeChickenMarinara.*

- **Minimum Cardinality Assumptions** are used to combine the implications of separate observations by assuming the smallest consistent number of distinct end events. For instance, an observation of *MakeSpaghetti* together with an observation of *MakeMarinara* implies a single end event,

$\forall x. MakePastaDish(x) \supset$

	MakeNoodles(step1(x)) $\wedge$
Components	*MakeSauce(step2(x))* $\wedge$
	Boil(step3(x)) $\wedge$
Equality	*agent(step1((x))* = *agent(x)* $\wedge$
Constraints	*result(step1(x))* = *input(step3(x))* $\wedge$
Temporal	*During(time(step1((x))), time(x))* $\wedge$
Constraints	*BeforeMeets(time(step1((x))), time(step3(x)))* $\wedge$
	Overlaps(time(x), postTime(x)) $\wedge$
Preconditions	*inKitchen(agent(x), time(x))* $\wedge$
	Dextrous(agent(x)) $\wedge$
Effects	*ReadyToEat(result(x),postTime(x))* $\wedge$
	PastaDish(result(x))

Figure 6.4: A Kautz Plan (from [Kautz, 1991b])

namely *MakeSpaghettiMarinara*. Without the minimum cardinality assumption, many other conclusions could be drawn, such as separate *MakeSpaghettiPesto* and *MakeChickenMarinara* plans.

Kautz's logical characterization of completeness assumptions in plan recognition are a major contribution in their own right.

Kautz uses circumscription and a cardinality-minimization operator to select minimal models of the event hierarchy, called *covering models*, that are consistent with the observations and these assumptions. Loosely speaking, circumscribing a predicate serves to minimize its extension. Kautz first circumscribes the specializations of an event, then circumscribes its uses. Finally, covering models minimize the cardinality of *EndEvent*. In addition to Kautz's formal theory of plan recognition, he devised more practical algorithms that approximate his theory.

Our plan recognition work is most closely related to that of Kautz [Kautz, 1991b]. Like Kautz, we make deductive inferences over a plan library that incorporates both a plan abstraction hierarchy and plan decomposition into constituent actions. Both approaches are restricted compared to other techniques because they do not chain on state information (e.g. preconditions and effects), and have strong assumptions such as plan library correctness and completeness.

Our distinction between actions and plans resembles Kautz's distinction between non-end events and end events, in the sense that our plan recognition algorithm does not generally recognize actions in isolation.[10] Our plan taxonomy also supports a distinction between concrete and abstract plans which Kautz does not; recall our assumption that the observations must reflect a concrete plan, while abstract plans describe classes of concrete plans at a more general level. Kautz represented END-EVENT as a distinguished member of his event abstraction hierarchy, and required other end events to specialize it. Rather than placing an otherwise meaningless special concept in our taxonomy to represent concreteness, we model

[10] However, this can be accomplished in T-REX, when appropriate, by defining plans with only one step.

the concrete / abstract distinction as a boolean status associated with each plan.

There are several reasons to prefer T-REX to Kautz's implementation. In terms of plan representation, Kautz's system uses a temporal language for relating actions that is more restricted than the one we use via MATS (which, we hasten to add, was also implemented by Kautz). We also use an underlying description logic system, either K-REP or CLASSIC, to represent and reason with the actions and objects that are the building blocks of plans, whereas atomic actions and objects in [Kautz, 1991b] and many other approaches lack defined semantics. Thus our approach allows the plan recognition system to share the advantages of existing description logic ontologies. In contrast with Kautz's approach of manual plan library construction, we extend work in description logic to formalize and automate the organization of the plan taxonomy via subsumption and classification. Moreover, we directly exploit the library's definitional nature to guide plan recognition. Like Kautz, we permit observation of actions at an abstract level; unlike him, we also provide for revision [11] of prior observations. Kautz's implementation performs certain expensive computations at run time. It computes the possible consequences of each observed action independently and records them in separate graph structures which are then combined by repeated graph-merging operations. We prefer to precompute possible relationships among actions as reflected in the plans by constructing a definitional plan taxonomy in advance. We then determine possible consequences from the observation network as a whole, on a context-dependent basis. Of course, our approach, like that of Kautz, is NP-complete. Finally, Kautz does not explicitly distinguish between necessary and optional plans, nor does he provide support for acting on recognition results, as T-REX does with demons.

6.6.2 Vilain

Following up on a proposal of earlier authors, e.g., [Sidner, 1985], Vilain carefully examined the idea that plan recognition has much in common with parsing

[11] Monotonic update, in the current implementation.

of text [Vilain, 1990]. The analogy makes these correspondences:

Plan Recognition	**Parsing**
plan library	grammar
plan description	grammar rule
observed steps	lexical tokens

Just as grammar rules are used to parse sentences, Vilain notes that plan descriptions might be use to "parse" strings of actions. For implementation, he suggested a chart-based parser to retain a record of the plan structure presumably reflected in the observations.

The primary goal of Vilain's investigation was to identify those portions of Kautz's framework [Kautz, 1991b] that are amenable to more efficient solution through parsing. By drawing a formal correspondence between Kautz's framework and context free grammars, Vilain indeed found certain portions of Kautz's formalism that admit tractable parsing-based solutions [Vilain, 1990]. Vilain summarized his central result on tractable parsing of plans as follows [Vilain, 1990]:

> **Proposition:** There is a $O(n^3)$-time plan recognition algorithm for hierarchies with ordered, unshared steps, and for disjunctive or abstract observations.

Unfortunately, plans with greater temporal expressiveness present problems. Vilain suggests that parsing of partially-ordered plans can be handled in practice with a combination of indirect dominance (ID) rules and linear precedence (LP) rules as in Shieber's ID/LP parser [Shieber, 1983]. However, he was still led to the following conclusion [Vilain, 1990]:

> **Proposition:** Recognizing plans with abstraction and partial step order is NP-complete, regardless of recognition tactic.

Vilain was not able to find a tractable parsing solution for plan parameters, for sharing steps between plans, or for interleaving steps of different plans.

One strength which Vilain's approach has in common with T-REX, and in contrast to Kautz's work, is an ability to focus recognition by considering all of the available observations together. As noted in Section 6.6.1, Kautz conducts a separate recognition process for each observation before combining the results. Vilain remarks that "This is computationally much more onerous, but may turn out to be unavoidable if one wants to allow for sharing and interleaving of steps." His first point is correct, but his second (speculative) point is contradicted by T-REX's approach to simultaneous plans, described in Section 5.11.

6.6.3 Song and Cohen

Song and Cohen have considered how to extract the intended temporal relations among situations described in natural language discourse [Song, 1991; Song and Cohen, 1991]. They called this the *temporal analysis* problem. Song and Cohen were motivated by the idea of a natural language interface to a plan recognition system. Their system, like ours, employs Allen-style temporal reasoning and can eliminate plans in the plan library that are inconsistent with the extracted temporal relations. Also, the extracted relations can be used to make prestored relations in the plan library more specific. Furthermore, based on a complete library assumption, the recognized plans may yield necessary constraints that further refine the extracted relations. This sounds similar to our use of the LCS inference to extract constraints on a configuration in Section 3.8. However, no details on this are reported. It is not clear how or even if they perform this refinement based on the intersection of more than one candidate plan, nor have they discussed the possibility that observations may match a single plan in more than one way i.e., multiple partial instantiation mappings in our framework.

Song and Cohen also presented an algorithm to infer strong qualitative temporal constraints between a plan and its substeps [Song and Cohen, 1991]. Suppose that a plan consists of two unconstrained substeps. Then, we can conclude that the relation of each substep to the plan itself is confined to {*starts, during, finishes, equals*}. However, we can often do better if we have information about the temporal relations among the substeps. The idea is to view the plan as a hierarchical structure as well as a temporal network. For example, if a plan has two substeps and one is *before* another, then the first necessarily *starts* the plan and the second necessarily *finishes* it. In this vein, Song and Cohen give an algorithm to strengthen the temporal constraints for plans with two substeps. They go on to show how it can be iterated to strengthen a decomposition with any number of substeps. This process is carried out repeatedly, in alternation with Allen's constraint propagation procedure, until reaching a fixpoint. We have implemented this procedure in T-REX.

The plan recognition part of Song and Cohen's system lacks many capabilities found in T-REX. While we shall now mention some of these limitations for the sake of contrast, we hasten to add that their work was largely concerned with the temporal analysis problem, where they made valuable contributions unrelated to plan reasoning. That said, their plan representation employs undefined, atomic actions and it does not support metric temporal constraints. Their system cannot compare plans with respect to generality or classify them; indeed, their plans are not organized into an abstraction taxonomy. Thus, from the standpoint of practical performance, they are unable to guide their search accordingly. From the standpoint of knowledge engineering, the relationship among their plans is obscured, especially in the case of large plan libraries. Their observations may not include abstract actions, hence refinement of observed actions is precluded, as is retraction of observations. Song and Cohen's plan recognition process only identifies possible plans, not necessary ones. Finally, they have not considered the prospect of simultaneous plans.

6.7 Candidate Elimination

In the area of machine learning, Mitchell investigated the problem of *generalizing* from examples [Mitchell, 1982]. There are some interesting similarities with our work on predictive recognition.[12] Mitchell summarized a class of generalization problems as follows (taken from [Mitchell, 1982]):

Given:

1. A language in which to describe instances.
2. A language in which to describe generalizations.
3. A matching predicate that matches generalizations to instances.
4. A set of positive and negative training instances of a target generalization to be learned.

Determine:

- Generalizations within the provided language that are consistent with the presented training instances (i.e., plausible descriptions of the target generalization).

Given an expressive generalization language, the *hypothesis space* of generalizations that are consistent with the known instances may be enormous. Noting that generality induces a partial order on the hypothesis space, Mitchell proposed a *version space* strategy to track the hypothesis space as instances are presented to the system. Two sets are used to capture it in a space-efficient manner (taken from [Mitchell, 1982]):

S = { s | s is a generalization that is consistent with the observed instances, and there is no generalization which is both more specific than s, and consistent with the observed instances }.

[12] Thanks to Bonnie Webber and Ehud Reiter who independently pointed out the similarity.

> G = { g | g is consistent with the observed instances, and there is no generalization which is both more general than g, and consistent with the instances }.

These sets are sufficient to bound the hypothesis space, as Mitchell says:

> A generalization, x, is contained in the version space represented by S and G
>
> if and only if
>
> 1. x is more specific than or equal to some member of G, and
> 2. x is more general than or equal to some member of S.

Mitchell also gave a *candidate-elimination* algorithm to update the hypothesis space as instances are presented. Our predictive recognition strategy bears a resemblance to the version space strategy: we also use sets to track the upper and lower bounds of a space containing concepts that may or may not ultimately turn out to be consistent with future input. Our work differs from the version space strategy in several key respects:

1. In the version space scenario, the space of generalizations is implicit in the generalization language and initially encompasses every generalization that the language can express. In our work, the space of generalizations is explicit in the concept taxonomy and is initially limited to just those concepts, i.e., an individual must ultimately be *finished* as in Definition 9 on page 52.
2. The candidate-elimination algorithm is given explicit negative examples that are used to exclude concepts, while our plan recognition algorithm derives the concepts to be excluded using the CTA.
3. Candidate-elimination operates on sets of different positive and negative instances, while we are concerned with successive descriptions of the same instance.

In work closely related to plan recognition, Lesh and Etzioni investigated the problem of recognizing a user's goal from a sequence of actions [Lesh and Etzioni, 1995]. Their sound and fast, but incomplete, goal recognizer uses a version space algorithm to track the space of goal schemas that are consistent with the observed actions. By maintaining a set of most specific goal schemas and a set of most general goal schemas, they need not enumerate every goal in the version space.

6.8 Intentional Plan Recognition

There have been many approaches to plan recognition that reason about the intentions of agents via precondition and effects (or goals) of actions and plans, e.g., [Allen and Perrault, 1980; Carberry, 1990; Cohen and Levesque, 1990; Litman and Allen, 1987; Pollack, 1990; Sidner, 1985]. This body of work emphasizes plan inference using state information as well as action decomposition. It is more comprehensive, but less formal than our work or that of [Kautz, 1991b], [Vilain, 1990], and [Song, 1991]. For reference, discussions of intention-based plan recognition are contained in [Carberry, 1990; Kautz, 1991b; Song, 1991].

Chapter 7

Conclusion

This chapter recaps the major contributions of the thesis and surveys some possible directions for continuing the work in the future.

7.1 Contributions

The main contributions of this thesis are as follows.

7.1.1 Predictive Recognition in Description Logic

Our work on predictive concept recognition has resulted in a whole new way of using description logics and encompasses several significant developments:

- This thesis introduced an explicit distinction between the knowledge engineering and problem solving phases of terminology usage in description logic, for those applications where the terminology remains fixed during problem solving. The notion of a *closed terminology assumption* was defined to make this distinction precise. Configuration was identified as an important problem where CTA applies during problem solving, and indeed fosters important new inferential services.

- Rigorous criteria were stated for judging whether an individual's description is sufficiently complete and specific for solving the problem at hand. For applications such as configuration, where the solution to a problem is expressed in terms of incrementally specified individuals, it is essential to know when an individual's specification can be considered done. No previous work in description logic has touched on this vital question. To help decide if an individual's description is specific enough, a distinction between *abstract* and *concrete* concepts was introduced to description logic. To help evaluate the description's completeness, a new *bijective instantiation* inference was created. Using these ideas, along with the CTA, the notion of a *finished* individual was precisely specified.

- *CTA-consistency* inferences were formulated to determine *potential instantiation*, i.e., whether a given partially described individual can ultimately instantiate a given concept under CTA. These include both *direct CTA-consistency* inferences that compare the individual to the concept, and *indirect CTA-consistency* inferences that account for the presence of other concepts in the terminology. Those inferences rely in turn on direct and indirect CTA-consistency inferences between pairs of concepts. The set of concepts that are CTA-consistent with an individual precisely captures possible future choices for further specifying the individual, given the closed terminology. In contrast, OTA-consistency is a much weaker inference that ignores the closed nature of some terminologies, and instead always assumes that *all* logically consistent concepts may exist.

- A *predictive concept recognition* methodology for description logic was developed to track the set of concepts that an individual may eventually instantiate, as its description is incrementally updated. By combining CTA with a provisional assumption of monotonic updates to an individual, it was shown how to decide if a concept is *necessary* (already instantiated by the individual), *optional* (potentially instantiated by the individual in the future), or *impossible* (ruled out). This partition of the terminology by modality, which we have identified, informs both the user and the application program, e.g., a configuration engine. CTA enhances the resulting information by greatly increasing our ability to rule out potential instantiations and thereby categorize concepts as impossible.

- To speed the recognition process, a *terminology augmentation* procedure was given. This obviates run-time searching for concepts that sanction indirect consistency inferences. Instead, indirect consistency inferences are made by simply traversing explicit subsumption links in the concept taxonomy.

- An *incremental partitioning* algorithm was designed and implemented to exploit the terminology's subsumption-based organization during predictive concept recognition. In particular, the current partition of the terminology with respect to an individual is used to compute the next partition whenever the individual is updated. In this process, the consequences of comparing an individual with a concept are propagated to other concepts, so those other concepts need not be directly compared with the individual. The state of the partition is concisely captured by sets of *most specific necessary* (MSN) concepts, *most general optional* (MGO) concepts, and *most specific optional* (MSO) concepts. In the context of an interactive system, e.g., for configuration, information about the current partition can be used to inform both the user and the application system. For example, the MGOs highlight the most general choices available in the current context.

- A *constraint derivation* technique was presented to further exploit the partition of a terminology through novel use of the least common subsumer inference [Cohen *et al.*, 1992]. Restrictions common to an appropriate set of optional concepts are propagated to the individual in question, perhaps yielding crucial new restrictions that would not be known otherwise.

- The preceding techniques have been incorporated in a version of the K-REP system. They have been tested with a pilot configuration knowledge base that was developed by others. Experimental results show that performance is easily fast enough for interactive applications.

In short, whenever CTA is appropriate, CTA reasoning enables a description logic system to support an application system with an array of useful inferential services that are far stronger than OTA allows.

7.1.2 Constraint Networks and Plans

A number of contributions involve representing and reasoning with constraint networks in a description logic setting:

- This thesis originated the idea of *constraint network concepts and individuals* based on description logic. It showed how to compute subsumption among constraint network concepts, and thus automatically classify them in definitional taxonomies. Recognition of constraint network individuals was handled in similar fashion. These ideas are not tied to any particular kinds of constraints. Consequently, the ability to reason with aggregations of concepts (and likewise individuals) in a description logic setting has been greatly enhanced.

- As a specific example of constraint network concepts and individuals, *QME constraint networks* were introduced, along with a language for describing them. We detailed how description logic and temporal logic can be united in a common framework, in part following initial ideas outlined by [Schmiedel, 1990]. In QME networks, nodes associate a standard description with a temporal interval to represent an occurrence in time. Qualitative and metric temporal constraints across intervals, together with equality constraints across standard descriptions, allow us to represent patterns of occurrences over time. For example, QME constraint networks engender a plan representation with extremely expressive temporal constraints that is uniquely rich in the field of plan recognition.

- The subsumption problem for constraint network concepts was proved to be NP-complete.

- A specific *algorithm for QME constraint network subsumption* was designed and implemented. It gains heuristic guidance from both concept subsumption and temporal precedence, as does a similar algorithm for plan instantiation. The subsumption algorithm is suitable for classifying descriptions of temporal patterns, e.g., plan bodies, in a strict taxonomy.

- *Subsumption-based plan libraries* were established as a powerful tool for organizing plans to be recognized. Our libraries offer important features that are unique in plan recognition:

  - Descriptions of actions and their constituents enjoy well-defined semantics due to K-REP/CLASSIC.
  - The plan taxonomy itself has a well-defined semantics and is automatically classified via subsumption.
  - Classification maintains the integrity of the taxonomy as plans are added and modified over time.
  - Classification enables a powerful pattern matching query facility for plan retrieval.
  - Redundant, inconsistent, and vacuous plan descriptions are identified by the system.
  - During knowledge engineering, *rules* based on constraint network subsumption can be used to enforce consistency with a *domain theory*.

In brief, constraint network descriptions significantly enhance the representation and reasoning capability of description logic systems.

7.1.3 Constraint Network Recognition and Plan Recognition

Another set of contributions revolves around predictive recognition of constraint network concepts:

- A mechanism for *predictive recognition of constraint network concepts* was advanced. It integrates our results on predictive concept recognition with our results on constraint network concepts and individuals.
- For constraint network descriptions in general, and QME networks in particular, *CTA-consistency* inferences were reported. As with standard descriptions, both *direct consistency* and *indirect consistency* cases were handled.
- An *augmentation* procedure for libraries of constraint network concepts was given to speed the recognition process. The process differs from terminology augmentation in standard description logic, due in part to the possibility of multiple consistency mappings between networks.
- In contrast with monotonic observation, the more restrictive case of *perfect observation* and the more general case of *unrestricted observation* were analyzed. Perfect observation permits *partial instantiation* inferences in lieu of consistency inferences; unrestricted observation requires an ability to undo conclusions about the necessity or impossibility of specific plans.
- A *deductive theory of plan recognition* in the spirit of Kautz's pioneering work [Kautz, 1991b] was proposed. This framework departs from Kautz's work in several major positive ways:
  - Recognition is guided by the plan taxonomy's subsumption-based organization.
  - Recognition proceeds from the observations as a whole, rather than computing the consequences of each observed action separately (without concern for the effect of other actions), and then combining the results.

- Observations, which are accepted in any order and at any level of abstraction, may be refined retroactively.
- The strategy for recognition of *simultaneous plans* is more informed.
- Procedures known as *demons* can be associated with plans. Demons actively respond to observations as they unfold.
- The overall recognition process is independent of plan language details.

- The generality of our predictive recognition methodology was exhibited by using the same techniques in plan recognition that we used earlier in configuration problems.

In sum, we have developed a powerful new approach to deductive plan recognition, a specific example of constraint network recognition, which takes full advantage of description logic technology and goes on to build considerably more functionality upon it.

7.1.4 Implementation of Plan-based Reasoning

The implementation of the *T-REX system* constitutes a contribution in its own right:

- T-REX clearly demonstrates the preceding contributions on plan subsumption, plan library classification, plan retrieval, and predictive plan recognition.

- T-REX further demonstrates that independently developed AI systems can be harnessed together in close cooperation to obtain a high degree of implementation synergy. Two description logic systems, either K-REP or alternatively CLASSIC, were used hand in hand with a temporal reasoner, MATS, to tackle descriptions of temporally rich patterns.

7.2 Limitations

The assumption of a closed terminology in description logic is a strong one. Indeed, the strength of this assumption directly empowers our recognition methodology to draw conclusions. Still, we must acknowledge that the closed terminology assumption is inappropriate for applications that expect to add or modify concepts during problem solving. For example, although CTA is completely appropriate for configuration systems, which consider ways to assemble existing artifacts (albeit in sometimes very complex ways), it is not suitable for *design* systems, which fundamentally address the creation of new artifacts. Likewise, the assumption of a closed plan library in plan recognition is very strong. It is appropriate when the observations will follow a plan from the plan library (which may be described only at an abstract level, allowing considerable latitude), but not for recognition of unforeseen plans.

Many other limitations of the current work can also be seen as opportunities for future work.

7.3 Future Work

There are many ways to expand on this thesis. Some of the leading opportunities are as follows.

7.3.1 Predictive Recognition in Description Logic

First we consider directions for continuing our work on predictive concept recognition in description logic:

- The CTA-consistency inferences described in this thesis are syntactic in nature, which makes them somewhat dependent on the description logic language under consideration. A formal semantics for description logic under the CTA should be developed. It should be sufficient to account for the following differences from standard description logic:

1. The extension of each abstract concept is the union of the extensions of the concrete concepts that it subsumes. Hence, it is the union of the extensions of its explicitly defined (immediate) descendants.
2. All roles not explicitly restricted by a concrete concept are incoherent (can not be filled).

Two possible frameworks that have been suggested for formalizing the CTA in description logic are type theory and circumscription.[1] There is a potentially useful analogy with Kautz's use of circumscription for closing his event hierarchy: Kautz's hierarchy is composed of events related by steps; our concept taxonomy is composed of concepts related by roles.

- Our predictive concept recognition methodology has been developed for the most fundamental concept-forming operators. Other common constructs should now be treated. So-called *host* concepts, such as strings, numbers, and intervals, seem easy to accommodate. Several kinds of role restrictions also appear straightforward, including *some* (existential role restriction) and *one-of* (a set of possible individual fillers, one of which must be an actual filler). Of particular interest for certain configuration systems are forward-chaining rules as in CLASSIC and MESON. These rules can apparently be treated as explicit declarations of CTA-consistency. That is, if individual *I* is directly consistent with concept *C1* and there exists a rule that *C1* implies *C2*, then *I* is indirectly consistent with *C2* via *C1*. Consequently, if the conjunction of *C1* and *C2* is not explicitly present in the knowledge base, it would be added during augmentation. To date, we have not been concerned with *or* (general disjunction) and *not* (general negation) operators because K-REP does not support them, but it is unclear how they might be incorporated in the present framework. Importantly, CLASSIC has achieved great

[1] Personal communication with Frank Oles and Bernhard Nebel, respectively.

success in the configuration arena without supporting either disjunction or negation [Wright *et al.*, 1993].

- The computational complexity of CTA-consistency inferences should be analyzed.

- Some consequences of the CTA become manifest as soon as the terminology is closed, independent of any problem solving context. Therefore, we should be able to condition the terminology through static analysis to make certain CTA consequences explicit prior to problem solving. As a simple example, consider the RISC-MULTIPROCESSOR-SYSTEM concept from Figure 2.2 on page 20 and imagine that it has several concrete subsumees, *each* of which happens to require exactly two processors. The definition of RISC-MULTIPROCESSOR-SYSTEM should not be specialized because other multiprocessor systems with more than two processors may be introduced in the future. Nonetheless, we might annotate it to the effect that for current problem-solving purposes, RISC-MULTIPROCESSOR-SYSTEMs do have exactly two processors. This information could be taken into account whenever RISC-MULTIPROCESSOR-SYSTEM figures in an LCS inference for constraint derivation as in Section 3.8. Furthermore, this kind of information might be propagated up the taxonomy. The general idea is to more explicitly differentiate a concept's currently effective "problem solving" definition from its timeless "knowledge engineering" definition.

- In this thesis, we have applied predictive concept recognition to the problem of computer system configuration. Our techniques should also be useful in a wide variety of "selection" applications where there is a taxonomy of concepts representing possible selections, along with associated concepts representing ways of tailoring properties of particular selections (i.e., as role value restrictions). For example, one might construct an investment terminology which describes (among other things) the thousands of existing mutual funds in terms of their objectives, investment mix, ratings, taxable status, minimum investment, and numerous other properties. Each specific mutual fund would be a concrete concept and a leaf node in the taxonomy. Internal nodes would be abstract concepts that categorize funds by the offering company, and many other well-known types such as GROWTH-AND-INCOME-FUND, INTERNATIONAL-FUND, STOCK-FUND, BOND-FUND, NY-STATE-TAX-FREE-FUND, NY-STATE-AND-CITY-TAX-FREE-FUND, etc. As with our configuration application, investors may wish to make choices in any order and at any level of abstraction, perhaps in collaboration with an advisory software system. Our methodology can effectively identify the range of known possibilities that are CTA-consistent with an investor's current choices, and perhaps infer additional constraints, as she incrementally homes in on a fund which meets her criteria.

In the same vein, one might create a pharmaceutical terminology. For example, a physician may wish to identify an appropriate drug to order for a patient. The physician can benefit from the services of our incremental recognition methodology as she incrementally specifies properties of drugs, such as conditions treated, pharmaceutic components, dosage form, route of administration, and so on. Similarly, our recognition methodology can be exploited profitably in conjunction with terminologies for camera and video equipment, automobiles, homes, commercial real estate, and many other goods and services where consumers are faced with a non-trivial selection process.

7.3.2 Constraint Networks and Plans

There is also room to improve upon our treatment of constraint networks based on description logic:

- Full integration of constraint network descriptions within a regular description logic system such as K-REP or CLASSIC would be elegant. For example, a fully compositional and integrated language would permit one to use plan concepts as value restrictions of roles, and individual plans as role fillers.

However, as discussed in Section 6.3.5, choosing the most satisfying way to handle plan conjunction will require some thought.

- The T-REX plan language could be extended in several directions, to meet the needs of future applications:

  - Plans might be given roles, just like the roles of standard concepts. Plan roles could be used to represent a plan's parameters, such as *agent*. Observe that a plan's agent need not be the agent of any step within the plan. For example, a manager may order a plan whose steps are carried out by his or her underlings. Naturally, co-reference constraints on a plan might correlate parameters of the plan with parameters of its actions.

  - To this point, we have said that the nodes of plan networks correspond to actions. We can easily extend plan networks by introducing nodes corresponding to *properties* which hold over particular time intervals. Like actions, these properties are represented by concepts. Our definitions of subsumption and potential subsumption continue to apply; due to the concept taxonomy, our procedures will only map actions to actions and properties to properties. Notice that property nodes can represent arbitrary *conditions* which generalize the notion of preconditions and effects, since they need not occur properly before and after all of the plan's actions, respectively. Instead, conditions can overlap and interleave with actions in arbitrary ways. Similarly, each action might have a set of associated conditions related to it by some temporal structure, as in [Allen, 1991]. Then we would be faced with plans that are arbitrary temporal networks of actions and conditions, and each action could have its own temporal network of associated conditions. Checking such plans for internal consistency and normalizing constraints on their conditions will be difficult problems to solve in principle. Even then, one must anticipate severe performance problems. Assuming that these problems could be addressed, we would want to study the use of state information in description logic-based plan recognition.

- For some applications it may be useful to support additional co-reference constraint operators such as inequality, subset, or superset. We might also extend the role portion of co-reference operand specifiers to permit role chains (compositions of role relations). In general, this would render the subsumption problem undecidable, for reasons similar to those in [Schmidt-Schauß, 1989], but attribute chains would be fine. Chains would increase the ability of rules to enforce integrity constraints according to a domain theory. For example, in our travel plan application, if a VISIT-CITY step occurs within a VISIT-COUNTRY step, we could ensure that *the country of the city of the* VISIT-CITY *step* is equal to *the country of the* VISIT-COUNTRY *step*.

- Each plan should implicitly contain a *self* step representing the entire plan. In our travel plan application, it would then be possible to state the overall duration of a plan, and to select plans based on their duration. Moreover, temporal constraints could be used to relate steps within a plan to the plan itself. For example, this would enable us to explicitly state that a step necessarily *starts* the plan. Additionally, Song and Cohen's algorithm for inferring constraints between a step and its substeps could be applied to the plan itself.

- Applications such as travel plan recognition would benefit from the integration of calendar time representation and reasoning, to include recurring intervals such as months and seasons of the year. Presumably this could be built on top of the absolute time representation supported by MATS.

- To support even greater temporal expressiveness in plan-subsumption systems, we might extend T-REX's language in the direction of CLASP's language [Devanbu and Litman, 1991]. Some ideas for introducing limited forms of disjunction and looping in temporal constraint networks were described in [Weida, 1993]. However, general disjunction and looping do not seem to fit naturally within the constraint network paradigm. Conversely, we might extend CLASP's language in the direction of T-REX's language. Simultaneous actions can be represented via regular expressions over sets of action extrema which correspond to time points. For example, when ACT1 and ACT2 occur over the same time interval, the regular expression would be[2]:

```
(SEQUENCE {(start act1), (start act2)}
          {(finish act1), (finish act2)})
```

If the interval of ACT1 overlaps that of ACT2, we would get this regular expression:

```
(SEQUENCE {(start act1)}
          {(start act2)}
          {(finish act1)}
          {(finish act2)})
```

Such expressions should be screened for well-formedness, e.g., for every starting point of an action, there must be a corresponding ending point that occurs later. This is easy to enforce if, e.g., the start and finish of a given action are required to be within the same non-disjunctive sequence.

Metric information can be encoded by separating the sets of time points with metric intervals. For example, consider the case where ACT1, which consumes 3 to 5 time units, precedes or follows ACT2, which consumes 7 to 9 time units:

```
(OR (SEQUENCE {(start act1)}
              [3,5]
              {(finish act1)}
              (0,∞)
              {(start act2)}
```

[2]We adopt the operators of [Devanbu and Litman, 1991] to compose regular expressions, as in Section 6.3.2.

[7,9]
{(finish act2)})
(SEQUENCE {(start act2)}
[7,9]
{(finish act2)}
(0,∞)
{(start act1)}
[3,5]
{(finish act1)}))

Subsumption between regular expressions continues to follow from subsumption between elements of the language. Subsumption of action extrema is based on action subsumption and extrema identity, e.g., (start VISIT-EUROPEAN-COUNTRY) subsumes (start VISIT-ITALY) but not (finish VISIT-ITALY). Metric constraint subsumption is done as usual. We conjecture that any QM (qualitative and metric) temporal constraint network over intervals can be encoded as a regular expression over sets of action extrema separated by metric intervals.

7.3.3 Constraint Network Recognition and Plan Recognition

Our approach to the recognition of constraint network concepts and its application to plan recognition can also be enhanced:

- Given partial and/or inexact observations of a plan in progress, T-REX may identify a set of optional plans, one of which presumably underlies the observations. In such cases, T-REX currently has no basis for preferring one plan over the others. This ambiguity might be resolved with goal-based reasoning, e.g.,[Litman and Allen, 1987], probabilistic reasoning, e.g.,[Charniak and Goldman, 1993], and/or selective querying of the user through clarification dialogues, e.g., [van Beek and Cohen, 1991]. In general, opportunities for collaboration between deductive and intentional plan recognition should be explored.

- In cases where the assumption of a single plan proves unjustified, we seek improved ways to identify sets of plans that best account for the observations. There are substantial open research questions here. For example, what principles govern possible interleavings of plans? When and how should substeps be shared among plans?

- For applications where users may carry out many plans over an extended period of time, there is a need for *continuous plan recognition*. Here, we will need to move beyond the minimum cardinality assumption, which quickly becomes inadequate. Acceptable performance might require stronger assumptions, e.g., with software interfaces we might make a *temporal progression assumption* that having observed some action instance ACT86, all subsequently observed action instances in fact *occur after* ACT86. Significant challenges also arise from the potential for very large observation networks. We would want to eliminate obsolete observations whenever possible. For example, once an action instance has been recognized as part of a certain plan, if that action instance cannot be shared with other plans, it should be pruned from the observation network. We might also want to define plans with a maximum overall duration so that we can discard potential subsumption mappings if they do not materialize within the specified period of time. This could result in failure to recognize some plan occurrences, i.e., plan recognition would be incomplete but would remain sound.

7.3.4 Generalization to N-Dimensional Space

We now sketch a possible future application of constraint network subsumption and instantiation to descriptions of spatial configurations.

Disjunctions of Allen's 13 primitives capture all possible relationships between intervals along a single dimension. While Allen's scheme was designed for the temporal domain, it is equally appropriate for one-dimensional space. Moreover, as pointed out in [Mukerjee and Joe, 1990], relationships in N-dimensional space can be modeled by N-tuples of Allen's constraints. As a first approximation to spatial relationships, we associate objects and locations with rectilinear bounding boxes aligned to the axes, i.e., we consider the projections onto the axes as intervals, and then use Allen's relations on them. The alignment can in fact be varied [Mukerjee and Joe, 1990].

The following constraint network specifies a C-SQUARE whose bounding box is disjoint from that of a C-POLYGON in 2-dimensional space[3]:

- C-SQUARE *((before, after), (before, after))* C-POLYGON

Orthogonal constraint networks maintain relationships along each axis. Constraint propagation can be applied independently in each dimension to discover, for example, that if there is an object that is properly contained in the C-SQUARE then it is spatially disjoint from the C-POLYGON.

Our idea of constraint network subsumption extends to multiple dimensions: constraint *C1* subsumes constraint *C2* iff each component of *C1* subsumes the corresponding component of *C2* as defined previously. Thus, the preceding description subsumes the following one, which says that a C-SQUARE is left of and above a C-RECTANGLE (assuming normal interpretation of the x and y axes, respectively):

- C-SQUARE *((before), (after))* C-RECTANGLE

Based on subsumption, we can automatically classify a library of such spatial descriptions.

There is a direct analogy from temporal duration to spatial extent, so the metric capability of MATS would allow us to represent and reason with extent

[3]For convenience, we dispense with labels here and just show concepts related by constraints.

in each dimension. Thus we would obtain volume for the bounding boxes. The shapes of the objects within the bounding boxes can be better modeled by K-REP concepts, which can capture ideas such as the fact that C-POLYGON subsumes C-RECTANGLE, which in turn subsumes C-SQUARE, etc.

Our formulation of potential instantiation and predictive constraint network recognition also extends readily to multiple dimensions. The following spatial concept neither subsumes, nor is subsumed by, either of the two presented above:

- C-SQUARE *((before, meets), (before, meets))* C-SQUARE

However, it does enjoy a direct CTA-consistency relationship with the first concept (in both directions). The following observation network, which consists of a single node specifying an individual rectangle, is consistent with all three spatial concepts above[4]:

- RECTANGLE86

After it is monotonically updated as follows, the observation network is only consistent with the first and third concepts:

- RECTANGLE86 *((before), (before, meets))* RECTANGLE99

A further monotonic update leaves it consistent only with the third concept:

- RECTANGLE86 *((before), (meets))* RECTANGLE99

Spatial subsumption and potential subsumption may perhaps be useful for computer vision and graphics tasks. Potential subsumption can recognize spatial configurations of objects described by library entries from partial and/or abstract observations recorded in N-dimensional observation networks.

[4]We assume that the concepts C-SQUARE and C-RECTANGLE are defined to be consistent.

As the preceding directions suggest, there are ample opportunities to continue the work begun in this thesis.

7.4 Summation

Motivated by an application to configuration problems, this thesis has introduced a powerful new problem solving methodology in description logic: predictive concept recognition by means of a novel closed terminology assumption. This thesis has also shown how to represent and reason with a brand new class of descriptions in a description logic setting, namely constraint network concepts and individuals, which we have applied to plan representation. Finally, both ideas were seamlessly combined in a new approach to predictive plan recognition.

Description logic, configuration, and plan recognition are very exciting areas of artificial intelligence research where much has already been accomplished, yet significant opportunities and challenges lie ahead!

Bibliography

[Allen and Perrault, 1980] J. F. Allen and C. R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15(3):143–178, 1980.

[Allen, 1983a] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[Allen, 1983b] J. F. Allen. Recognizing intentions from natural language utterances. In *Computational Models of Discourse*. MIT Press, Cambridge, MA, 1983.

[Allen, 1984] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[Allen, 1991] J. F. Allen. Planning as temporal reasoning. In *Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 3–14, Cambridge, MA, 1991.

[Apté *et al.*, 1992] C. Apté, R. Dionne, J. Griesmer, M. Karnaugh, J. Kastner, M. Laker, and E. Mays. An experiment in constructing an open expert system using a knowledge substrate. IBM Journal of Research and Development, 1992.

[Artale and Franconi, 1993] A. Artale and E. Franconi. A unified framework for representing time, actions and plans. Workshop on Object-based Representation Systems, August 1993.

[Artale and Franconi, 1994] A. Artale and E. Franconi. A computational account for a description logic of time and action. In *Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, pages 3–14, 1994.

[Baader and Hollunder, 1991] F. Baader and B. Hollunder. Kris: Knowledge representation and inference system. *SIGART Bulletin*, 2(3):8–14, June 1991. Special Issue on Implemented Knowledge Representation and Reasoning Systems.

[Baader and Hollunder, 1992] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92)*, pages 306–317, Cambridge, MA, 1992.

[Baader *et al.*, 1991] F. Baader, H. J. Bürckert, J. Heinsohn, B. Hollunder, J. Müller, B. Nebel, W. Nutt, and H.J. Profitlich. Terminological knowledge representation: A proposal for a terminological logic, 1991.

[Baader *et al.*, 1992] F. Baader, B. Hollunder, B. Nebel, H.J. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological systems or: Making kris get a move on. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92)*, pages 270–281, Cambridge, MA, 1992.

[Bettini, 1994] C. Bettini. A formalization of interval-based temporal subsumption in first order logic. In *Foundations of Knowledge Representation and Reasoning*, pages 53–73. Springer Verlag, 1994.

[Borgida and Patel-Schneider, 1994] A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the classic description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[Borgida *et al.*, 1989] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. Classic: A structural data model for objects. In *Proc. 1989 ACM SIGMOD International Conference on the Management of Data*, pages 58–67, Portland, OR, 1989.

[Borgida, 1992] A. Borgida. Towards the systematic development of terminological reasoners: Clasp reconstructed. In *Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, Cambridge, MA, 1992.

[Brachman and Levesque, 1984] R. J. Brachman and H. J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of AAAI-84*, pages 34–37, Austin, Texas, 1984. American Association of Artificial Intelligence.

[Brachman and Schmolze, 1985] R. J. Brachman and J. G. Schmolze. An overview of the kl-one knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[Brachman, 1985] R. J. Brachman. "i lied about the trees" (or, defaults and definitions in a knowledge representation). *AI Magazine*, 6(3):80–93, 1985.

[Bresciani *et al.*, 1995] P. Bresciani, E. Franconi, and S Tessaris. Implementing and testing expressive description logics: a preliminary report. In *International*

Workshop on Description Logics, Rome, Italy, 1995. Also Proceedings of the International KRUSE Symposium, pp 28-39, Santa Cruz, August 1995.

[Brownston *et al.*, 1985] L. Brownston, R. Farrell, E. Kant, and N. Martin. *Programming Expert Systems in OPS5.* Addison Wesley, Reading, MA, 1985.

[Buchheit *et al.*, 1993] M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–137, 1993.

[Carberry, 1990] S. Carberry. *Plan Recognition in Natural Language Dialogue.* MIT Press, Cambridge, MA, 1990.

[Charniak and Goldman, 1993] E. Charniak and R. P. Goldman. A bayesian model of plan recognition. *Artificial Intelligence*, 64:53–79, 1993.

[Cohen and Levesque, 1990] P. R. Cohen and H. J. Levesque. Rational interaction as the basis for communication. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 221–255. MIT Press, Cambridge, MA, 1990.

[Cohen *et al.*, 1992] W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In *Proceedings of AAAI-92*, pages 754–760, San Jose, CA, 1992. American Association of Artificial Intelligence.

[Dechter *et al.*, 1991] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[Devanbu and Litman, 1991] P. T. Devanbu and D. J. Litman. Plan-based terminological reasoning. In *Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 128–138, Cambridge, MA, 1991.

[Devanbu and Litman, 1996] P. Devanbu and D. Litman. Taxonomic plan reasoning. *Artificial Intelligence Journal*, 1996. Accepted, to appear.

[Devanbu *et al.*, 1991] P. Devanbu, R. J. Brachman, B. W. Ballard, and P. E. Selfridge. Lassie: A knowledge-based software information system. *Communications of the ACM*, 34(5), May 1991.

[Di Eugenio, 1994] B. Di Eugenio. Action representation for interpreting purpose clauses in natural language instructions. In *Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, pages 158–169, 1994.

[DiEugenio and Webber, 1992] B. DiEugenio and B. Webber. Plan recognition in understanding instructions. In *Artificial Intelligence Planning Systems*, College Park, MD, June 1992.

[Dionne, 1991] R. Dionne. personal communication, 1991.

[Donini *et al.*, 1991] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, Cambridge, MA, 1991.

[Donini *et al.*, 1992] F. Donini, M. Lenzerini, D. Nardi, A. Schaerf, and W. Nutt. Adding epistemic operators to concept languages. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92)*, pages 342–354, Cambridge, MA, 1992.

[Doyle and Patil, 1991] J. Doyle and R. Patil. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence*, 48(3):261–297, April 1991.

[Feiner and McKeown, 1990] S. Feiner and K.R. McKeown. Coordinating text and graphics in explanation generation. In *Proceedings of AAAI-90*, pages 442–449, Boston, MA, 1990.

[Fikes and Nilsson, 1971] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Freeman, 1986] M. W. Freeman. Case study of the beacon project: The burroughs browser/editor and automated configurator. In *Proc. of the IEEE Symposium on Logic Programming*, 1986.

[Garey and Johnson, 1979] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of Incompleteness*. W. H. Freeman, San Francisco, CA, 1979.

[Gertner *et al.*, 1995] A. Gertner, B. Webber, and J. R. Clark. Recognizing and evaluating plans with diagnostic actions. In *The Next Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI*, Montreal, Canada, 1995.

[Goodman and Litman, 1992] Bradley A. Goodman and Diane J. Litman. On the interaction between plan recognition and intelligent interfaces. *User Modeling and User-Adapted Interaction*, 2(1-2):55–82, 1992.

[Heinsohn *et al.*, 1992a] J. Heinsohn, D. Kudenko, B. Nebel, and H. J. Profitlich. Rat: Representation of actions using terminological logics. DFKI, 1992.

[Heinsohn *et al.*, 1992b] J. Heinsohn, B. Kundeko, D. Nebel, and H-J. Profitlich. An empirical analysis of terminological representation systems. In *Proceedings of AAAI-92*, San Jose, CA, 1992. American Association of Artificial Intelligence.

[Hollunder *et al.*, 1990] B. Hollunder, Nutt. W., and M. Schmidt-Schauß. Subsumption algorithms for concept description. In *Proceedings of the 9th ECAI*, pages 348–353, Stockholm, Sweden, 1990.

[Kantrowitz, 1991] M. Kantrowitz. Portable utilities for common lisp: User guide & implementation notes. Technical Report CMU-CS-91-143, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, May 1991.

[Kautz and Ladkin, 1991] H. A. Kautz and P. B. Ladkin. Integrating metric and qualitative temporal reasoning. In *Proceedings of AAAI-91*, pages 241–246, Anaheim, CA, 1991.

[Kautz, 1991a] H. Kautz. Metric/allen time system. Documentation, April 1991.

[Kautz, 1991b] H. A. Kautz. A formal theory of plan recognition and its implementation. In J. Allen, H. Kautz, R. Pelavin, and J. Tenenberg, editors, *Reasoning About Plans*, pages 69–125. Morgan Kaufmann, San Mateo, CA, 1991.

[Kindermann, 1992] C. Kindermann. Retraction in terminological knowledge bases. In *Proceedings of the 10th ECAI*, pages 420–424, 1992.

[Kobsa, 1991] A. Kobsa. Utilizing knowledge: The components of the sb-one knowledge representation workbench. In J. Sowa, editor, *Principles of Semantic Networks*, pages 457–486. Morgan Kaufmann, Los Altos, CA, 1991.

[Kramer, 1991] B. Kramer. Knowledge-based configuration of computer systems using hierarchical partial choice. In *Int. Conf. on Tools for AI*, pages 368–375, San Jose, November 1991.

[Kumar, 1992] V. Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, 13(1):32–41, 1992.

[Lambrix and Rönnquist, 1993] P. Lambrix and R. Rönnquist. Tterminological logic involving time and evolution: A preliminary report. In *Proceedings of ISMIS-93*, 1993.

[Lesh and Etzioni, 1995] N. Lesh and O. Etzioni. Insights from machine learning for plan recognition. In *The Next Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI*, pages 78–83, Montreal, Canada, 1995.

[Levesque and Brachman, 1985] H. J. Levesque and R. J. Brachman. A fundamental tradeoff in knowledge representation and reasoning. In *Readings in Knowledge Representation*, pages 42–70. Morgan Kaufmann, Los Altos, CA, 1985.

[Levesque and Brachman, 1987] H. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[Litman and Allen, 1987] D. J. Litman and J. F. Allen. A plan recognition model for subdialogues in conversation. *Cognitive Science*, 11:163–200, 1987.

[MacGregor, 1988] R. MacGregor. A deductive pattern matcher. In *Proceedings of AAAI-88*, pages 403–408, Saint Paul, MN, 1988. American Association of Artificial Intelligence.

[MacGregor, 1991a] R. MacGregor. The evolving technology of classification-based knowledge representation systems. In J. Sowa, editor, *Principles of Semantic Networks*, pages 385–400. Morgan Kaufmann, Los Altos, CA, 1991.

[MacGregor, 1991b] R. MacGregor. Using a description classifier to enhance deductive inference. In *Proc. of the Seventh Conference on Artificial Intelligence Applications (CAIA-91)*, pages 141–147, Miami Beach, FL, 1991.

[MacGregor, 1994] R. M. MacGregor. A description classifier for the predicate calculus. In *Proceedings of AAAI-94*, Seattle, WA, 1994. American Association of Artificial Intelligence.

[Mackworth, 1977] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[Malik and Binford, 1983] J. Malik and T. O. Binford. Reasoning in time and space. In *Proceedings of AAAI-83*, pages 343–345, Washington, D.C., 1983. American Association of Artificial Intelligence.

[Markoff, 1996] J. Markoff. New venture plans system on internet for travelers. *New York Times*, page D1, February 6 1996.

[Mays *et al.*, 1991a] E. Mays, R. Dionne, and R. Weida. K-rep system overview. *SIGART Bulletin*, 2(3):93–97, June 1991. Special Issue on Implemented Knowledge Representation and Reasoning Systems.

[Mays *et al.*, 1991b] E. Mays, S. Lanka, B. Dionne, and R. Weida. A persistent store for large shared knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 3(1):33–41, March 1991.

[Mays *et al.*, 1996] E. Mays, R. Weida, R. Dionne, M. Laker, B. White, C. Liang, and F. Oles. Scalable and expressive medical terminologies. Submitted for publication, 1996.

[McCarthy, 1980] J. McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[McDermott, 1982] J. McDermott. R1: A rule-based configurer of computer systems. *Artificial Intelligence*, 19:39–88, 1982.

[McGuinness and Resnick, 1995] D. L. McGuinness and L. A. Resnick. Description logic-based configuration for consumers. In *International Workshop on Description Logics*, Rome, Italy, 1995.

[Meiri, 1991] I. Meiri. Combining qualitative and quantitative constraints in temporal reasoning. In *Proceedings of AAAI-91*, pages 260–267, Anaheim, CA, 1991.

[Mitchell, 1982] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[Montanari, 1974] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Science*, 7:95–132, 1974.

[Mukerjee and Joe, 1990] A. Mukerjee and G. Joe. A qualitative model for space. In *Proceedings of AAAI-90*, pages 721–727, Boston, MA, 1990.

[Nebel, 1988] B. Nebel. Computational complexity of terminological reasoning in back. *Artificial Intelligence*, 34(3):371–383, 1988.

[Nebel, 1990a] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*. Springer Verlag, Berlin; New York, 1990.

[Nebel, 1990b] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

[Owsnicki-Klewe, 1988] B. Owsnicki-Klewe. Configuration as a consistency maintenance task. In W. Hoeppner, editor, *12th German Workshop on Artificial Intelligence (GWAI-88)*, pages 77–87, Berlin, Germany, 1988. Springer.

[Patel-Schneider and Swartout, 1993] P.F. Patel-Schneider and B. Swartout. Working version (draft): Description logic specification from the krss effort, June 1993.

[Patel-Schneider, 1989] P. F. Patel-Schneider. Undecidability of subsumption in nikl. *Artificial Intelligence*, 39(2):263–272, 1989.

[Pollack, 1990] M. E. Pollack. Plans as complex mental attitudes. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 77–104. MIT Press, Cambridge, 1990.

[Quantz and Royer, 1992] J. J. Quantz and V. Royer. A preference semantics for defaults in terminological lofics. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92)*, pages 294–305, Cambridge, MA, 1992.

[Rönnquist, 1990] R. Rönnquist. A logic for propagation based characterisation of process behavior. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems*, pages 297–304, 1990.

[Reiter, 1978] R. Reiter. On reasoning by default. In *Proceedings of Theoretical Issues in Natural Language Processing-2*, pages 210–218. University of Illinois at Urbana-Champaign, 1978. reprinted in *Readings in Knowledge Representation.*

[Schmidt-Schauß, 1989] M. Schmidt-Schauß. Subsumption in kl-one is undecidable. In *First International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 421–431, Toronto, Ontario, Canada, 1989.

[Schmiedel, 1990] A. Schmiedel. A temporal terminological logic. In *Proceedings of AAAI-90*, pages 640–645, Boston, MA, 1990.

[Schmolze and Mark, 1991] J. G. Schmolze and W. S. Mark. The nikl experience. *Computational Intelligence*, pages 48–69, 1991.

[Searls and Norton, 1990] D. B. Searls and L. M. Norton. Logic-based configuration with a semantic network. *Journal of Logic Programming*, 8:53–73, 1990.

[Shieber, 1983] S. Shieber. Direct parsing of id/lp grammars. *Linguistics and Philosophy*, 7(2):135–154, 1983.

[Shoham and McDermott, 1990] Y. Shoham and D. McDermott. Temporal reasoning. In *Encyclopedia of Artificial Intelligence, 2nd. ed.*, pages 1334–1338. John Wiley and Sons, New York, 1990.

[Shoham, 1987] Y. Shoham. Temporal logics in ai: Semantical and ontological considerations. *Artificial Intelligence*, 33(1):89–104, 1987.

[Sidner, 1985] C. L. Sidner. Plan parsing for intended response recognition in discourse. *Computational Intelligence*, 1(1):1–10, Feb 1985.

[Song and Cohen, 1991] F. Song and R. Cohen. Temporal reasoning during plan recognition. In *Proceedings of AAAI-91*, pages 247–252, Anaheim, CA, 1991.

[Song, 1991] F. Song. A processing model for temporal analysis and its application to plan recognition. Technical Report CS-91-15, Department of Computer Science, University of Waterloo, Waterloo, Ontario, 1991. Ph.D. Thesis.

[Steele, 1990] G. L. Jr. Steele. *Common Lisp: The Language (Second Edition).* Digital Press, Bedford, MA, 1990.

[Swartout and Neches, 1986] W. Swartout and R. Neches. The shifting terminological space: An impediment to evolvability. In *Proceedings of AAAI-86*, pages 936–941, Philadelphia, PA, 1986. American Association of Artificial Intelligence.

[Valdes-Perez, 1986] R. E. Valdes-Perez. Spatio-temporal reasoning and linear inequalities. Technical Report AI Memo No. 875, MIT/Artificial Intelligence Laboratory, Cambridge, MA, 1986.

[van Beek and Cohen, 1991] P. van Beek and R. Cohen. Resolving plan ambiguity for cooperative response generation. In *IJCAI 91*, pages 938–944, Sydney, Australia, 1991.

[van Beek, 1989] P. van Beek. Approximation algorithms for temporal reasoning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1291–1296, Detroit, 1989.

[Vilain and Kautz, 1986] M. Vilain and H. Kautz. Constraint-propagation algorithms for temporal reasoning. In *Proceedings of AAAI-86*, pages 377–382, Philadelphia, PA, 1986. American Association of Artificial Intelligence.

[Vilain *et al.*, 1989] M. Vilain, H. Kautz, and P. van Beek. Constraint propagation algorithms for temporal reasoning: a revised report. In J. deKleer and D. Weld, editors, *Readings in Qualitative Reasoning About Physical Systems.* Morgan Kaufmann, Los Altos, CA, 1989.

[Vilain, 1990] M. Vilain. Getting serious about parsing plans: a grammatical analysis of plan recognition. In *Proceedings of AAAI-90*, pages 190–196, Boston, MA, 1990.

[von Luck *et al.*, 1987] K. von Luck, B. Nebel, C. Peltason, and A. Schmiedel. The anatomy of the back system. Technical Report KIT- Report 41, Technische Universitat Berlin, Berlin, 1987.

[Wahlster *et al.*, 1993] W. Wahlster, E. André, W. Finkler, H.-J. Profitlich, and T. Rist. Plan-based integration of natural language and graphics. *Artificial Intelligence*, 63(1-2):387–428, 1993.

[Weida and Litman, 1992] R. Weida and D. Litman. Terminological reasoning with constraint networks and an application to plan recognition. In *Principles* *of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92)*, pages 282–293, Cambridge, MA, 1992.

[Weida and Litman, 1994] R. Weida and D. Litman. Subsumption and recognition of heterogeneous constraint networks. In *The Tenth Conference on Artificial Intelligence for Applications*, pages 381–388. IEEE Computer Society, 1994.

[Weida, 1993] R. A. Weida. Terminological constraint network reasoning and its application to plan recognition *(ph.d. dissertation proposal)*. Technical Report CUCS-027-93, Columbia University Department of Computer Science, 1993.

[Weida, 1995a] R. Weida. Closing the terminology. In *International Workshop on Description Logics*, Rome, Italy, 1995.

[Weida, 1995b] R. Weida. Knowledge representation for plan recognition. In *The Next Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI*, pages 119–123, Montreal, Canada, 1995.

[Weida, 1996] R. A. Weida. Closed terminologies in description logics. In *Proceedings of AAAI-96*, Portland, OR, 1996. American Association of Artificial Intelligence.

[Wellman, 1990] M. P. Wellman. *Formulation of Tradeoffs in Planning Under Uncertainty*. Morgan Kaufmann, San Mateo, CA, 1990.

[Woods and Schmolze, 1992] W. A. Woods and J. G. Schmolze. The kl-one family. *Computers and Mathematics with Applications*, 74(2-5), 1992.

[Woods, 1986] W. A. Woods. Important issues in knowledge representation. *Proceedings of the IEEE*, 74(10):1322–1334, 1986.

[Woods, 1991] W. A. Woods. Understanding subsumption and taxonomy: A framework for progress. In J. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 45–94. Morgan Kaufmann, Los Altos, CA, 1991.

[Wright *et al.*, 1993] J. R. Wright, E. S. Weixelbaum, K. E. Brown, G. T. Vesonder, S. R. Palmer, J. I. Berman, and H. H. Moore. A knowledge-based configurator that supports sales, engineering, and manufacturing at at&t network systems. In *Proceedings of Fifth Innovative Applications of Artificial Intelligence Conference*, Washington, DC, 1993.

[Yampratoom and Allen, 1993] E. Yampratoom and L. Allen. Performance of temporal reasoning systems. Technical Report TRAINS Technical Note 93-1, The University of Rochester, Computer Science Department, Rochester, NY, March 1993.

[Yen *et al.*, 1991] J. Yen, R. Neches, and R. MacGregor. Clasp: Integrating term subsumption systems and production systems. *IEEE Transactions on Knowledge and Data Engineering*, 3(1):25–31, March 1991.

[Yen, 1990] J. Yen. A principled approach to reasoning about the specificity of rules. In *Proceedings of AAAI-90*, pages 701–707, Boston, MA, 1990. American Association of Artificial Intelligence.

Appendix A

Extended Plan Recognition Example

This appendix contains a larger example of a T-REX travel plan library as described in Section 5.6. All figures are taken verbatim from a T-REX input file. Note that the syntax varies slightly from the formatted examples used in the body of the thesis.[1] Plan definitions appear in Figure A.2 beginning on page 274, and the resulting plan taxonomy is diagrammed in Figure A.3 on page 282. This example reinforces the value of classifying a plan library. Even with only 35 plans in the library, most humans would find it very difficult, if not impossible, to visualize all the subsumption relationships among these definitions without the classifier. For reference, the underlying concept taxonomy is shown in Figure A.4 on page 283. Although the underlying concept definitions are not shown, for the purposes of this example, the reader may rely on the intuitive meanings of the concept's names. A set of rules appears in Figure A.5 on page 284. In addition, a set of demons appears in Figure A.6 on page 285. Starting on page 286 is a transcript of T-REX output for an extended travel plan recognition problem using the plans, rules, and

[1] In particular, steps appear immediately after a plan's name, and are not preceded by the *:steps* keyword. These definitions use the MATS abbreviations for temporal relations reproduced from [Kautz, 1991a] in Figure A.1. Metric constraints should be clear, with the understanding that MATS uses *left* and *right* where this thesis has used *start* and *finish*, respectively.

demons defined in this appendix. Comment lines beginning with semicolons were added by hand.[2]

[2]Insignificant reformatting to fit the material on the printed pages was also performed.

```
The following names are used for the 13 Allen relations:

        =       equals
        d       during (proper)
        di      contains (proper)
        s       starts
        si      started-by
        f       finishes
        fi      finished-by
        p       precedes (before)
        pi      preceded by (after)
        m       meets
        mi      met-by
        o       overlaps
        oi      overlapped-by

The system also understands the following abbreviations:

        any             = d di s si f fi p pi m mi o oi
        during          d s f
        contains        di si fi
        disjoint        p m mi pi
        b               p
        a               pi
        c               di
        e               =
        <               p
        >               pi
        (not r1 ...)    every relation other than (r1 ...).
                        E.g., (not p m) = (= di s si f fi pi mi o oi)
        (~ r1 ...)      same as (not r1 ...)
```

Figure A.1: Specifying Qualitative Relations for MATS (from [Kautz, 1991a])

```
(defplan CULTURAL-TRIP
  ((s1 cultural-act)))

(defplan HIKING-TRIP
  ((s1 hike)))

;;; The conjunction specified in the step of the following plan
;;; results in a system-defined concept

(defplan SKI-TOUR
  ((s1 (and xc-ski travel-act))))

;;; The following plan is equivalent to the preceding one.
;;; T-REX issues a warning and ignores the redundant definition.

(defplan EQUIVALENT-SKI-TOUR
  ((s1 xc-ski)
   (s1 travel-act)))

(defplan CLIMBING-TRIP
  ((s1 climb-mountain)))

(defplan NORTH-AMERICAN-TRIP
  ((s1 visit-north-american-country)))

(defplan NORTH-AMERICAN-MUSEUM-TRIP
  ((s1 visit-north-american-country)
   (s2 visit-museum))
  :qualitative-constraints ((s1 c s2)))

(defplan NORTH-AMERICAN-CAPITOL-TRIP
  ((s1 visit-north-american-country)
   (s2 visit-capitol-city))
  :qualitative-constraints ((s2 d s1)))
```

Figure A.2: Travel Plan Definitions

```
(defplan EUROPEAN-TRIP
  ((s1 visit-european-country)))

(defplan EUROPEAN-MUSEUM-TRIP
  ((s1 visit-european-country)
   (s2 visit-museum))
  :qualitative-constraints ((s1 c s2)))

(defplan EUROPEAN-CAPITOL-TRIP
  ((s1 visit-european-country)
   (s2 visit-capitol-city))
  :qualitative-constraints ((s2 d s1)))

(defplan ENGLAND-TRIP
  ((s1 visit-england)))

(defplan MULTIPLE-EUROPEAN-COUNTRIES
  ((s1 visit-european-country)
   (s2 visit-european-country)))

(defplan TOUR-ENGLAND-A
  ((s1 visit-england)
   (s2 visit-london)
   (s3 visit-british-museum)
   (s4 attend-theater))
  :qualitative-constraints ((s1 c s2) (s2 c s3) (s2 c s4))
  :metric-constraints ((10 <= right s1 - left s1 <= 10)
                       (3 <= right s2 - left s2 <= 3)))
```

Figure A.2: Travel Plan Definitions (continued)

```
(defplan TOUR-ENGLAND-B
  ((s1 visit-england)
   (s2 visit-london)
   (s3 visit-york)
   (s4 visit-bath)
   (s5 visit-stonehenge))
  :qualitative-constraints ((s1 si s2) (s1 c s3) (s1 c s4) (s1 c s4))
  :metric-constraints ((10 <= right s1 - left s1 <= 10)
                       (3 <= right s2 - left s2 <= 3)))

(defplan TOUR-FRANCE-A
  ((s1 visit-france)
   (s2 visit-paris)
   (s3 visit-louvre)
   (s4 visit-musee-dorsay))
  :qualitative-constraints ((s1 c s2) (s2 c s3) (s2 c s4))
  :metric-constraints ((21 <= right s1 - left s1 <= 21)
                       (4 <= right s2 - left s2 <= 4)))

(defplan TOUR-FRANCE-B
  ((s1 visit-france)
   (s2 visit-chamonix)
   (s3 climb-mont-blanc))
  :qualitative-constraints ((s2 d s1)
                            (s3 d s2))
  :metric-constraints ((7 <= right s1 - left s1 <= 7)
                       (1 <= right s2 - left s2 <= 1)))
```

Figure A.2: Travel Plan Definitions (continued)

```
(defplan TOUR-ITALY
  ((s1 visit-italy)
   (s2 visit-rome)
   (s3 visit-coliseum)
   (s4 attend-opera))
  :qualitative-constraints ((s1 c s2) (s2 c s3) (s2 c s4))
  :metric-constraints ((17 <= right s1 - left s1 <= 17)
                       (4 <= right s2 - left s2 <= 5)))

(defplan ALPINE-PEAKS
  ((s1 visit-france)
   (s2 climb-mont-blanc)
   (s3 visit-switzerland)
   (s4 climb-matterhorn))
  :qualitative-constraints ((s2 d s1)
                            (s4 d s3)
                            (s1 m s3)))

;;; Both steps of the following plan have macro actions

(defplan TOUR-ENGLAND-FRANCE-COMBO
  ((s1 tour-england-a)
   (s2 tour-france-a)))

;;; Macros may be arbitrarily nested.  The "ef" step of the next plan
;;; has a macro action which in turn contains macros.

(defplan TOUR-ENGLAND-FRANCE-ITALY-COMBO
  ((ef tour-england-france-combo)
   (i tour-italy))
  :qualitative-constraints ((ef (b m) i)))

(defplan USA-TRIP
  ((s1 visit-usa)))
```

Figure A.2: Travel Plan Definitions (continued)

```
(defplan TOUR-WASHINGTON-DC
  ((s1 visit-usa)
   (s2 visit-washington)
   (s3 visit-smithsonian))
  :qualitative-constraints ((s2 d s1) (s3 d s2))
  :metric-constraints ((3 <= right s1 - left s1 <= 3)))

(defplan TOUR-CALIFORNIA-PARKS
  ((s1 visit-usa)
   (s2 visit-yosemite-np)
   (s3 visit-sequoia-np)
   (s4 visit-kings-canyon-np))
  :qualitative-constraints ((s2 d s1) (s3 d s1) (s4 d s1))
  :metric-constraints ((3 <= right s1 - left s1 <= 3)))

(defplan TOUR-WYOMING
  ((s1 visit-usa)
   (s2 visit-teton-np)
   (s3 visit-yosemite-np)
   (s4 visit-jackson))
  :qualitative-constraints ((s2 d s1) (s3 d s1))
  :metric-constraints ((1 <= right s4 - left s4 <= 1)
                       (3 <= right s2 - left s2 <= 3)
                       (4 <= right s3 - left s3 <= 4)
                       (8 <= right s1 - left s1 <= 8)))

(defplan NATIONAL-PARK-CLIMBING-TRIP
  ((step1 visit-national-park)
   (step2 climb-mountain))
  :qualitative-constraints ((step2 d step1)))
```

Figure A.2: Travel Plan Definitions (continued)

```
(defplan RAINIER-HIKING-AND-CLIMBING-TRIP
  ((step1 visit-usa)
   (step2 visit-mt-rainier-np)
   (step3 hike-wonderland-trail)
   (step4 climb-mount-rainier))
  :qualitative-constraints ((step1 c step2)
                            (step2 c step3)
                            (step2 c step4)))

(defplan TETONS-HIKING-AND-CLIMBING-TRIP
  ((step1 visit-usa)
   (step2 visit-teton-np)
   (step3 hike-jenny-lake)
   (step4 climb-grand-teton))
  :qualitative-constraints ((step2 d step1)
                            (step3 d step2)
                            (step4 d step2)))

(defplan TOUR-SWITZERLAND
  ((s1 visit-switzerland)
   (s2 visit-zermatt)
   (s3 climb-matterhorn))
  :qualitative-constraints ((s2 d s1) (s3 d s2))
  :metric-constraints ((4 <= right s1 - left s1 <= 4)
                       (1 <= right s2 - left s2 <= 1)))
```

Figure A.2: Travel Plan Definitions (continued)

```
;;; The TOUR-DU-MONT-BLANC plan serves as a template for the nearly
;;; equivalent pair of plans which follow it

(defplan TOUR-DU-MONT-BLANC
  ((s1 visit-france)
   (s2 visit-italy)
   (s3 visit-switzerland)
   (s4 visit-france)
   (c1 visit-chamonix)
   (c2 visit-courmayeur)
   (c4 visit-chamonix)
   (t travel-act))
  :qualitative-constraints ((s1 m s2) (s2 m s3) (s3 m s4)
                            (c1 d s1) (c2 d s2) (c4 d s4)
                            (c1 o t) (t o c4)))

;;; Here, the travel-act in the TOUR-DU-MONT-BLANC macro is refined
;;; to HIKE.  Notice that the label associated with HIKE refers
;;; to a substep within the macro and thus does not introduce an
;;; additional step.

(defplan TOUR-DU-MONT-BLANC-HIKING
  ((m tour-du-mont-blanc)
   ((t m) hike)))

;;; Here, the travel-act in the TOUR-DE-MONT-BLANC macro is refined
;;; to XC-SKI

(defplan TOUR-DE-MONT-BLANC-SKIING
  ((m tour-du-mont-blanc)
   ((t m) xc-ski)))
```

Figure A.2: Travel Plan Definitions (continued)

```
;;; The HAUTE-ROUTE plan serves as a template for the nearly
;;; equivalent pair of plans which follow it

(defplan HAUTE-ROUTE
  ((s1 visit-france)
   (s2 visit-switzerland)
   (c1 visit-chamonix)
   (c2 visit-zermatt)
   (t travel-act))
  :qualitative-constraints ((s1 (m mi) s2)
                            (c1 d s1) (c2 d s2)
                            (c1 o t) (t o c2)))

(defplan HAUTE-ROUTE-HIKING
  ((m haute-route)
   ((t m) hike)))

(defplan HAUTE-ROUTE-SKIING
  ((m haute-route)
   ((t m) xc-ski)))
```

Figure A.2: Travel Plan Definitions (conclusion)

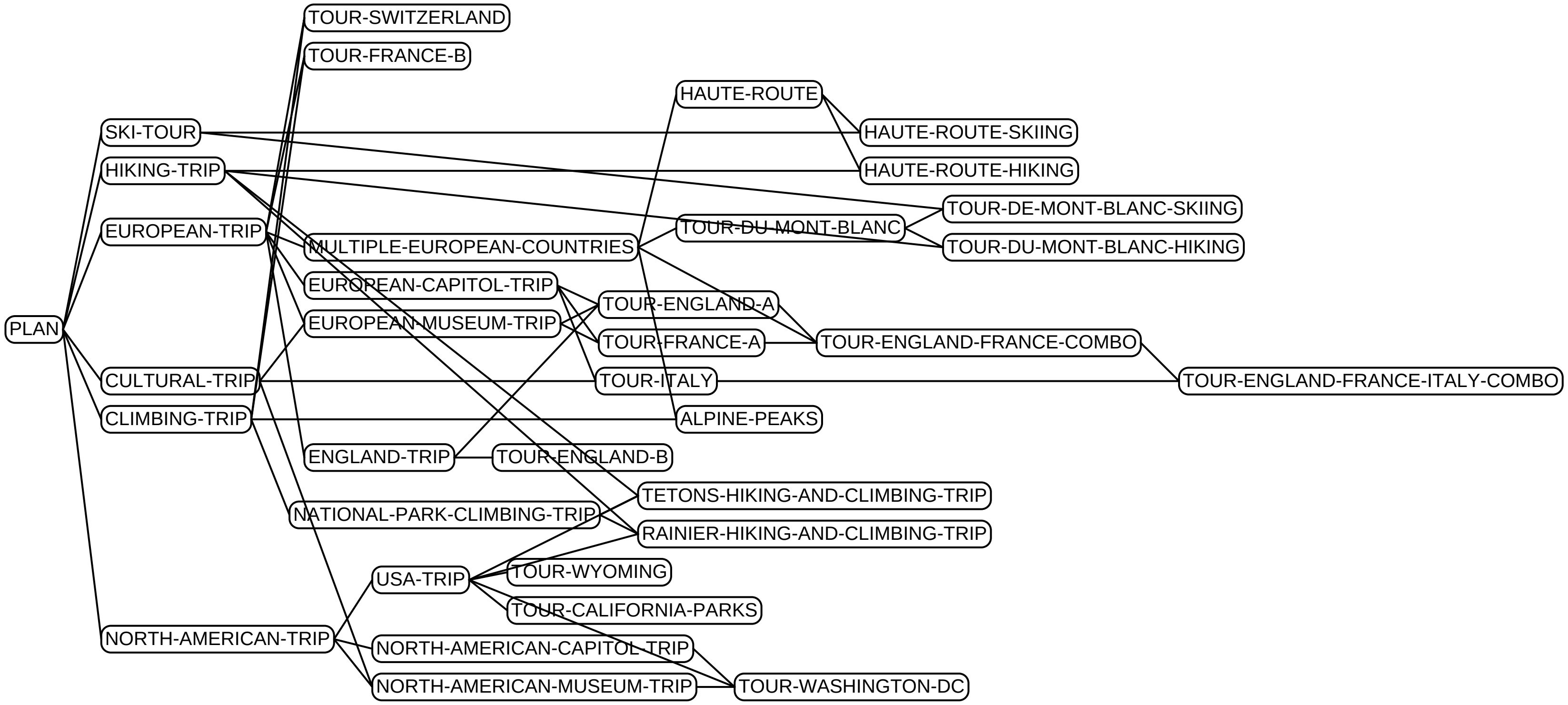


Figure A.3: Travel Plan Taxonomy

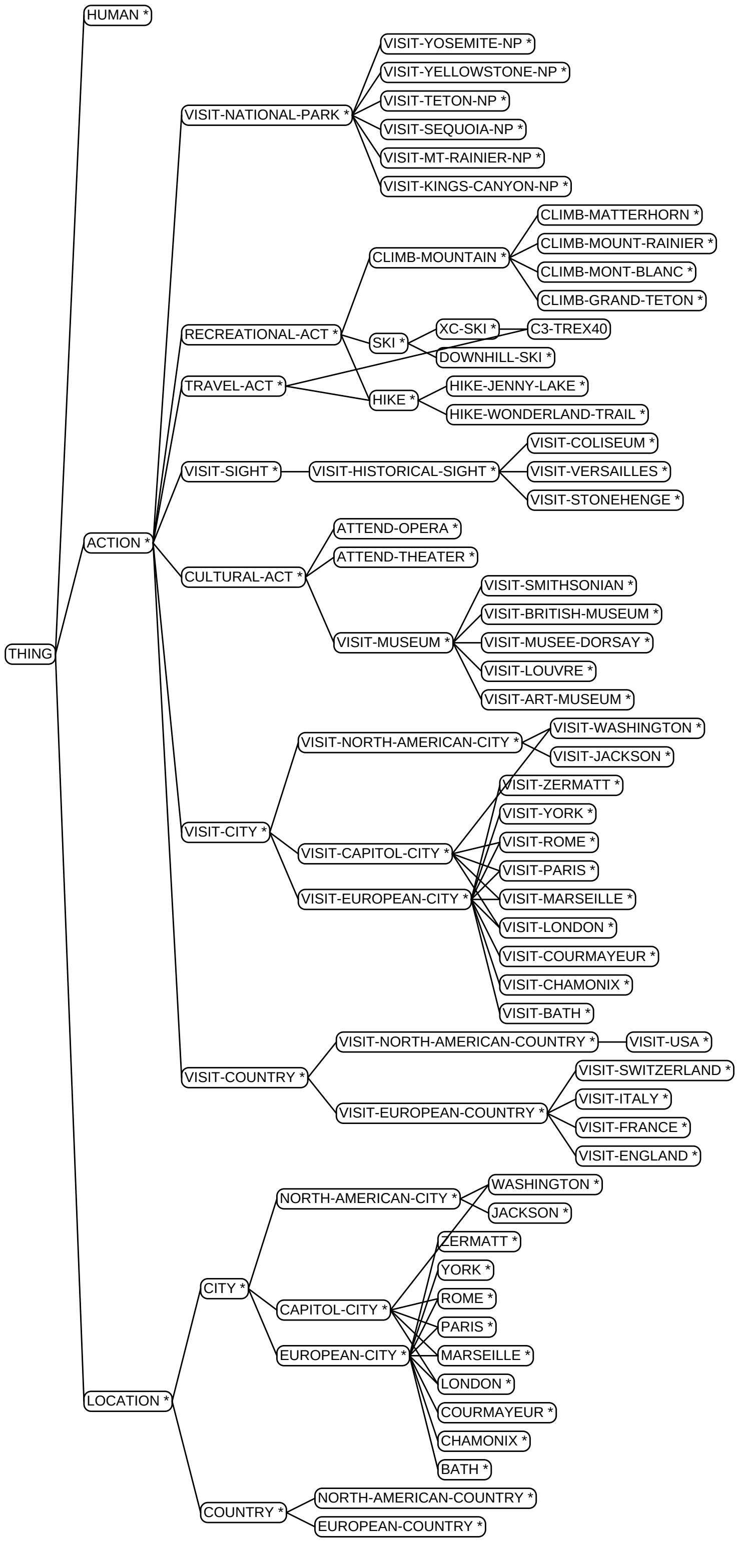


Figure A.4: Travel Concept Taxonomy

```
;;; One must leave a location before returning (presuming
;;; no embedded cycles)

(defrule ROUND-TRIP
  (((s1 travel-act)
    (s2 travel-act))
   :coref-constraints ((equal (origin s1) (destination s2))))
  :qualitative-consequents ((s1 (b m mi a) s2)))


;;; Consecutive legs of a trip must connect in one place

(defrule CONNECTING-TRAVEL
  (((s1 travel-act)
    (s2 travel-act))
   :qualitative-constraints ((s1 (m) s2)))
  :coref-consequents ((equal (origin s2) (destination s1))))

(defrule DISJOINT-COUNTRIES   ;;; one country at a time
  (((s1 visit-country)
    (s2 visit-country)))
  :qualitative-consequents ((s1 (b m mi a) s2)))

(defrule DISJOINT-CITIES   ;;; one city at a time
  (((s1 visit-city)
    (s2 visit-city)))
  :qualitative-consequents ((s1 (b m mi a) s2)))

(defrule VISIT-COUNTRY-DURATION
  (((s1 visit-country)))
  :metric-consequents ((1 <= right s1 - left s1 <= 28)))

(defrule VISIT-EUROPEAN-COUNTRY-DURATION
  (((s1 visit-european-country)))
  :metric-consequents ((3 <= right s1 - left s1 <= 28)))
```

Figure A.5: Travel Plan Rules

```
;;; Integrity constraint: the universal plan should always be necessary

(defdemon TREX::PLAN
  (lambda (self)
    (format t "~%A bug exists if this demon ever fires."))
  :to (:optional :impossible))

(defdemon CLIMBING-TRIP
  (lambda (self)
    (format t "~%A climbing tour requires a physician's approval."))
  :to (:necessary))

(defdemon CLIMBING-TRIP
  (lambda (self)
    (format t "~%Discretion is the better part of valor!"))
  :from (:necessary)
  :to (:impossible))

(defdemon MULTIPLE-EUROPEAN-COUNTRIES
  (lambda (self)
    (format t "~%For the ~s package, a Eurailpass is included."
            (tr::name self)))
  :to (:necessary))

(defdemon TOUR-ITALY
  (lambda (self)
    (format t "~%For the ~s package, a special airfare is available."
            (tr::name self)))
  :to (:necessary))

(defdemon USA-TRIP
  (lambda (self)
    (format t "~%For traveling abroad, a passport is necessary."
            (tr::name self)))
  :to (:impossible))
```

Figure A.6: Travel Plan Demons

```
;;; The RUN-TEST-SUITE function conducts incremental plan recognition
;;; while stepping through successive (re)definitions of the observation
;;; network.  Evaluation of this LISP expression generates all of the
;;; remaining system output in this example.  We will refer back to
;;; these versions of the observations in later comments.
;;;
;;; If we imagine a user interacting with a travel planning system
;;; through a natural language interface, the user's incremental
;;; description of the desired trip as reflected in these observations
;;; can be paraphrased as:
;;;
;;; (1) I'm interested in doing some climbing.
;;; (2) I also want to visit Europe.
;;; (3) In particular, I'd like to visit France.
;;; (4) And I'd like to see another country too.

USER> (run-test-suite `(((defobservations VERSION-1
                           ((s1 climb-mountain)))
                         (defobservations VERSION-2
                           ((s1 climb-mountain)
                            (s2 visit-european-country)))
                         (defobservations VERSION-3
                           ((s1 climb-mountain)
                            (s2 visit-france)))
                         (defobservations VERSION-4
                           ((s1 climb-mountain)
                            (s2 visit-france)
                            (s3 visit-european-country))))))

;;; T-REX prints a formatted representation of the first version of the
;;; observation network, omitting details of trivial constraints.  In
;;; this case, there is only one step (named s1, with a CLIMB-MOUNTAIN
;;; action) and all arcs are trivial.  K-REP prefixes concepts (as
;;; opposed to symbols which name them) with "#K".

VERSION-1:
```

```
  Qualitative Constraints:

     (S1)/#KCLIMB-MOUNTAIN

  Metric Constraints:

     LEFT (S1)/#KCLIMB-MOUNTAIN
     RIGHT (S1)/#KCLIMB-MOUNTAIN

;;; During predictive recognition, T-REX discovers that the
;;; CLIMBING-TRIP plan has become necessary.  A demon fires and
;;; prints the following message:

A climbing tour requires a physician's approval.

;;; A complete list of necessary, optional, and impossible plans
;;; follows.  This output is produced by the non-incremental version
;;; of T-REX's predictive recognition code.  At our option, the non-
;;; incremental version is being run interleaved with the incremental
;;; version.  There are two reasons for making this choice, which is
;;; used during development and testing: (1) it happens to cause the
;;; complete list of plans to be printed by modality for reference,
;;; and (2) T-REX automatically compares the results of the
;;; incremental and non-incremental versions to ensure that they agree
;;; with one another.  T-REX prefixes plans (as opposed to symbols
;;; which name them) with "#p".

 NECESSARY: #pTREX::PLAN
            #pCLIMBING-TRIP

  OPTIONAL: #pEUROPEAN-TRIP
            #pMULTIPLE-EUROPEAN-COUNTRIES
            #pNORTH-AMERICAN-TRIP
            #pUSA-TRIP
            #pTOUR-SWITZERLAND
            #pNATIONAL-PARK-CLIMBING-TRIP
            #pALPINE-PEAKS
```

```
            #pTOUR-FRANCE-B
            #pHIKING-TRIP
            #pTETONS-HIKING-AND-CLIMBING-TRIP
            #pRAINIER-HIKING-AND-CLIMBING-TRIP

IMPOSSIBLE: #pHAUTE-ROUTE
            #pTOUR-DU-MONT-BLANC
            #pENGLAND-TRIP
            #pTOUR-ENGLAND-B
            #pEUROPEAN-CAPITOL-TRIP
            #pTOUR-WYOMING
            #pTOUR-CALIFORNIA-PARKS
            #pNORTH-AMERICAN-CAPITOL-TRIP
            #pSKI-TOUR
            #pHAUTE-ROUTE-SKIING
            #pTOUR-DE-MONT-BLANC-SKIING
            #pHAUTE-ROUTE-HIKING
            #pTOUR-DU-MONT-BLANC-HIKING
            #pCULTURAL-TRIP
            #pTOUR-ITALY
            #pEUROPEAN-MUSEUM-TRIP
            #pTOUR-FRANCE-A
            #pTOUR-ENGLAND-A
            #pTOUR-ENGLAND-FRANCE-COMBO
            #pTOUR-ENGLAND-FRANCE-ITALY-COMBO
            #pNORTH-AMERICAN-MUSEUM-TRIP
            #pTOUR-WASHINGTON-DC

;;; The following recognition state is generated by the incremental
;;; version of T-REX's predictive recognition code:

 MSNs = (#pCLIMBING-TRIP)
 MGOs = (#pEUROPEAN-TRIP #pNORTH-AMERICAN-TRIP #pNATIONAL-PARK-CLIMBING-TRIP
         #pHIKING-TRIP)
 MSOs = (#pTOUR-SWITZERLAND #pALPINE-PEAKS #pTOUR-FRANCE-B
         #pTETONS-HIKING-AND-CLIMBING-TRIP #pRAINIER-HIKING-AND-CLIMBING-TRIP)
```

```
;;; T-REX next prints a formatted representation of the second version
;;; of the observation network.  Notice that a metric constraint on
;;; the duration of the newly added VISIT-EUROPEAN-COUNTRY step has
;;; been inferred by the VISIT-EUROPEAN-COUNTRY-DURATION rule, which
;;; matched this version of the observations.  (The VISIT-COUNTRY-DURATION
;;; rule fired also, but it only ensures a more general constraint
;;; than that of VISIT-EUROPEAN-COUNTRY-DURATION.)

VERSION-2:

  Qualitative Constraints:

     (S2)/#KVISIT-EUROPEAN-COUNTRY
     (S1)/#KCLIMB-MOUNTAIN

  Metric Constraints:

     LEFT (S2)/#KVISIT-EUROPEAN-COUNTRY [-28,-3] RIGHT (S2)/#KVISIT-EUROPEAN-COUNTRY

     RIGHT (S2)/#KVISIT-EUROPEAN-COUNTRY [3,28]  LEFT (S2)/#KVISIT-EUROPEAN-COUNTRY

     LEFT (S1)/#KCLIMB-MOUNTAIN
     RIGHT (S1)/#KCLIMB-MOUNTAIN

;;; During predictive recognition, T-REX discovers that the USA-TRIP
;;; plan has become impossible.  A demon fires and prints the following
;;; message:

For traveling abroad, a passport is necessary.


;;; Here are the necessary, optional, and impossible plans after the
;;; second version of the observations:

 NECESSARY: #pTREX::PLAN
            #pEUROPEAN-TRIP
```

```
            #pCLIMBING-TRIP

  OPTIONAL: #pMULTIPLE-EUROPEAN-COUNTRIES
            #pTOUR-SWITZERLAND
            #pALPINE-PEAKS
            #pTOUR-FRANCE-B

IMPOSSIBLE: #pHAUTE-ROUTE
            #pTOUR-DU-MONT-BLANC
            #pENGLAND-TRIP
            #pTOUR-ENGLAND-B
            #pEUROPEAN-CAPITOL-TRIP
            #pNORTH-AMERICAN-TRIP
            #pUSA-TRIP
            #pTOUR-WYOMING
            #pTOUR-CALIFORNIA-PARKS
            #pNORTH-AMERICAN-CAPITOL-TRIP
            #pNATIONAL-PARK-CLIMBING-TRIP
            #pSKI-TOUR
            #pHAUTE-ROUTE-SKIING
            #pTOUR-DE-MONT-BLANC-SKIING
            #pHIKING-TRIP
            #pHAUTE-ROUTE-HIKING
            #pTOUR-DU-MONT-BLANC-HIKING
            #pTETONS-HIKING-AND-CLIMBING-TRIP
            #pRAINIER-HIKING-AND-CLIMBING-TRIP
            #pCULTURAL-TRIP
            #pTOUR-ITALY
            #pEUROPEAN-MUSEUM-TRIP
            #pTOUR-FRANCE-A
            #pTOUR-ENGLAND-A
            #pTOUR-ENGLAND-FRANCE-COMBO
            #pTOUR-ENGLAND-FRANCE-ITALY-COMBO
            #pNORTH-AMERICAN-MUSEUM-TRIP
            #pTOUR-WASHINGTON-DC

;;; Now the incremental recognition state captures a significantly
```

```
;;; smaller optional region in the plan taxonomy than it did after the
;;; previous version of the observations:

 MSNs = (#pCLIMBING-TRIP #pEUROPEAN-TRIP)
 MGOs = (#pTOUR-SWITZERLAND #pTOUR-FRANCE-B #pMULTIPLE-EUROPEAN-COUNTRIES)
 MSOs = (#pTOUR-SWITZERLAND #pALPINE-PEAKS #pTOUR-FRANCE-B)


;;; Here is the third version of the observation network.  Notice that
;;; the VISIT-EUROPEAN-COUNTRY step has been refined to VISIT-FRANCE.

VERSION-3:

  Qualitative Constraints:

     (S2)/#KVISIT-FRANCE
     (S1)/#KCLIMB-MOUNTAIN

  Metric Constraints:

     LEFT (S2)/#KVISIT-FRANCE       [-28,-3]       RIGHT (S2)/#KVISIT-FRANCE

     RIGHT (S2)/#KVISIT-FRANCE      [3,28]         LEFT (S2)/#KVISIT-FRANCE

     LEFT (S1)/#KCLIMB-MOUNTAIN
     RIGHT (S1)/#KCLIMB-MOUNTAIN


;;; Here are the necessary, optional, and impossible plans after the
;;; third version of the observations:

 NECESSARY: #pTREX::PLAN
            #pEUROPEAN-TRIP
            #pCLIMBING-TRIP

  OPTIONAL: #pMULTIPLE-EUROPEAN-COUNTRIES
            #pALPINE-PEAKS
```

```
            #pTOUR-FRANCE-B

IMPOSSIBLE: #pHAUTE-ROUTE
            #pTOUR-DU-MONT-BLANC
            #pENGLAND-TRIP
            #pTOUR-ENGLAND-B
            #pEUROPEAN-CAPITOL-TRIP
            #pNORTH-AMERICAN-TRIP
            #pUSA-TRIP
            #pTOUR-WYOMING
            #pTOUR-CALIFORNIA-PARKS
            #pNORTH-AMERICAN-CAPITOL-TRIP
            #pTOUR-SWITZERLAND
            #pNATIONAL-PARK-CLIMBING-TRIP
            #pSKI-TOUR
            #pHAUTE-ROUTE-SKIING
            #pTOUR-DE-MONT-BLANC-SKIING
            #pHIKING-TRIP
            #pHAUTE-ROUTE-HIKING
            #pTOUR-DU-MONT-BLANC-HIKING
            #pTETONS-HIKING-AND-CLIMBING-TRIP
            #pRAINIER-HIKING-AND-CLIMBING-TRIP
            #pCULTURAL-TRIP
            #pTOUR-ITALY
            #pEUROPEAN-MUSEUM-TRIP
            #pTOUR-FRANCE-A
            #pTOUR-ENGLAND-A
            #pTOUR-ENGLAND-FRANCE-COMBO
            #pTOUR-ENGLAND-FRANCE-ITALY-COMBO
            #pNORTH-AMERICAN-MUSEUM-TRIP
            #pTOUR-WASHINGTON-DC

;;; Again, the incremental version captures a further reduced set of
;;; optional plans:

 MSNs = (#pCLIMBING-TRIP #pEUROPEAN-TRIP)
 MGOs = (#pTOUR-FRANCE-B #pMULTIPLE-EUROPEAN-COUNTRIES)
```

```
 MSOs = (#pALPINE-PEAKS #pTOUR-FRANCE-B)

;;; T-REX next prints a formatted representation of the fourth version
;;; of the observation network, where another VISIT-EUROPEAN-COUNTRY
;;; step has been added.  As before, a metric constraint on the duration
;;; of the new VISIT-EUROPEAN-COUNTRY step has been added by the rule
;;; named VISIT-EUROPEAN-COUNTRY-DURATION.  Furthermore, the
;;; DISJOINT-COUNTRIES rule has inferred that the VISIT-FRANCE and
;;; VISIT-EUROPEAN-COUNTRY steps are temporally disjoint.


VERSION-4:

  Qualitative Constraints:

     (S2)/#KVISIT-FRANCE      DISJOINT  (S3)/#KVISIT-EUROPEAN-COUNTRY

     (S1)/#KCLIMB-MOUNTAIN
     (S3)/#KVISIT-EUROPEAN-COUNTRY DISJOINT (S2)/#KVISIT-FRANCE


  Metric Constraints:

     LEFT (S2)/#KVISIT-FRANCE       [-28,-3]     RIGHT (S2)/#KVISIT-FRANCE

     RIGHT (S2)/#KVISIT-FRANCE      [3,28]       LEFT (S2)/#KVISIT-FRANCE

     LEFT (S1)/#KCLIMB-MOUNTAIN
     RIGHT (S1)/#KCLIMB-MOUNTAIN
     LEFT (S3)/#KVISIT-EUROPEAN-COUNTRY [-28,-3] RIGHT (S3)/#KVISIT-EUROPEAN-COUNTRY

     RIGHT (S3)/#KVISIT-EUROPEAN-COUNTRY [3,28]  LEFT (S3)/#KVISIT-EUROPEAN-COUNTRY


;;; During predictive recognition, T-REX discovers that the
;;; MULTIPLE-EUROPEAN-COUNTRIES plan has become necessary.  A demon
;;; fires and prints the following message:
```

```
For the MULTIPLE-EUROPEAN-COUNTRIES package, a Eurailpass is included.


;;; The following are the necessary, optional, and impossible plans
;;; after the fourth version of the observations.  Notice that there is
;;; only one optional plan left:

 NECESSARY: #pTREX::PLAN
            #pEUROPEAN-TRIP
            #pMULTIPLE-EUROPEAN-COUNTRIES
            #pCLIMBING-TRIP

  OPTIONAL: #pALPINE-PEAKS

IMPOSSIBLE: #pHAUTE-ROUTE
            #pTOUR-DU-MONT-BLANC
            #pENGLAND-TRIP
            #pTOUR-ENGLAND-B
            #pEUROPEAN-CAPITOL-TRIP
            #pNORTH-AMERICAN-TRIP
            #pUSA-TRIP
            #pTOUR-WYOMING
            #pTOUR-CALIFORNIA-PARKS
            #pNORTH-AMERICAN-CAPITOL-TRIP
            #pTOUR-SWITZERLAND
            #pNATIONAL-PARK-CLIMBING-TRIP
            #pTOUR-FRANCE-B
            #pSKI-TOUR
            #pHAUTE-ROUTE-SKIING
            #pTOUR-DE-MONT-BLANC-SKIING
            #pHIKING-TRIP
            #pHAUTE-ROUTE-HIKING
            #pTOUR-DU-MONT-BLANC-HIKING
            #pTETONS-HIKING-AND-CLIMBING-TRIP
            #pRAINIER-HIKING-AND-CLIMBING-TRIP
            #pCULTURAL-TRIP
```

```
            #pTOUR-ITALY
            #pEUROPEAN-MUSEUM-TRIP
            #pTOUR-FRANCE-A
            #pTOUR-ENGLAND-A
            #pTOUR-ENGLAND-FRANCE-COMBO
            #pTOUR-ENGLAND-FRANCE-ITALY-COMBO
            #pNORTH-AMERICAN-MUSEUM-TRIP
            #pTOUR-WASHINGTON-DC

;;; The incremental recognition state confirms that only one optional plan,
;;; ALPINE-PEAKS, is left:

 MSNs = (#pCLIMBING-TRIP #pMULTIPLE-EUROPEAN-COUNTRIES)
 MGOs = (#pALPINE-PEAKS)
 MSOs = (#pALPINE-PEAKS)

;;; The End.
```

Appendix B

Proofs of Theorems

Theorem 1

Concepts *C1* and *C2* are *OTA-consistent* iff

1. No primitive of *C1* is disjoint from any primitive of *C2*
2. For every role R restricted by both *C1* and *C2*
   (a) The cardinality restrictions on R_{C1} and R_{C2} intersect
   (b) If at-least(R_{C1}) > 0 or at-least(R_{C2}) > 0, then value-restriction(R_{C1}) and value-restriction(R_{C2}) are OTA-consistent
   (c) $|$ fillers(R_{C1}) $\cup$ fillers(R_{C2}) $|$ $\leq$ minimum(at-most(R_{C1}), at-most(R_{C2}))
   (d) Every filler of R_{C1} is OTA-consistent with value-restriction(R_{C2}) and every filler of R_{C2} is OTA-consistent with value-restriction(R_{C1})

Proof

If: When all stated conditions are met, there can be no contradiction between the primitives of *C1* and *C2*, or between their roles, so *C1* $\wedge$ *C2* is satisfiable. □

Only If: Violation of any stated condition clearly renders *C1* $\wedge$ *C2* unsatisfiable. □

Theorem 2

Under CTA, the extensions of concepts *C1* and *C2* intersect iff they are CTA-consistent.

Proof

If: Given that *C1* and *C2* are CTA-consistent, we will show how to construct an individual belonging to both of their extensions. From Definition 17 on page 61, there are two direct cases of CTA-consistency:

1. When $C1 \mapsto C2$, as in Definition 16 on page 60, primitives($C1$) $\subseteq$ primitives($C2$) and restricted-roles($C1$) $\subseteq$ restricted-roles($C2$). We can construct an instance I of *C2* that instantiates *C1* as well:
   - Make every base concept of *C2* a base concept of I. Hence, primitives(I) $\supseteq$ primitives($C1$).
   - For every role R restricted by both *C1* and *C2*, restrict R_I as the conjunction of R_{C1} and R_{C2} by choosing the greater of their at-least restrictions, the lesser of their at-most restrictions, the conjunction of their value restrictions, and the union of their fillers. Clause 4 of Definition 16 guarantees that the result is satisfiable. Hence, I satisfies every role restriction on *C1*.
   - For every role R restricted by *C2* but not *C1*, restrict R_I exactly as R_{C2}.
2. When $C2 \mapsto C1$, the reasoning is essentially similar.

The third case of CTA-consistency is indirect. Considering Definition 20 on page 62, we can construct an instance I of *C3* that instantiates both *C1* and *C2* as well:

- Make every base concept of *C3* a base concept of I. Hence, primitives(I) $\supseteq$ primitives(*C1*) and primitives(I) $\supseteq$ primitives(*C2*).

- For every role R of *C3* restricted by *C1* and/or *C2*, correspondingly restrict R_I as the conjunction of R_{C3} with R_{C1} and/or R_{C2} by choosing the greatest of their at-least restrictions, the least of their at-most restrictions, the conjunction of their value restrictions, and the union of their fillers. Clause 3 of Definition 20 guarantees that the result is satisfiable. Hence, I satisfies every role restriction on *C1* and *C2*.

- For every role R restricted by *C3* but neither *C1* nor *C2*, restrict R_I exactly as R_{C3}.

In each case, individual I shows that the extensions of *C1* and *C2* intersect. □

Only If: Given that the extensions of *C1* and *C2* intersect under the CTA, we will show that *C1* and *C2* are CTA-consistent. From our assumption, some individual I must instantiate both *C1* and *C2*. Due to the CTA, I must bijectively instantiate some concept explicitly defined in the terminology. That concept must be either *C1*, *C2*, or some third concept, *C3*.

In case that concept is *C2*, we demonstrate that *C1* $\mapsto$*C2* according to Definition 16 on page 60:

1. By Definition 6 on page 51, primitives(I) $\equiv$ primitives(*C2*), and since I also instantiates *C1*, primitives(*C1*) $\subseteq$ primitives(I), so primitives(*C1*) $\subseteq$ primitives(*C2*).

2. The fact that I bijectively instantiates *C2* means *C2* is satisfiable, so no pair of concepts in primitives(*C2*) are disjoint. When primitives(*C1*) $\subseteq$ primitives(*C2*) as shown in item 1, this implies that no primitive of *C1* is disjoint from any (additional) primitive of *C2*.

3. By Definition 6, restricted-roles(I) $\equiv$ restricted-roles($C2$), and since I also instantiates $C1$, restricted-roles($C1$) $\subseteq$ restricted-roles(I), so restricted-roles($C1$) $\subseteq$ restricted-roles($C2$).

4. Considering that I instantiates both $C1$ and $C2$, it must simultaneously satisfy all of their role restrictions, so for every role R on restricted-roles($C1$), R_{C1} and R_{C2} are CTA-consistent.

Hence $C1$ and $C2$ are (directly) CTA-consistent.

In case the concept bijectively instantiated by I is $C1$, essentially the same reasoning demonstrates that $C2 \mapsto C1$. Again, $C1$ and $C2$ are (directly) CTA-consistent.

In the remaining case, I bijectively instantiates some concept $C3$ that is neither $C1$ nor $C2$. To start, we show that $C1 \mapsto C3$:

1. By Definition 6, primitives(I) $\equiv$ primitives($C3$). Since I also instantiates $C1$, primitives($C1$) $\subseteq$ primitives(I), so primitives($C1$) $\subseteq$ primitives($C3$).

2. The fact that I bijectively instantiates $C3$ means $C3$ is satisfiable, so no pair of concepts in primitives($C3$) are disjoint. When primitives($C1$) $\subseteq$ primitives($C3$) as shown in item 1, this implies that no primitive of $C1$ is disjoint from any (additional) primitive of $C3$.

3. By Definition 6, restricted-roles(I) $\equiv$ restricted-roles($C3$). Since I instantiates $C1$, restricted-roles($C1$) $\subseteq$ restricted-roles(I), so restricted-roles($C1$) $\subseteq$ restricted-roles($C3$).

4. Considering that I instantiates $C1$, $C2$, and $C3$, it must simultaneously satisfy all of their role restrictions, so for every role R on restricted-roles($C1$), R_{C1} and R_{C3} are CTA-consistent.

Now, considering Definition 20, we have:

1. *C1* $\mapsto$ *C3*, as just shown.

2. *C2* $\mapsto$ *C3*, by analogous reasoning.

3. As noted above, I simultaneously instantiates *C1* and *C2*, so for every role R that they both restrict, R_{C1} and R_{C2} are CTA-consistent.

Consequently, *C1* and *C2* are (indirectly) CTA-consistent.

In conclusion, if the extensions of *C1* and *C2* intersect, they must be CTA-consistent. $\square$

Theorem 3

Under CTA, individual I can be monotonically updated to instantiate concept C iff I and C are CTA-consistent.

Proof

If: Given that I and C are CTA-consistent, we show how to monotonically update I so that it instantiates C. There are direct and indirect cases of CTA-consistency:

1. In the direct case, $I \mapsto C$, as in Definition 22 on page 64. Then we can monotonically update I to instantiate C simply by adding any base concepts of C that I lacks, along with any role restrictions on C not implied by I's description.

2. In the case of indirect consistency, there exists a concept C' in accord with Definition 24 on page 66, and we can monotonically update I to simultaneously instantiate C and C':

   - We add any base concepts of C' that I lacks. Then primitives(I) $\equiv$ primitives(C') and since $C \mapsto C'$, we also know that primitives(I) $\supseteq$ primitives(C).
   - We add any role restrictions on either C or C' not implied by I's description. With respect to roles, the conjunction of restrictions from I and C' is satisfiable due to the first condition of Definition 24; the conjunction of restrictions from I and C is satisfiable due to its third condition. In case a role has restrictions from I, C' and C, all three conditions of Definition 24 interact to ensure they are mutually satisfiable.

Either way, we have shown that I can be monotonically updated to instantiate C. $\square$

Only If: When I can be monotonically updated to instantiate C, we will show that I and C must be CTA-consistent. Under the CTA, when updates to I are finished, I will bijectively instantiate at least one concept explicitly defined in the terminology. Such a concept is referred to as I's ultimate concept. Let the current and finished versions of I be denoted $I^{current}$ and $I^{finished}$, respectively. Notice that when updates to I are monotonic, we have:

1. primitives($I^{finished}$) $\supseteq$ primitives($I^{current}$)
2. restricted-roles($I^{finished}$) $\supseteq$ restricted-roles($I^{current}$)
3. For every role R on restricted-roles($I^{current}$)
   (a) at-least($R_{I^{finished}}$) $\geq$ at-least($R_{I^{current}}$)
   (b) at-most($R_{I^{finished}}$) $\leq$ at-most($R_{I^{current}}$)
   (c) value-restriction($R_{I^{finished}}$) $\Rightarrow$ value-restriction($R_{I^{current}}$)
   (d) fillers($R_{I^{finished}}$) $\supseteq$ fillers($R_{I^{current}}$)

If C is an ultimate concept of I, then $I^{finished} \mapsto C$ by definition. From the preceding relationships between $I^{current}$ and $I^{finished}$, it is readily apparent that $I^{current} \mapsto C$ too. Hence, I and C are (directly) CTA-consistent.

On the other hand, if C is not an ultimate concept of I, then some other concept C' is. We will show that $I^{current}$ is indirectly CTA-consistent with C via C' in accordance with Definition 24 on page 66:

1. Since C' is an ultimate concept, $I^{finished} \mapsto C'$ by definition. Given the preceding relationships between $I^{current}$ and $I^{finished}$, it is readily apparent that $I^{current} \mapsto C'$.
2. The following four results show that $C \mapsto C'$ according to Definition 16 on page 60:

(a) From Definition 6 on page 51, primitives($I^{finished}$) $\equiv$ primitives(C'). Since $I^{finished}$ also instantiates C, primitives($I^{finished}$) $\supseteq$ primitives(C), and so primitives(C) $\subseteq$ primitives(C').

(b) The fact that $I^{finished}$ bijectively instantiates C' means that C' is satisfiable, so no pair of concepts in primitives(C') are disjoint. When primitives(C) $\subseteq$ primitives(C') as just demonstrated, this implies that no primitive of C is disjoint from any (additional) primitive of C'.

(c) From Definition 6, restricted-roles($I^{finished}$) $\equiv$ restricted-roles(C'). Since $I^{finished}$ also instantiates C, restricted-roles($I^{finished}$) $\supseteq$ restricted-roles(C), and so restricted-roles(C) $\subseteq$ restricted-roles(C').

(d) From Definition 6, for every role R on restricted-roles($I^{finished}$), $R_{I^{finished}}$ satisfies every restriction of $R_{C'}$. Since $I^{finished}$ also instantiates C, for every role R on restricted-roles(C), $R_{I^{finished}}$ satisfies every restriction of R_C. Thus, $I^{finished}$ simultaneously satisfies every role of C and every role of C', and since we just showed that restricted-roles(C) $\subseteq$ restricted-roles(C'), we conclude that for every role R on restricted-roles(C), R_C and $R_{C'}$ are CTA-consistent.

3. Given that $I^{finished}$ instantiates C, we know that for every role R restricted by both $I^{finished}$ and C, $R_{I^{finished}}$ and R_C are CTA-consistent. Recall that when updates to I are monotonic, restricted-roles($I^{current}$) $\subseteq$ restricted-roles($I^{finished}$). Recall also that for every role R on restricted-roles($I^{current}$), the restrictions on $R_{I^{current}}$ are more general than those on $R_{I^{finished}}$. It follows that for every role R currently restricted by both $I^{current}$ and C, $R_{I^{current}}$ and R_C are CTA-consistent.

These three results establish that I is (indirectly) CTA-consistent with C.

In conclusion, if I can be monotonically updated to instantiate concept C, then I and C are CTA-consistent. $\square$

Theorem 4

Under the CTA and with an augmented terminology, individual I can be monotonically updated to instantiate concept C iff I and C are CTA-consistent using Definition 27 for indirect consistency instead of Definition 24.

Proof

If: There are two cases. The first case, where $I \mapsto C$, is the same as in Proof 3. In the second case, I can be monotonically updated to instantiate some concept C' as described in Proof 3, and C' is subsumed by C. Thus, I can be monotonically updated to instantiate C as well. $\Box$

Only If: Proof by contradiction. With an augmented terminology, it is easy to see that if I instantiates C, it must instantiate a subsumee of C (inclusive). However, assuming that I and C are not CTA-consistent using Definition 27 for indirect consistency, there is no direct consistency inference from I to C or to any subsumee of C. With only monotonic updates under CTA, it was shown in the proof of Theorem 3 that there must always be a direct consistency inference from I to its ultimate concept(s). Since no ultimate concept of I can be a subsumee of C (inclusive), we conclude that I cannot be monotonically updated to instantiate C, a contradiction. $\Box$

Theorem 5

Network concept *N1* subsumes network concept *N2* iff there exists a structural subsumption mapping from *N1* to *N2* (assuming that closure of *N2* is complete).

Proof

If: Clearly, the subsumption mapping demonstrates that any instance in the extension of *N2* is also in the extension of *N1*. □

Only If: We assume that closure of *N2* is complete. When there is no subsumption mapping from *N1* to *N2*, we will see that *N2*'s extension is not a subset of *N1*'s extension, so the notion that *N1* subsumes *N2* is contradictory. There are two cases to consider. First, the nodes of *N2* may not permit a mapping from *N1* with a distinct subsumee for each node in *N1*. Then *N1* contains at least one node without a counterpart *N2*. Second, the nodes of *N2* may permit such a mapping, but not so that every arc between a pair of nodes in *N1* subsumes the corresponding arc in *N2*. Then *N1* contains at least one arc that mandates a relationship not required by its counterpart in *N2*. In either case, there clearly exists an instantiation of *N2* that is not an instantiation of *N1*, hence the contradiction. □

Theorem 6

Subsumption mapping between QM networks is NP-complete.

Proof

The problem is in NP because a nondeterministic algorithm can guess a subsumption mapping and check it in polynomial time. Clearly, if the subsumer has n nodes, this entails n node subsumption tests. It is also trivial to check that no two nodes in the subsumer are mapped to the same node in the subsumee. Since there are two directed qualitative arcs and four directed metric arcs between every pair of nodes, half of which are redundant, subsumption mapping entails no more than $O(n^2)$ arc subsumption tests.

There is a polynomial time transformation from subgraph isomorphism, which is NP-complete [Garey and Johnson, 1979], to subsumption mapping between QM network concepts. Digraphs $G1 = (V1, E1)$ and $G2 = (V2, E2)$ are transformed into QM constraint networks *N1* and *N2*, respectively, as follows:

- Associate the primitive concept VERTEX with each element of *V1* and each element of *V2*.
- Associate the symmetric qualitative temporal constraint *before* $\vee$ *after* with each element of *E1* and each element of *E2*.

Then, it is evident that *G2* contains a subgraph isomorphic to *G1* just in case there is a subsumption mapping from *N1* to *N2*. □

Corollary 1

Subsumption mapping between complete QM networks is NP-complete.

Proof

The proof is similar and uses the same transformation from subgraph isomorphism. We make the following observation: the fact that node u is *before* $\vee$ *after* node v and that node v is *before* $\vee$ *after* node w implies nothing about the relationship between u and w. Therefore, after *N1* and *N2* are completed, every arc that was labeled *before* $\vee$ *after* during the transformation is still labeled the same way. Every other qualitative arc is labeled with the universal qualitative temporal constraint. (Thus, *before* $\vee$ *after* corresponds to *connected* in the original graph, and the universal qualitative temporal constraint corresponds to *unconnected*.) Again, it can be seen that *G2* contains a subgraph isomorphic to *G1* just in case there is a subsumption mapping from *N2* to *N1*. $\square$

Theorem 7

Subsumption mapping between complete QME networks is NP-complete.

Proof

In Theorem 6, the problem size for QM networks was characterized by the number of nodes. For structural subsumption, the problem size for QME networks is characterized by the number of nodes plus the number of binary equality constraints. This problem is still in NP because we can also check subsumption for each of the subsumer's equality constraints in constant time, assuming that the network completion process has ensured the presence of equality constraints for any pair of operands that are necessarily identical. For non-structural subsumption (taking into account part 3(b) of Definition 44) the problem size for QME networks also includes the size of the concepts associated with each node, as the sets of fillers to be compared are independent of the number of nodes and binary equality constraints in a QME network. Obviously, set equality for fillers can be computed in time quadratic in the size of the sets.

QME network subsumption can be reduced from subgraph isomorphism because all QM network subsumption problems are QME network subsumption problems, and the proof of Theorem 6 showed that the former can be reduced from subgraph isomorphism. □

Theorem 8

Network individual I instantiates network concept C iff there exists an instantiation mapping from C to I.

Proof

Essentially similar to Theorem 5.

Theorem 9

Under CTA, the extensions of QME network concepts *N1* and *N2* intersect iff they are CTA-consistent.

Proof

If: Given that *N1* and *N2* are CTA-consistent, we will show how to construct an individual belonging to both of their extensions. From Definition 65 on page 175, there are two direct cases of CTA-consistency:

1. When $N1 \mapsto N2$, as in Definition 65 on page 175, we can construct an instance I of *N2* that instantiates *N1* as well, based on any consistency mapping from *N1* to *N2* (recall there may be more than one). In the following, refer to Figure B.1 for an illustration, where dashed arrows map nodes under a consistency mapping and solid arrows map nodes under an instantiation mapping:

    (a) For every node *n2* of *N2* with associated concept *C2* mapped from node *n1* of *N1* with associated concept *C1*, create distinct node of I with an associated individual defined as $C1 \wedge C2$

    (b) For every (additional) node of *N2* with associated concept *C2* not mapped from a node of *N1*, create a distinct node of I with associated concept *C2*

    (c) For every ordered pair of nodes *m2* and *n2* of *N2* mapped from nodes *m1* and *n1* of *N1*, respectively, along with their counterparts h and i in I, respectively

        i. For every temporal (qualitative or metric) arc from *m2* to *n2*, create a corresponding arc from h to i whose constraint is the conjunction of the constraints on the arc from *m2* to *n2* and the corresponding arc from *m1* to *n1*

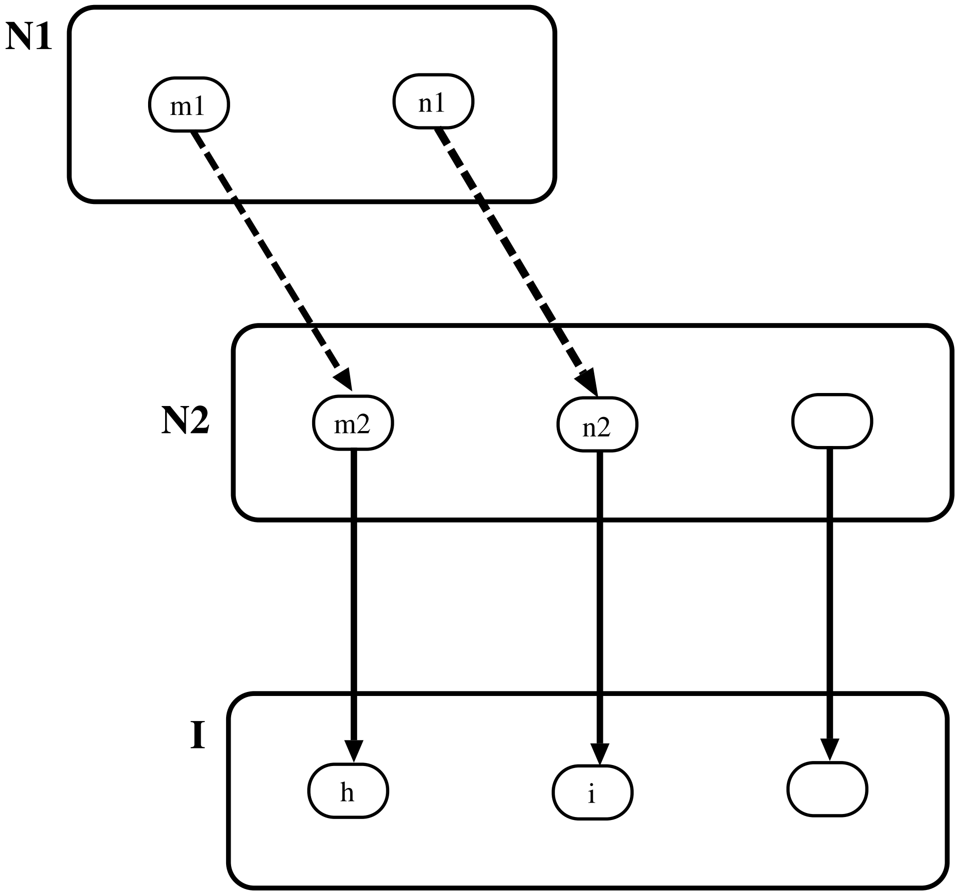


Figure B.1: An Illustration of Directly CTA-consistent Networks

ii. For every equality arc from *m1* to *n1* create an identical equality arc from h to i

iii. For every equality arc from *m2* to *n2* create an identical equality arc from h to i

(d) For every pair of nodes *m2* and *n2* of *N2* not both mapped from nodes in *N1*, and for every arc between *m2* and *n2*, create a corresponding arc between h and i whose constraint is identical

2. When $N2 \mapsto N1$, the reasoning is essentially similar.

The third case of CTA-consistency is indirect. Considering Definition 66 on page 178, we can construct an instance I of *N3* that instantiates both *N1* and *N2* as well,

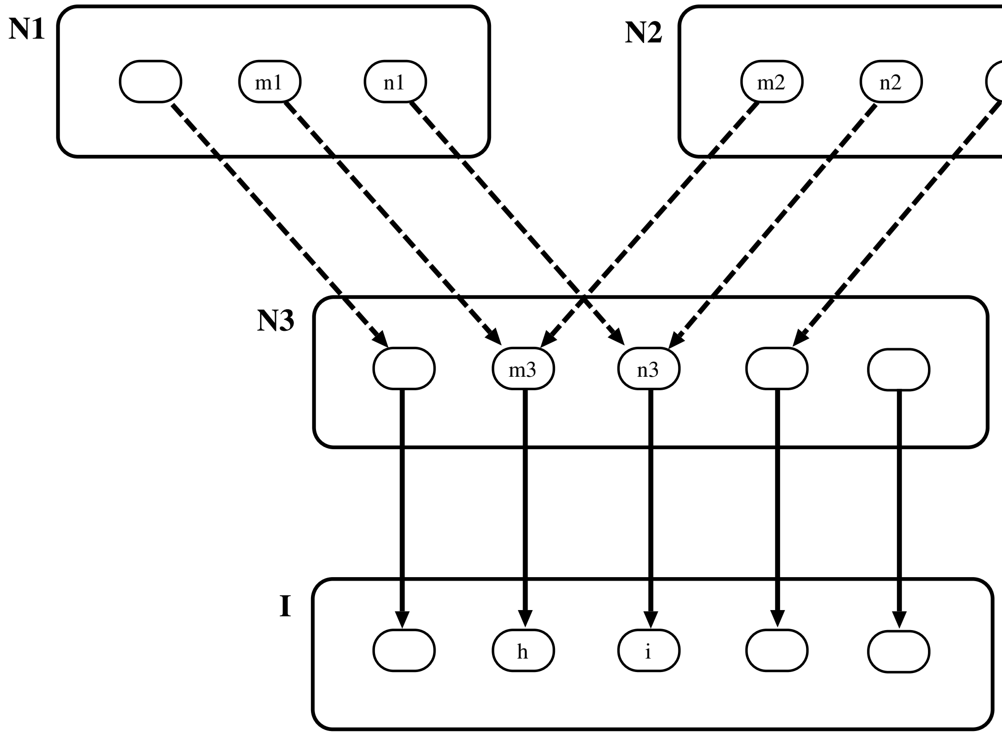


Figure B.2: An Illustration of Indirectly CTA-consistent Networks

based on any consistency mapping from *N1* to *N3* together with any consistency mapping from *N2* to *N3* (in the following, refer to Figure B.2 for an illustration, where dashed arrows map nodes under a consistency mapping and solid arrows map nodes under an instantiation mapping:

1. For every node *n3* of *N3* with associated concept *C3* mapped from node *n1* of *N1* with associated concept *C1* and/or mapped from node *n2* of *N2* with associated concept *C2*, create a distinct node of *I* with an associated individual defined as the conjunction of *C3* with *C1* and/or *C2*

2. For every (additional) node of *N3* with associated concept *C3* neither mapped from a node of *N1* nor mapped from a node of *N2*, create a distinct node of *I* with associated concept *C3*

3. For every ordered pair of nodes *m3* and *n3* of *N3* mapped from nodes *m1* and *n1* of *N1*, respectively, and/or mapped from nodes *m2* and *n2* of *N2*,

respectively, along with their counterparts h and i in I, respectively

(a) For every temporal (qualitative or metric) arc from *m3* to *n3*, create a corresponding arc from h to i whose constraint is the conjunction of the constraints on the arc from *m3* to *n3* with the corresponding arc from *m1* to *n1* and/or the corresponding arc from *m2* to *n2*

(b) For every equality arc from *m1* to *n1* create an identical equality arc from h to i

(c) For every equality arc from *m2* to *n2* create an identical equality arc from h to i

(d) For every equality arc from *m3* to *n3* create an identical equality arc from h to i

4. For every pair of nodes *m3* and *n3* of *N3* neither both mapped from nodes in *N1* nor both mapped from nodes in *N2*, and for every arc between *m3* and *n3*, create a corresponding arc between h and i whose constraint is identical

In each case, individual I shows that the extensions of *N1* and *N2* intersect. □

Only If: Given that the extensions of *N1* and *N2* intersect under the CTA, we will show that *N1* and *N2* are CTA-consistent. From our assumption, some individual I must instantiate both *N1* and *N2*. Due to the CTA, I must bijectively instantiate some network concept explicitly defined in the library. That concept must be either *N1*, *N2*, or some third network concept, *N3*.

In case that network concept is *N2*, we demonstrate that *N1* $\mapsto$*N2* according to Definition 64 on page 175. Given that I instantiates both *N1* and *N2*, each node of *N1* is instantiated by a distinct node of I, and likewise each node of *N2* is instantiated by a distinct node of I. We simply specify the mapping from *N1* to *N2* as follows: For every node i of I mapped from node *n1* of *N1* and also mapped from node *n2* of *N2*, we map from *n1* to *n2*. Then, we have met the requirements of Definition 64 as follows:

1. The existence of i ensures that $n1$ is CTA-consistent with $n2$

2. For every pair of nodes h and i of I mapped from $m1$ and $n1$ of $N1$ and also mapped from $m2$ and $n2$ of $N2$, the arcs between h and i jointly satisfy the arcs between $m1$ and $n1$ and the arcs between $m2$ and $n2$. Consequently, every temporal (qualitative or metric) arc between a pair of nodes in $N1$ is consistent with the corresponding temporal arc in $N2$.

3. Considering the relationships among nodes in $N1$ and $N2$ under the mapping we have constructed, based on the existence of I, it can be seen that the operands of every equality constraint in $N1$ have CTA-restriction-consistent counterparts in $N2$, and conversely.

Hence $N1$ and $N2$ are (directly) CTA-consistent.

In case the concept bijectively instantiated by I is $N1$, essentially the same reasoning demonstrates that $N2 \mapsto N1$. Again, $N1$ and $N2$ are (directly) CTA-consistent.

In the remaining case, I bijectively instantiates some network concept $N3$ that is neither $N1$ nor $N2$. Considering Definition 66, we show that $N1$ and $N2$ are (indirectly) CTA-consistent.

1. $N1 \mapsto N3$, just as we showed $N1 \mapsto N2$ in the direct case (simply substitute $N3$ for $N2$).

2. $N2 \mapsto N3$, by analogous reasoning.

3. As noted above, I simultaneously instantiates $N1$ and $N2$, so it simultaneously satisfies all of their constraints.

Consequently, $N1$ and $N2$ are (indirectly) CTA-consistent.

In conclusion, if the extensions of $N1$ and $N2$ intersect, they must be CTA-consistent. □

Theorem 10

Under CTA, network individual I can be monotonically updated to instantiate network concept C iff I and C are CTA-consistent.

Proof

If: Given that I and C are CTA-consistent, we show how to monotonically update I so that it instantiates C. There are direct and indirect cases of CTA-consistency:

1. In the direct case, $I \mapsto C$. Then we can monotonically update I to instantiate C by extending any mapping from I to C as follows:

   (a) For every node n_C of C with associated concept c mapped from node n_I of I with associated individual i, redefine i by conjoining c with its current definition.

   For every node of C with associated concept c not mapped from I, add a corresponding node to I with associated individual i defined as c.

   (b) For every every pair of nodes in C, apply the constraints between those nodes to the corresponding nodes in I.[1]

2. In the case of indirect consistency, there exists a network concept C' in accord with Definition 70 on page 185, and we can monotonically update I to simultaneously instantiate C and C'. We extend a mapping from I to C' in coordination with a mapping from C to C' as follows (recall there may be more than one mapping in each case):

   (a) Consider every node $n_{C'}$ of C' with associated concept c' mapped from node n_I of I with associated individual i. If $n_{C'}$ is also mapped from node n_C of C with associated concept c, redefine i by conjoining c and

[1] One or both of the corresponding nodes may have been added to I in the preceding step.

c' with its current definition. Otherwise, redefine i by conjoining c' with its current definition.

(b) For every node $n_{C'}$ of C' with associated concept c' not mapped from a node of I, create a corresponding node n_I of I with associated individual i. If $n_{C'}$ is mapped from node n_C of C with associated concept c, define i as $c \wedge c'$. Otherwise, define i as c'.

(c) For every every pair of nodes in C', apply the constraints between those nodes to the corresponding nodes in I.[2] In case the pair of nodes in C' are both mapped from nodes in C, also apply the constraints between the nodes from C to the corresponding nodes in I.

Either way, we have shown that I can be monotonically updated to instantiate C. □

Only If: When I can be monotonically updated to instantiate C, we will show that I and C must be CTA-consistent. Under the CTA, when updates to I are finished, I will bijectively instantiate at least one network concept explicitly defined in the terminology. Such a concept is referred to as I's ultimate concept. Let *nodes(X)* denote the set of nodes in network concept or individual X, and let *realization(I)* denote a concept whose definition is identical to that of I.[3] Also, let the current and finished versions of I be denoted with superscripts *current* and *finished*, respectively. Notice that when updates to I are monotonic, we have:

1. $\text{nodes}(I^{finished}) \supseteq \text{nodes}(I^{current})$
2. For every node n on $\text{nodes}(I^{current})$, $\text{realization}(n_{I^{finished}}) \Rightarrow \text{realization}(n_{I^{current}})$
3. For every pair of nodes on $\text{nodes}(I^{current})$, and for every arc A between that pair, $A^{finished} \Rightarrow A^{current}$.

[2]One or both of the corresponding nodes may have been added to I in the preceding step.
[3]In other words, *realization(I)* is the most specific possible concept which I can instantiate.

If C is an ultimate concept of I, then $I^{finished} \mapsto C$ by definition. From the preceding relationships between $I^{current}$ and $I^{finished}$, it is readily apparent that $I^{current} \mapsto C$ too. Hence, I and C are (directly) CTA-consistent.

On the other hand, if C is not an ultimate concept of I, then some other concept C' is. We will show that $I^{current}$ is indirectly CTA-consistent with C via C' in accordance with Definition 70:

1. Since C' is an ultimate concept, $I^{finished} \mapsto C'$ by definition. Given the preceding relationships between $I^{current}$ and $I^{finished}$, it is readily apparent that $I^{current} \mapsto C'$.

2. Since $I^{finished}$ bijectively instantiates C' and also instantiates C, it is easy to see that $C \mapsto C'$ according to Definition 65 on page 175.

3. The fact that $I^{finished}$ bijectively instantiates C' and simultaneously instantiates C demonstrates that for every node c' of C' mapped from node i of $I^{finished}$ and also mapped from node c of C, i and c are CTA-consistent. Recall that nodes($I^{finished}$) $\supseteq$ nodes($I^{current}$). Recall also that for every node n on nodes($I^{current}$), realization($n_{I^{finished}}$) $\Rightarrow$ realization($n_{I^{current}}$). It follows that for every node c' of C' mapped from node i of I and also mapped from node c of C, i and c are CTA-consistent.

4. Furthermore, the fact that $I^{finished}$ bijectively instantiates C' and simultaneously instantiates C demonstrates that the conditions of clause 4 of Definition 65 are met once I is finished. Remember that for every pair of nodes on nodes($I^{current}$), and for every arc A between that pair, $A^{finished} \Rightarrow A^{current}$. It follows that those conditions are also met by $I^{current}$.

These four results establish that I is (indirectly) CTA-consistent with C.

In conclusion, if I can be monotonically updated to instantiate concept C, then I and C are CTA-consistent. $\square$

Theorem 11

Under the CTA and with an augmented library, network individual I can be monotonically updated to instantiate network concept C iff I and C are CTA-consistent using Definition 73 instead of Definition 71.

Proof

The reasoning follows the proof of Theorem 4.

If: There are two cases. The first case, where $I \mapsto C$, is the same as in Proof 10. In the second case, I can be monotonically updated to instantiate some concept C' in similar fashion, and C' is subsumed by C. Thus, I can be monotonically updated to instantiate C as well. □

Only If: Proof by contradiction. With an augmented terminology, it is easy to see that if I instantiates C, it must instantiate a subsumee of C (inclusive). However, assuming that I and C are not CTA-consistent using Definition 73 for indirect consistency, there is no direct consistency mapping from I to C or to any subsumee of C. With only monotonic updates under CTA, it was shown in the proof of Theorem 10 that there must always be a direct consistency mapping from I to its ultimate concept(s). Since no ultimate concept of I can be a subsumee of C (inclusive), we conclude that I cannot be monotonically updated to instantiate C, a contradiction. □

Theorem 12

Under perfect observation, CTA, and with an augmented library, network individual I can be monotonically updated to instantiate network concept C iff

1. $I \hookrightarrow C$, or
2. I is indirectly CTA-consistent with C according to Definition 79.

Proof

If: Given that I and C are CTA-consistent, we show how to monotonically update I so that it instantiates C. There are two cases.

1. In the first case, $I \hookrightarrow C$ as in Definition 78 on page 200. It is clear that I can be extended to instantiate C, by adding nodes instantiating the remaining nodes of C as well as corresponding arcs instantiating the remaining arcs of C.
2. In the second case, there exists a network concept C' in accord with Definition 79 on page 202. By reasoning analogous to that of item 1, I can be extended to instantiate C'. Since C subsumes C', I then instantiates C as well.

□

Only If: Under perfect observation, it is easy to see that there must always be a partial instantiation mapping from I to its ultimate concept or concepts. Thus, only those concepts enjoying a partial instantiation mapping from I can be an ultimate concept. Hence, assuming that the library has been augmented, only those concepts and their subsumers can be instantiated by I. □

Index

abstract
- concept 52
- plan 162

augmented
- library 191
- terminology 67

bijective instantiation
- mapping 156
- of a concept 51
- of a network concept 161

classification 2, 26
closed library assumption 165
closed terminology assumption .. 57
closed world assumption 37
coherent concept 25
completion
- concept 27
- network 107

concrete
- concept 52
- plan 161

consistent
- (see also CTA-consistent)
- arcs 171
- concepts 26
- equality constraints 170
- metric constraints 169
- qualitative constraints 169

CTA-consistency mapping
- network concept to individual 181
- network concepts 175

CTA-consistent
- arcs 171
- concepts 63
- individual and concept 66
- individuals 168
- network concepts
  - (general) 167
  - with monotonic observation 186
  - with perfect observation .. 202
- network individual and concept
  - perfect observation 202
- nodes 168
- QME network concepts 179
- roles 62

CTA-restriction-consistent (roles) 61
direct CTA-consistency
- concepts 61
- individual and concept 64
- network concepts 175
- network individual
  - and concept 183

direct CTA-consistency inference ($\mapsto$)
- concept to concept 60
- individual to concept 64

disjoint concepts 26
equality
- arc 101
- constraint 102

equivalent concepts 26
finished
- individual 52
- network individual 162

impossible
- concept 59
- plan 166

incoherent concept 25
indirect CTA-consistency
- concept to concept 62
- individual to concept

(augmented) 68
individual to concept
(unaugmented) 66
network concept to concept . 178
network individual to concept
(augmented)
monotonic observation ... 192
perfect observation 202
network individual to concept
(unaugmented) 185
instantiation 2
node 155
knowledge base 2
least common subsumer 34
library (of plans) 6
macro plan 112
metric
arc 100
constraint 43
monotonic update assumption ... 58
most general optional (MGO)
concept 72
plan 194
most specific necessary (MSN)
concept 72
plan 194
most specific optional (MSO)
concept 72
plan 194
necessary
concept 59
plan 166
normalization 27
observation network 160
optional
concept 59
plan 166
partial instantiation mapping .. 200
plan 103
body 8, 103
concept 8, 103
individual 107
library 6
macro 112
network 104
recognition problem 159
potential instantiation 49
predictive recognition 5
primitives
of a concept 24
of an individual 24
QME network 98
qualitative
arc 98
constraint 38
realization 26
recognition 2, 26
predictive 5
restricted roles 23
structural
consistency mapping 171
instantiation mapping 155
partial instantiation mapping 198
subsumption 28
subsumption mapping 121
subsumption 2, 25
arc 120
equality constraint 120
inequality relation 119
metric constraint 119
network concept 117
node 118
qualitative constraint 118
subsumption mapping 124
terminology 2, 16
ultimate
concept 58
plan 165
vacuous concept 25