Terminological Constraint Network Reasoning and its Application to Plan Recognition (Thesis Proposal)

Columbia University
Department of Computer Science
Technical Report CUCS-027-93

Robert Weida

September 2, 1993

Contents

List of Figures

Abstract

Terminological systems in the tradition of KL-ONE are widely used in AI to represent and reason with concept descriptions. They compute subsumption relations between concepts and automatically classify concepts into a taxonomy having well-founded semantics. Each concept in the taxonomy describes a set of possible instances which are a superset of those described by its descendants. One limitation of current systems is their inability to handle complex compositions of concepts, such as constraint networks where each node is described by an associated concept. For example, plans are often represented (in part) as collections of actions related by a rich variety of temporal and other constraints. The T-REX system integrates terminological reasoning with constraint network reasoning to classify such plans, producing a "terminological" plan library. T-REX also introduces a new theory of plan recognition as a deductive process which dynamically partitions the plan library by modalities, e.g., necessary, possible and impossible, while observations are made. Plan recognition is guided by the plan library's terminological nature. Varying assumptions about the accuracy and monotonicity of the observations are addressed. Although this work focuses on temporal constraint networks used to represent plans, terminological systems can be extended to encompass constraint networks in other domains as well.

1 Introduction

Terminological systems in the tradition of KL-ONE and NIKL [Brachman and Schmolze, 1985, Woods and Schmolze, 1992] are widely used to represent and reason with conceptual knowledge required by "intelligent" software applications. Examples include database querying [Beck and Gala, 1989], financial marketing [Apte *et al.*, 1992], software information systems [Devanbu *et al.*, 1991], and multimedia explanation of repair and maintenance procedures [Feiner and McKeown, 1990]. However, contemporary terminological systems are limited by their inability to handle complex compositions of concepts. Therefore, we propose to extend their scope and utility via *terminological constraint networks*, whose nodes are described by associated concepts. Noting that much artificial intelligence research has involved reasoning with plans, we will focus on plans which are described in terms of constraints on their constituent actions and temporal constraints between their actions. We will employ a methodology that supports creation, management and utilization of *terminological plan libraries*. A major thrust of plan-based reasoning is plan recognition, which seeks to infer underlying plans from observed actions. We believe that development of practical plan recognition technology can foster more responsive user interfaces. Therefore, this proposal also introduces a new, terminological approach to plan recognition.

The following section provides an overview of core issues in our proposed research. Section 3 provides background information on terminological knowledge representation and on temporal constraint reasoning. Section 4 presents our results to date. Section 5 reviews related work in extending terminological knowledge representation and in plan recognition. In Section 6, open research issues are described along with possible ways to address them. Section 7 sketches one potential application and points out several others. Evaluation of our ultimate results is considered in Section 8. Finally, Section 9 concludes by recapping our proposal, summarizing our contributions, and establishing ongoing research priorities. Appendix A defines a sample plan library used for many of our examples, Appendix B speculates on the relevance of our work to the problem of plan synthesis, and Appendix C contains proofs of theorems.

2 Overview

Terminological knowledge representation (TKR) systems support automatic classification of definitional taxonomies based on subsumption inferences [Brachman and Schmolze, 1985, MacGregor, 1991, Weida, 1991, Woods and Schmolze, 1992]. In a *definitional taxonomy*, each class describes a set of possible instances which are a superset of those described by its descendant classes. Many systems compute subsumption (subset relationships) between classes according to the structure of their definitions, i.e., *structural subsumption*. Thus, classification via structural subsumption endows a taxonomy with formal meaning. Classification ensures that the proper location of any class within the taxonomy is uniquely determined from its definition. This in turn supports automatic detection of redundant, inconsistent and vacuous definitions. Classification also facilitates incremental construction of taxonomies, enforcement of semantics, type-checking and pattern matching. For elaboration on these benefits, see [Brachman and Schmolze, 1985, MacGregor, 1991, Weida, 1991, Woods, 1986].

While terminological systems are widely used in many application areas, to date they have focused on representing structured conceptual descriptions, or *concepts*. A critique of contemporary TKR which argues for greater expressiveness is [Doyle and Patil, 1991]. One limitation of current terminological systems, e.g., BACK [von Luck *et al.*, 1987], CLASSIC [Borgida *et al.*, 1989], K-Rep [Mays *et al.*, 1991b], KRIS [Baader and Hollunder, 1991] and LOOM [MacGregor and Bates, 1987], is their inability to represent and reason with complex compositions of concepts such as constraint networks where each node is described by a concept.

Plans are central to many areas of AI. We propose a knowledge representation system that computes subsumption among plans represented as collections of temporally related actions. In particular, we employ a plan representation which builds on temporal constraint networks in the style of [Allen, 1983]. We show how to extend the ideas of structural subsumption and classification found in TKR systems to automatically organize these plans into a definitional taxonomy which constitutes a "terminological" plan library. The advantages obtained from representing knowledge in standard terminological systems are achieved here as well. Our approach is similar in spirit to previous work on plan subsumption [Devanbu and Litman, 1991, Wellman, 1990], but provides a much richer temporal representation language. We also use our notion of constraint network subsumption to develop a new, terminological approach to plan recognition. While terminological plan systems have been applied in the areas of plan synthesis [Wellman, 1990] and plan retrieval [Devanbu and Litman, 1991], the application of terminological reasoning to the area of plan recognition has previously been unexplored.

The definitional plan taxonomy provides a natural basis to guide plan recognition. Many plan recognition systems infer agents' plans from their actions by searching libraries of possible plans for suitable (perhaps nondeductively inferred) matches. We introduce a new view of plan recognition as a process which dynamically partitions the plan library into modalities, e.g., *necessary*, *possible* and *impossible*, according to observations of the environment. We will also leverage the taxonomy's enforced semantics to minimize the number of plans that must be examined. Our approach unifies representation and reasoning work in plan recognition and terminological systems.

A prototype plan recognition system, called T-REX[1], serves as a testbed for our ideas. T-REX integrates and builds upon existing systems for TKR and temporal reasoning. It represents and reasons about actions and their constituents using K-Rep [Mays *et al.*, 1991a] and temporal relationships using MATS [Kautz and Ladkin, 1991]. A system diagram appears in Figure 1. When a plan is defined, T-REX checks its syntactic correctness, normalizes the definition by deriving implicit information, and classifies it in the plan library by means of subsumption tests against previously defined plans. When observations are presented, T-REX recognizes several sets of candidate plans corresponding to modalities like necessary, possible and impossible.

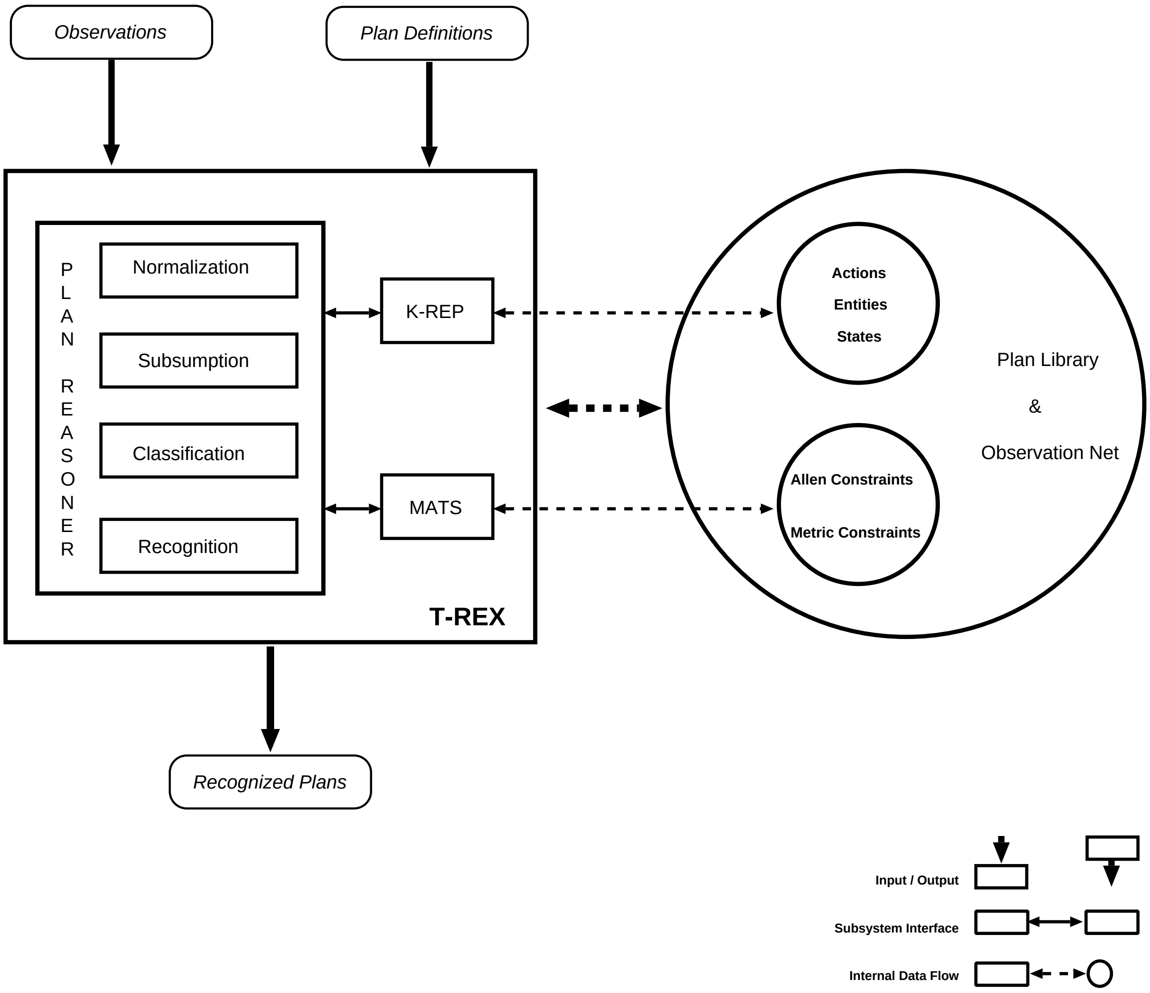

Figure 1: The T-REX System

Although we focus on temporal constraint networks used to represent plans, our methods apply to any kind of constraint network where we can define subsumption operations on the nodes and arcs, and hence on the networks themselves. We call such

[1]The name derives as follows: Terminological RECognition System → T-RECS → T-REX.

networks *terminological constraint networks*. In Section 6.7, we outline an approach to terminological reasoning with, and recognition of, N-dimensional spatial descriptions.

3 Foundations

Our work draws upon a terminological knowledge representation system to represent states and events such as actions, along with their constituents. We also employ a temporal reasoning system to manage information about qualitative and metric temporal relations. Therefore, we briefly introduce each of these technologies in turn.

3.1 Terminological Knowledge Representation

There is ample evidence that systems of the KL-ONE family are well-suited for representing the classes of actions which make up plans and, in turn, the objects that are acted upon, e.g., [Apte *et al.*, 1992, Devanbu *et al.*, 1991, Feiner and McKeown, 1990, Heinsohn *et al.*, 1992, Wellman, 1990]. The system we propose will build upon such a knowledge representation system and extend its capabilities to reason with structured plan descriptions.

3.1.1 Concept Languages and their Semantics

Terminological knowledge representation, which originated with KL-ONE [Brachman and Schmolze, 1985], is an object-centered approach in the tradition of semantic networks and frames. Contemporary systems include BACK [von Luck *et al.*, 1987], CLASSIC [Borgida *et al.*, 1989], K-Rep [Mays *et al.*, 1991b], KRIS [Baader and Hollunder, 1991], and LOOM [MacGregor and Bates, 1987]. Terminological systems share several distinguishing characteristics relevant to our discussion:

1. They are intended to support the definition of conceptual terms comprising a "terminology" and to facilitate reasoning about those terms. As such, they are distinct from assertional systems which make statements of fact based on a terminology.

2. Concept definitions generally specify both necessary and sufficient conditions for membership in the class denoted by the concept.

3. The concepts are arranged in a taxonomy based on strict subsumption, so that the features of each concept are inherited by its descendants without exception.

Thus, the proper location of any concept within the taxonomy can be uniquely determined from its definition by an automatic process known as *classification.*

4. Terminological systems restrict the expressiveness of their language to achieve relatively good performance.

Terminological languages support a taxonomy composed of *generic concepts*. Assertional languages record information about associated *individual concepts*, or *instances*. Generic concepts specify classes of entities whereas individual concepts specify unique entities that hold membership in at least one generic class. Generic concepts are *defined* if their specification provides both necessary and sufficient conditions for class membership; otherwise they are *primitive*. Primitive concepts are understood to entail certain sufficient conditions for class membership which are not or can not be expressed in the language. Concepts are defined principally in terms of *roles* which express potential binary relationships with another concept. Single-valued roles are called *attributes*. Role relations may be composed as *role chains*; they are expressed as sequences of role names. For notational convenience, a concept may be also be defined in terms of other concepts, called *superconcepts*, from which it inherits part of its definition. Terminological languages also support *role constraints* (or *role value maps*) which consist of an operator, such as $=$, $\subset$ or $\supset$, and a pair of role chains which designate its operands. For example, an ACADEMIC-DEPARTMENT concept might require the fillers of its SOFTWARE-FACULTY role to be a subset of the fillers of its FACULTY role.

3.1.2 Subsumption and Classification

Subsumption can be computed by a special purpose algorithm. One concept structurally subsumes another if and only if each feature of the first (recursively) subsumes some feature of the second. Thus, every role of the first concept must subsume a role of the second. This criterion assumes that concept definitions specify both necessary and sufficient conditions. Since a primitive concept lacks certain sufficient conditions, there is no basis for inferring that it subsumes another concept. Nonetheless, one can explicitly define a concept to be subsumed by a certain primitive or primitives. It need not be primitive itself. For example, even though the concepts for PERSON and FEMALE may be primitive, the concept for WOMAN may be fully defined as their logical conjunction.

Classification is a process which places concepts into a taxonomy according to subsumption relationships, i.e., it establishes the correct taxonomic links among concepts. Where should a concept be installed in a taxonomy? It belongs in exactly one place: underneath its most specific subsumers and above its most general subsumees.

The classifier employs an algorithm which adds concepts to the taxonomy one at a time, taking advantage of the already existing taxonomy's hierarchical organization. The process of classifying an individual concept, i.e., determining the most specific set of generic concepts which describe it, has sometimes been referred to distinctly as *realization* [Mark, 1981]. The classification process can be automated with reasonable efficiency. Schmolze and Lipkis formally specify the classification algorithm in KL-ONE [Schmolze and Lipkis, 1983]. NIKL's classifier is described in [Robins, 1986] and LOOM's is presented in [MacGregor, 1988]. Automatic classification is useful for incremental construction of a taxonomy, enforcing semantics, type checking, and pattern matching.

3.1.3 Computational Complexity

A seemingly insignificant extension to the expressiveness of a representation language may drastically compromise its worst case tractability. Brachman and Levesque focus their analysis on one such "crossover point" in the computation of subsumption relationships [Brachman and Levesque, 1984] [2]. They examine a typical language for which subsumption can be computed in $O(n^2)$ time. Next, they show that an apparently simple variant of that language is co-NP-complete. This leads to the conclusion that one must make careful choices in trading expressiveness for tractability. Moreover, there is no single best choice. Instead, different choices may complement one another nicely.

The result of Brachman and Levesque has practical significance because their co-NP-complete language is a subset of the terminological languages employed by such systems as KL-ONE. Nebel later showed that for another subset of the languages used in such systems as KL-ONE and BACK, subsumption is NP-hard [Nebel, 1988]. Patel-Schneider demonstrated that subsumption in NIKL and similar systems is undecidable as well [Patel-Schneider, 1989]. By showing that no complete algorithm for such languages is possible, his result underscored the trend towards sound but consciously incomplete subsumption and classification algorithms that was instigated by the intractability of subsumption in these languages. Schmidt-Schauss proved that a very simple concept language limited to conjunction of concepts, restrictions on values of roles, and role value maps with an equality operator is undecidable [Schmidt-Schauss, 1989] (however there is no problem when the chains are restricted to attributes). Recently, Nebel showed that subsumption in *terminologies*, which permit definitions to reference previously defined concepts, is inherently intractable (co-NP-complete) [Nebel, 1990]. Since our terminological reasoning with plans is founded on subsumption relations among their constituent concepts, these results have impact on our work. However, this impact is attenuated by the fact that in our work, we can

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

generally classify the underlying concepts in advance.

While intractability results for the concept subsumption problem are sobering, it must be emphasized that they are worst-case analyses. Under a set of reasonable assumptions, e.g., that concepts are composed from previously classified concepts, it can be argued that the cost of classifying a new concept is typically logarithmic in the size of the concept taxonomy [Woods, 1991]. We are hopeful that similar analysis may yield similar results for classification in our plan language. In particular, our optimism is reinforced by the normal case results of [Yen, 1990] as mentioned in Section 5.1.3.

3.2 Temporal Constraints

Allen, in his influential work on maintaining knowledge about temporal intervals, enumerated a total of seven primitive relationships, plus their inverses, that might hold between any ordered pair of intervals [Allen, 1983]. These primitives are illustrated in Figure 2. A *temporal constraint* records the possible relationships between a particular pair of intervals as a disjunctive subset of these primitive relationships. For example, the constraint {*before, after*} mandates temporal disjointness. As more information becomes available to the system, a temporal constraint may be refined by eliminating disjuncts. A *temporal network* consists of nodes that represent intervals and arcs that represent constraints between pairs of intervals.

Allen proposed a simple polynomial-time constraint propagation algorithm to *close* (or *normalize*) a temporal network by computing the implicit consequences of explicitly stated temporal constraints, i.e., a transitive closure. Allen's algorithm is an instantiation of the path consistency algorithm for constraint satisfaction [Montanari, 1974, Mackworth, 1977]. The practical performance of Allen-style constraint propagation can be enhanced by introducing a hierarchy of *reference intervals* to partition the space of temporal intervals, thereby limiting the scope of propagation. In fact, effective algorithms exist for automatic construction of reference interval hierarchies [Koomen, 1989].

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

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: Allen's Primitive Temporal Relations

1. Adopt an approximation algorithm such as Allen's, and live with the possible consequence that some plan subsumption relationships will remain undetected. Allen contends that his algorithm's inferences correspond to those which humans find natural [Allen, 1983]. There is a family of variations on Allen's algorithm which produce successively better approximations, but only at increasingly exorbitant cost [van Beek, 1989].

2. 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.

3. Restrict the expressiveness of the temporal constraints so that exact solutions can be obtained tractably. For example, Vilain, Kautz and van Beek [Vilain *et al.*, 1989] identified a subset of Allen's interval calculus derived from a point-based representation which admits complete polynomial-time constraint prop-

agation.

T-REX currently exercises the first option. We expect that practical experience will educate us as to the best choice.

A separate body of work 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]. Linear inequalities can express absolute times as constraints on a single time point, e.g.:

- TIME-POINT1 < 2000
- 2001 <= TIME-POINT2
- 1999 <= TIME-POINT3 < 2001

For notational convenience, the last of these examples combines two linear inequalities on the same time point. Linear inequalities can also express durations as the difference between two time points, e.g.:

- TIME-POINT5 - TIME-POINT4 < 2001
- 2010 <= TIME-POINT7 - TIME-POINT6 <= 2010

Sets of metric constraints form metric constraint networks. Determining the consistency of a metric constraint network is NP-hard [Dechter *et al.*, 1991].

Kautz and Ladkin designed a constraint reasoner which integrates reasoning over an Allen-style constraint network for intervals and a metric constraint network for the starting and ending points of those intervals [Kautz and Ladkin, 1991]. Thus, metric constraints convey durations of intervals, gaps between intervals, and so on. These ideas were implemented by Kautz in the MATS system which we are using in our research. We will have more to say about MATS in Section 4.1.3.

4 Results to Date

This section summarizes the present state of our work, some of which was reported in [Weida and Litman, 1992]. First, we introduce our plan representation language based on constraint networks. Next, we discuss our results in terminological reasoning with constraint networks, e.g., our plans. Then we present our new, terminological approach to plan recognition. Finally, we cover recent results on reasoning with metric temporal constraints and coreference constraints.

4.1 Plan Representation

Since we apply our ideas to plan-based reasoning, we must detail our plan representation. 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 TKR systems [Devanbu and Litman, 1991, Heinsohn *et al.*, 1992, Wellman, 1990]. T-REX is the only plan recognition system to do so. We use K-Rep to handle actions and their constituents; we will also use it for preconditions and effects of actions and plans. 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.

4.1.1 Basic Temporal Networks

Plan descriptions typically include preconditions, effects, a body composed of steps to carry out the plan, and some constraints. Following [Kautz, 1991], we will concentrate on plan recognition via plan bodies and their relationship via abstraction. We define a *plan body* as a collection of steps along with some temporal constraints between pairs of steps. Each step has a label and a type of action associated with it. Action types are represented by generic concepts in K-Rep [Mays *et al.*, 1991a], which we shall call *action concepts*. Together, these concepts constitute an *action taxonomy*. Action types can be thought of as atomic plans. We assume that the taxonomy includes every type of action which appears in a plan or is observed during plan recognition. Hence, action types are considered disjoint if there is no action type that they both subsume (note that subsumption is reflexive). K-Rep also represents instantiated action concepts, or *action instances*. When there is no ambiguity, we may simply refer to action concepts and action instances as actions. Each temporal constraint is an arbitrary disjunction of Allen's exhaustive set of 13 primitive temporal relationships between intervals [Allen, 1983] (see Figure 2). A *plan network* is a temporal constraint network [Allen, 1983] whose nodes correspond to time intervals when the steps of the plan's body occur. Hence, an action concept is associated with each node. Plans may be embedded as *macro actions* within other plans (but not within themselves). Any temporal constraint on a step with a macro action can be propagated to each substep within that macro by appropriate use of a constraint propagation algorithm such as in [Allen, 1983]. Song and Cohen have show how to do this [Song, 1991, Song and Cohen, 1991].

Following precedent, e.g., [Kautz, 1991, Song, 1991, Song and Cohen, 1991, van Beek and Cohen, 1991], we draw examples from the cooking domain. By convention, generic concept names are prefixed by C-. Names of instances are formed by concatenating a concept name with a unique number and stripping off the leading C-. All example plan descriptions in this proposal will be constructed from action concepts in the taxonomy shown in Figure 3. Observe that although we just use descriptive names, concepts and their instances are really represented in greater detail in K-Rep. For example, action concepts have roles such as *agent* and *object*. When a concept definition specifies necessary and sufficient conditions for class membership, K-Rep determines the concept's proper location within the taxonomy using classification. Such concepts are shown without an asterisk in Figure 3.

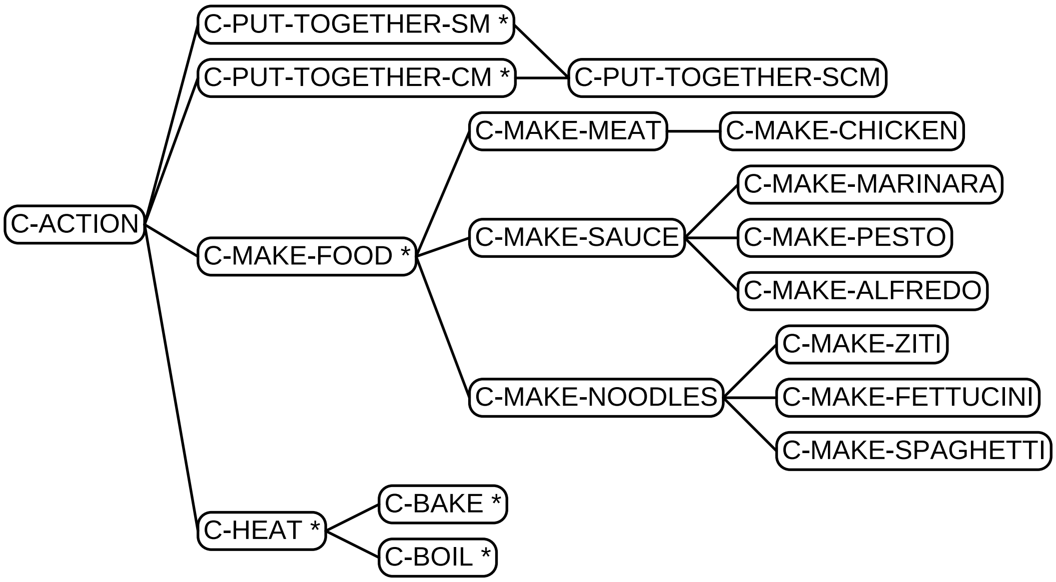

Figure 3: A Definitional Action Taxonomy

Our plan language is introduced in Appendix A. Below is a Lisp-style definition of a simple plan to assemble chicken marinara, taken from Appendix A, which is diagrammed in Figure 4. The labels of the steps are strictly for identification purposes; they do not convey temporal ordering. Also, note that there are alternate ways to state the same temporal information content. For example, instead of saying that STEP3 was *after* STEP2, we could have said that STEP2 was *before* STEP3.

```
(defplan ASSEMBLE-CHICKEN-MARINARA
  ((step1 c-make-chicken)
   (step2 c-make-marinara)
   (step3 c-put-together-cm))
  :allen-constraints
    ((step1 (before after) step2)
     (step1 (before) step3)
     (step3 (after) step2)))
```

To simplify our diagrams, we omit trivial constraints and some other constraints which can be inferred via transitivity, and we label nodes with the names of associated action concepts.

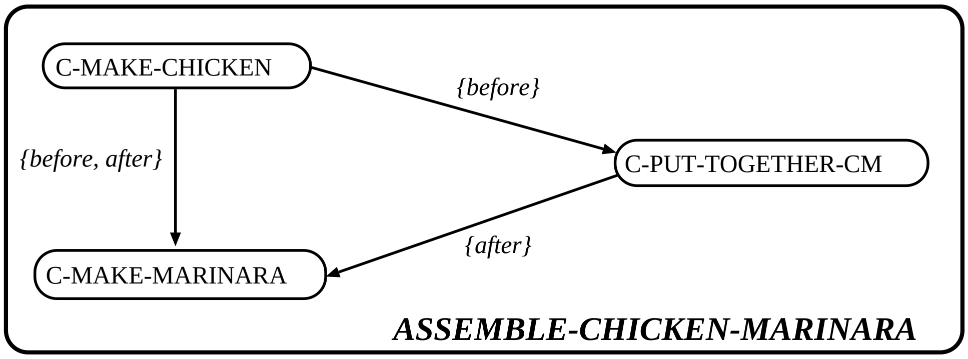

Figure 4: A Simple Plan Network

4.1.2 Plan Instances

Plans denote a set of possible *plan instances*, which have bodies composed of action instances and nondisjunctive temporal constraints chosen in accordance with the plan. When possible, we write networks as a sequence of nodes separated by constraints. The following might be an instance of the plan in Figure 4, since it satisfies its terminological and temporal constraints (after closure).

- MAKE-CHICKEN12 {*before*} MAKE-MARINARA54 {*before*} PUT-TOGETHER-CM27

4.1.3 Metric Temporal Constraints

MATS [Kautz and Ladkin, 1991] allows us to specify both Allen-style qualitative constraints on intervals and metric constraints on their starting and ending points.

Metric information accounts for durations of intervals and gaps between intervals, along with absolute times (which can be useful in the case of observations processed during plan recognition). For example, the following plan description incorporates metric constraints which restrict the gap between the two steps to exactly 5 time units and the duration of STEP2 to between 6 and 9 time units:

```
(defplan DEMO-METRIC-CONSTRAINTS
  ((step1 c-make-noodles)
   (step2 c-heat))
  :metric-constraints
   ((5 <= left step2 - right step1 <= 5)
    (6 <= right step2 - left step2 <= 9)))
```

The notation uses *left* and *right* to refer to starting and ending points of intervals, respectively.

Notice that metric constraints can imply Allen constraints and *vice versa*. For example, the first metric constraint in the preceding plan implies that STEP1 is *before* STEP2. MATS alternates between Allen 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-Allen and Allen-to-metric translation schemes [Kautz and Ladkin, 1991]. T-REX stores the final metric and Allen temporal constraints in the plan's internal representation.

4.1.4 Coreference Constraints

A T-REX plan description may include coreference constraints on roles of its constituent action concepts. A coreference constraint consists of an operator and any number of *operand specifiers* which designate the roles that provide its operands. Coreference constraints in plans resemble *role value maps* in standard TKR (see Section 3), but here they apply across concept definitions. This discussion is confined to coreference constraints with an equality operator, called equality constraints.[3] Each operand specifier consists of a *label* which identifies a step within the plan, and a *role-name* which identifies a role of the action concept associated with that step. We do not allow role composition in our operand specifiers, thus avoiding a potential source of undecidability when computing subsumption [Schmidt-Schauss, 1989]. The two coreference constraints in the following definition state that for any instantiation of the SOLO-BOIL-SPAGHETTI plan, the agents of the two steps must be the same,

[3] Alternatives include *inequality*, *subset* and *proper subset* operators.

as must their objects. More precisely, the filler of the AGENT role of the MAKE-SPAGHETTI instance associated with step S1 must be equal to the filler of the AGENT role of the BOIL instance associated with step S2 (and similarly for their object roles).

```
(defplan SOLO-BOIL-SPAGHETTI
  ((s1 c-make-spaghetti)
   (s2 c-boil))
  :allen-constraints ((s1 (before meets) s2))
  :coref-constraints ((equal (agent s1) (agent s2))
                      (equal (object s1) (object s2))))
```

When a plan is defined, its equality constraints are normalized, i.e., (1) any equality constraints sharing a common operand specifier are merged, and (2) redundant operand specifiers within a constraint are removed. Whereas T-REX relies on K-Rep and MATS to normalize concept definitions and temporal constraints respectively, it must normalize coreference constraints itself. For instance, normalization would merge the following definition's first two coreference constraints into a single coreference constraint with three operand specifiers.

```
(defplan SOLO-MAKE-SPAGHETTI-MARINARA
  ((s1 c-make-spaghetti)
   (s2 c-boil)
   (s3 c-make-marinara))
  :allen-constraints ((s1 (before meets) s2))
  :coref-constraints ((equal (agent s1) (agent s2))
                      (equal (agent s2) (agent s3))
                      (equal (object s2) (object s1))))
```

In addition, all value restrictions designated by the operand specifiers are verified to be mutually compatible[4].

Where applicable, T-REX effectively replaces all role value restrictions specified by the operand specifiers of an equality constraint with their conjunction. Hence, T-REX may need to define new action concepts which are specializations of the action concepts referred to in a plan description. Internally, the plan will be defined in terms of these new concepts. For example, suppose the actions of the SOLO-BOIL-SPAGHETTI plan are defined thusly:

[4]Compatibility is intransitive. Here we rely on the reader's intuitive notion of compatibility. A technical definition is forthcoming in Section 4.3.2.

```
(defprimconcept c-make-spaghetti
  (and c-action
       (all agent c-human)
       (all object c-spaghetti)))

(defprimconcept c-boil
  (and c-action
       (all agent c-italian)
       (all object c-pasta)))
```

Based on the coreference constraints, T-REX would replace these concepts within the MAKE-SPAGHETTI plan by, respectively:

```
(defprimconcept c-make-spaghetti-prime
  (and c-action
       (all agent c-italian)
       (all object c-spaghetti)))

(defprimconcept c-boil-prime
  (and c-action
       (all agent c-italian)
       (all object c-spaghetti)))
```

4.1.5 End Plans

Plan library entries often describe a course of action that an agent can deliberately carry out to achieve a goal. Following [Kautz, 1991], we call this subset of our plan library entries *end* plans [5]. However, the plan library administrator may also wish to introduce descriptions which should not be recognized as plans *per se*, perhaps for purposes of indexing or inheritance, or to trigger some functionality upon recognition. We can compute the possibility and necessity of arbitrary patterns of events (other than end plans, but presumably still meaningful) by classifying them in the plan taxonomy and marking them distinctly. Defined plans are taken to be end plans by default, including all the plans defined in Appendix A.

[5] Whereas Kautz represents END as a plan in his plan abstraction hierarchy which other plans may explicitly specialize [Kautz, 1991], we feel it is more appropriately modeled as a boolean attribute associated with each plan.

4.2 Terminological Reasoning with Constraint Networks

This section and Section 4.3 present terminological reasoning with constraint networks and terminological plan recognition, respectively, in a fairly formal manner. We will confine our attention to networks composed of action concepts and Allen's temporal constraints. Then Section 4.4 informally (yet carefully) describes our recent extensions to T-REX for metric and coreference constraints. We are now in the process of formalizing this work.

4.2.1 Structural Subsumption

Set theoretically, one plan *subsumes* another just in case every possible instance of the second is also an instance of the first. In this proposal, we restrict our attention to inferences via plan bodies. Then structural plan subsumption can be characterized in terms of graph matching. Plan subsumption is based on subsumption between nodes and subsumption between arcs. We define *node subsumption* and *temporal constraint subsumption* as follows:

Definition 1 *Node* N1 *subsumes node* N2 *iff the concept associated with* N1 *subsumes the concept of* N2*.*

Definition 2 *Allen temporal constraint* C1 *subsumes temporal constraint* C2 *iff* C1*'s disjuncts are a superset of* C2*'s disjuncts.*

For example, *before or after* subsumes (1) *before*, and (2) *after*, as well as (3) *before or after*. For plan networks, *arc subsumption* follows immediately from temporal constraint subsumption. In other applications, we might use K-Rep concepts to represent the semantics of arcs. Structural subsumption between terminological constraint networks such as plan networks entails an appropriate mapping:

Definition 3 *A* subsumption mapping *from terminological constraint network* T1 *to terminological constraint network* T2 *maps every node* N1 *of* T1 *to a distinct node* N2 *of* T2 *such that* N1 *subsumes* N2*, and every arc between a pair of nodes in* T1 *subsumes the arc between the corresponding nodes in* T2*.*

In the case of plan networks, Definition 3 assumes that all nodes correspond to atomic actions, i.e., any macro actions have already been fully expanded and constraints on nodes with macro actions have been propagated to the constituents. It also assumes that constraint propagation on *T2* is complete. We require distinct images in *T2* for the nodes of *T1* because, by definition, distinct nodes within a terminological constraint network denote distinct entities, e.g., action. Then we can prove the following theorem, which formally justifies the subsumption algorithm presented in Section 4.2.1:

Theorem 1 *Terminological constraint network* T1 *subsumes terminological constraint network* T2 *iff there exists a subsumption mapping from* T1 *to* T2.

The proof appears in Appendix C.

Appendix A defines a plan library from which many of this proposal's examples are drawn. Figure 5 illustrates the key subsumption relations which demonstrate that the HEAT-NOODLES plan network subsumes the ASSEMBLE-SPAGHETTI-MARINARA plan network (after expansion of the latter's BOIL-SPAGHETTI macro action). In Figure 6, dashed arrows indicate the subsumption mapping from the subsumer to the subsumee. From now on, we will simply show mappings between corresponding nodes, with the understanding that the intervening arcs are mapped accordingly.

Notice that the two plans differ in the number and specificity of their actions, as well as the specificity of the relevant constraint. This is analogous to structural subsumption in TKR, where a concept may specialize its parent(s) by further constraining their roles (constraints) and/or adding additional roles (constraints).

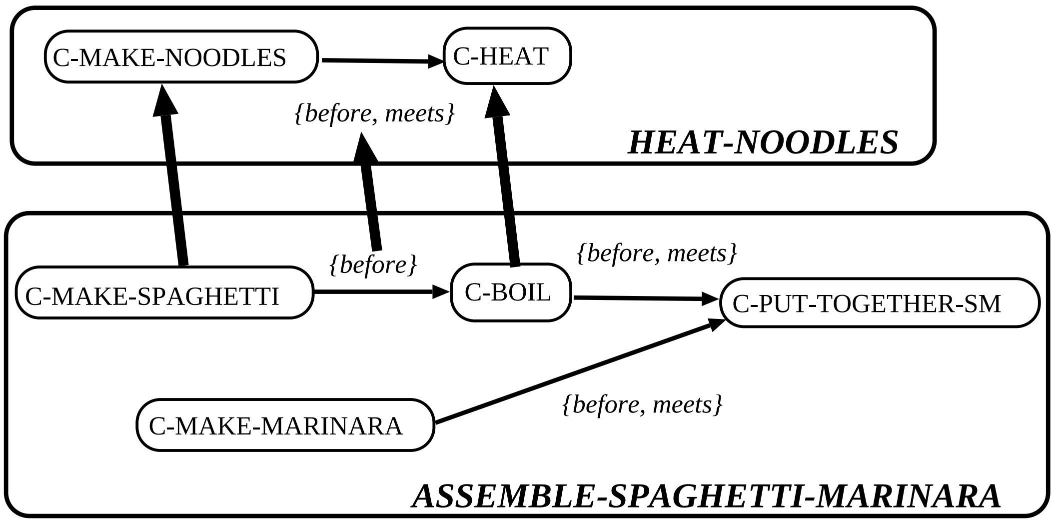

Figure 5: Node and Arc Subsumption between Plans

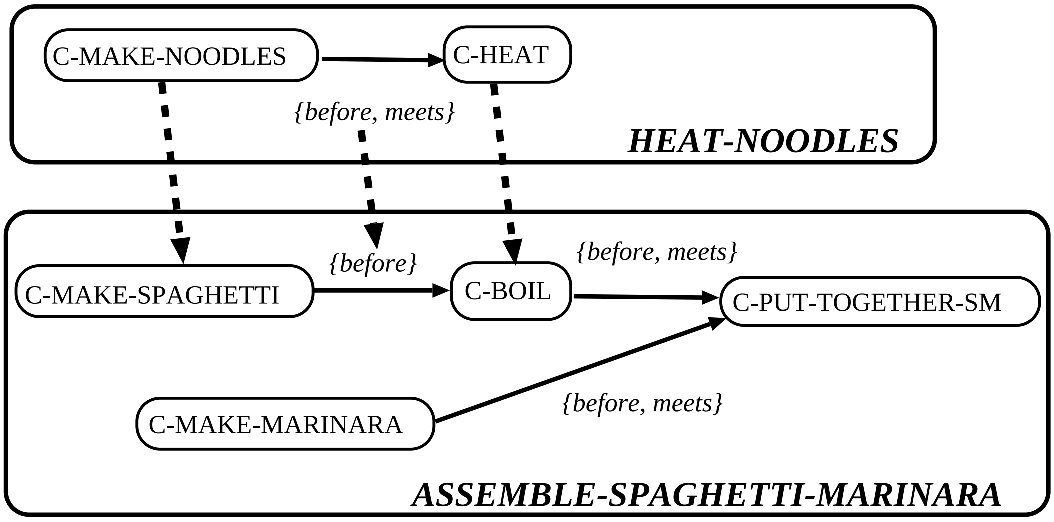

Figure 6: Plan Subsumption Mapping

4.2.2 Computational Complexity of Subsumption

Computing subsumption between the concepts associated with nodes amounts to querying K-Rep. In our case, the concept taxonomy will be constructed in advance, so the results can be retrieved in constant time (we can precompute the transitive closure of the subsumption relations for each concept). Temporal constraints can be represented as bitstrings of length 13, so subsumption between them can also be determined in constant time.

The crux of the plan subsumption problem is to establish a suitable mapping from one plan network to another. This problem is clearly in NP, and there is a polynomial time transformation from directed subgraph isomorphism, which is NP-complete, to subsumption mapping between terminological constraint networks. Thus we have:

Theorem 2 *Subsumption mapping between terminological constraint networks is NP-complete.*

The full proof appears in Appendix C.

4.2.3 Practical Subsumption Performance

We share the view of Doyle and Patil, who argue against the *restricted language thesis* that KR systems should limit their expressiveness to achieve polynomial worst-case response times [Doyle and Patil, 1991]. It is important to observe that in our instantiation of the subgraph isomorphism problem, both the nodes and the arcs are "labeled," so powerful heuristics can be brought to bear. As Sowa noted, albeit in a different context: "The labels help to guide the pattern match when it is going to be successful, and a mismatch of labels can cause it to fail quickly when there is no chance of success. Therefore, the labels speed up the pattern matching in many cases."

Terminological network subsumption exemplifies the well-known constraint satisfaction problem (CSP). In CSP, we are given a set of variables (corresponding to nodes in the putative subsuming network), and our task is to instantiate each variable with values from a specified domain (nodes in the putative subsumee), subject to certain constraints (for plan networks, the action types of the nodes plus the set of temporal relationships described by the arcs).

We now summarize an algorithm to decide whether some terminological constraint network *T1* subsumes another one *T2*:

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 using a procedure such as Allen's.

2. **Closure:** Close both networks via constraint propagation.

3. **Preliminary Analysis:** For each node *N1* in *T1*, determine which nodes in *T2* are subsumed by *N1* according to the concept taxonomy. Call those nodes the *potential images* of *N1*. If the number of potential images for any node in *T1* is zero, return *false*. Otherwise, sort the nodes of *T1* in increasing order of potential image count to help guide the subsequent graph matching process.

4. **Matching by Backtracking:** Using the preliminary analysis for heuristic guidance, extend the mapping from *T1* to *T2* one step at a time. Each extension consists of selecting an additional node *N1* from *T1* and associating with it an additional node *N2* from among its potential images, such that the constraints on all nodes selected from *T2* continue to respect the constraints on the corresponding nodes from *T1*. When each node from *T1* has been mapped to a distinct node from *T2*, return the mapping. At any point, if there is a node from *T1* which cannot be so mapped, backtrack. If the backtracking process is exhausted without finding a suitable mapping, return *false*.

T-REX currently implements this algorithm, which is sound, and complete to the extent that constraint propagation on *T2* is complete. Similar to an existing algorithm for production rule subsumption [Yen *et al.*, 1991], it employs well-known CSP techniques. CSP has been widely studied, and improvements are possible. The preliminary analysis that restricts a node's image to be one of its potential subsumees is an example of the *node consistency* technique. Many other powerful CSP methods such as those based on *arc consistency* are available [Mackworth, 1977]. Choosing the optimal mix is domain-dependent and largely still a black art [Kumar, 1992].

4.2.4 Plan Instantiation

We are also concerned with discovering when a plan instance *instantiates* a particular plan, sometimes referred to in the literature as *plan realization.* The computation for actions and temporal relations is essentially the same as for subsumption, so we will not elaborate here (but see Section 4.4).

4.2.5 Classification

Structural plan subsumption allows T-REX to automatically classify plan taxonomies strictly according to the semantics of the plans. Our initial implementation of classification is entirely unremarkable, but see Section 6.2.2 for future work. Figure 7 shows a plan taxonomy constructed by T-REX using the set of plan definitions presented in Appendix A. The root of the plan taxonomy is the trivial plan, PLAN, which has no actions.

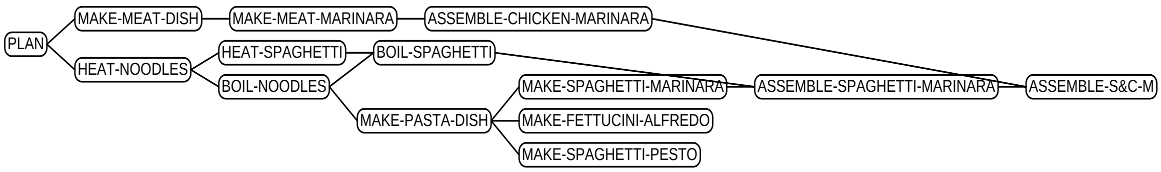

Figure 7: A Definitional Plan Taxonomy

Taxonomies formed by classifying plan networks, like terminological constraint networks in general, enjoy all the benefits of classification cited in Section 2. Furthermore, as plan libraries grow in size and scope, their organization and maintenance becomes increasingly critical. Search procedures can utilize the definitional placement of plans within a taxonomy for fast and accurate results. Also, since most present day plan libraries are organized by hand, the clerical demands placed on the plan librarian may become burdensome. Our experience with knowledge engineering shows that when confronted with large quantities of information, the enforced semantics of the terminological approach offers significant advantages [Mays *et al.*, 1991b].

4.3 Terminological Plan Recognition

We now exploit the plan library's terminological nature to guide plan recognition. By searching for suitable mappings between the observations and the plans, we can assign the plans modalities, e.g., *necessary*, *possible* and *impossible*, that indicate their status with respect to the observations. This process, which partitions the plan library by modality, is unique to our work. We shall examine plan recognition under varying assumptions about the accuracy and monotonicity of the observations.

An observation represents a determination that actions(s) have occurred and/or that temporal constraint(s) hold between actions. The system records its observations in a network similar to plan networks which we call the *observation network*. Action instances are associated with the nodes of an observation network. In general, the observation network may be an inexact or incomplete model of the events. A sample observation network is shown in Figure 8.

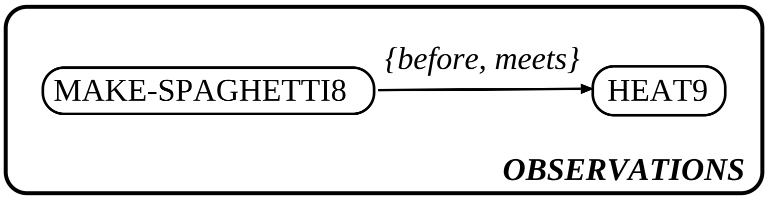

Figure 8: A Sample Observation Network

As events unfold and observations are made, the observation network is updated, yielding successive versions. An update may entail *extension* and/or *refinement*. Extensions add new actions and/or temporal constraints, while refinements further constrain (specialize) existing actions and/or temporal constraints. More generally, observations can be retracted or generalized.

We make a *complete library assumption* that each observed action is directed towards fulfilling a plan or plans in the plan library. This is a *closed-world assumption* [Reiter, 1978]. Consequently, at least one plan is possible at all times and at least one plan will eventually prove necessary. As mentioned in Section 4.1.1, we assume that the taxonomy includes every type of action which appears in a plan, or is observed during plan recognition.

Until Section 4.3.5, we further make the *single plan assumption* that the observations will ultimately be fully accounted for by a single plan (they may also be partially accounted for by more general plans). Both assumptions are common in the field of plan recognition. The latter is the most restricted version of Kautz's *minimum cardinality assumption*, which always prefers to account for observations with the smallest number of plans [Kautz, 1991]. It is often reasonable to suppose that observed actions are related.

Terminological plan recognition is based on *potential* subsumption relationships between the observations and plans in the plan library. We will say that a plan is *possible* with respect to the observations if it subsumes or might eventually subsume the observations, i.e., perhaps pending suitable further observations. When a plan cannot subsume the observations under the prevailing assumptions, the plan is *impossible*. A possible plan which actually subsumes the observations is also *necessary*. Stronger plan recognition results may follow from cardinality assumptions, e.g., due to the complete library assumption, we know that when only one plan remains possible, it is effectively necessary. For convenience, we refer to plans which are possible but not necessary as *optional*. Before any observations are made, all plans are optional except for PLAN, which is trivially necessary. Afterwards, both the plan definitions and the prevailing assumptions interact with the observations to determine the modality of each plan.

The recognition process relies on the terminological nature of the plan taxonomy to partition the taxonomy into three connected regions. Figure 9 illustrates the division of the plan taxonomy into necessary (N), optional (O), and impossible (I) regions (the rendering of the border between optional and impossible plans emphasizes the point that an optional plan may subsume an impossible plan under a minimum cardinality assumption). Since the taxonomy is definitional, we need not compare every plan with the observations to accomplish the partitioning, e.g., except for PLAN, a plan is not possible unless one of its parents is possible.

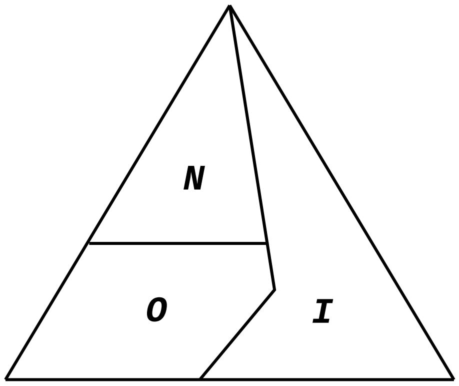

Figure 9: Modalities in a Definitional Plan Taxonomy

4.3.1 Perfect Observations

Let us begin with the stringent assumption that the observation network is perfect. In our framework, this implies that the types of observed actions are leaves in the action taxonomy and that observed temporal relationships are nondisjunctive. 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 or retracted.

The *perfect observation assumption* is sometimes quite justified. For instance, we can flawlessly capture a user's interactions with software systems such as operating systems or graphical user interfaces. Indeed, we view user interfaces as a likely application for our ideas.

Suppose we have the following plan network with two actions (also shown in Figures 5 and 6 and in Appendix A):

- HEAT-NOODLES: C-MAKE-NOODLES {*before, meets*} C-HEAT

It subsumes the following (unrelated) observation networks, among others:

- OBS-1: MAKE-SPAGHETTI1 {*before*} BOIL2
- OBS-2: MAKE-ZITI3 {*meets*} BAKE4

Thus, both of the preceding observation networks license the conclusion that the HEAT-NOODLES plan is necessary.

Intuitively, a plan is possible with respect to the observations if it subsumes or might eventually subsume them, i.e., it is necessary or optional. We introduce *inverse subsumption* to characterize optional plans which directly reflect the present observations:

Definition 4 *An* inverse subsumption mapping *from terminological constraint network* T2 *to terminological constraint network* T1 *maps every node* N2 *of* T2 *to a distinct node* N1 *of* T1 *such that* N2 *is subsumed by* N1*, and every arc between a pair of nodes in* T2 *is subsumed by the arc between the corresponding nodes in* T1*.*

An optional plan network P which enjoys an inverse subsumption mapping from the observations will actually subsume them if we subsequently observe nodes and arcs subsumed by the as yet unobserved portion of P. Since an inverse subsumption mapping constitutes direct evidence that a plan may be in progress, we will call such plans *directly optional*. The next observation network, which consists of a single action, is potentially subsumed by HEAT-NOODLES in this way:

- OBS-3: MAKE-SPAGHETTI5

For example, OBS-3 would become subsumed by HEAT-NOODLES if a C-BOIL action were observed to occur after the MAKE-SPAGHETTI5. Hence, the status of HEAT-NOODLES would change from directly optional to necessary. On the other hand, an observation of MAKE-CHICKEN would render HEAT-NOODLES not directly optional, given that a MAKE-CHICKEN action is not subsumed by any action in HEAT-NOODLES.

Now, we can formally define *potential subsumption* of an observation network by a plan in isolation:

Definition 5 *Plan network* P potentially subsumes *observation network* O *under perfect observation iff (1)* P *subsumes* O*, or (2) there exists an inverse subsumption mapping from* O *to* P*.*

When we consider plan recognition with respect to a plan library, there is another class of optional plans, not covered by Definition 4, which may eventually subsume the observations. To see this, consider again OBS-3, now in the context of the portion of Figure 7 detailed in Figure 10 (which shows the pertinent subsumption mapping). The MAKE-SPAGHETTI5 has no counterpart in the ASSEMBLE-CHICKEN-MARINARA plan, nor have we observed any other action in that plan. A notion of possibility based on inverse subsumption alone would lead to the conclusion that ASSEMBLE-CHICKEN-MARINARA is "impossible." However, it admits the possibility that the agent is following the ASSEMBLE-S&C-M plan. This seems somewhat paradoxical since ASSEMBLE-CHICKEN-MARINARA subsumes ASSEMBLE-S&C-M. Based on our evidence that the latter is optional, we want to sanction the indirect conclusion that ASSEMBLE-CHICKEN-MARINARA is also optional. T-REX therefore recognizes a supplemental class of optional plans. Any plan which does not enjoy an inverse subsumption mapping from the observations, but does subsume an optional plan, is itself *indirectly optional.*

There is one remaining case where a plan is indirectly optional. Consider a plan library with two plans; neither subsumes the other:

- PLAN-X: C-MAKE-SPAGHETTI {*before*} C-BOIL

- PLAN-Y: C-MAKE-MARINARA {*before*} C-MAKE-NOODLES {*before, meets*} C-BOIL

Now consider this perfect observation:

- OBS-4: MAKE-MARINARA34

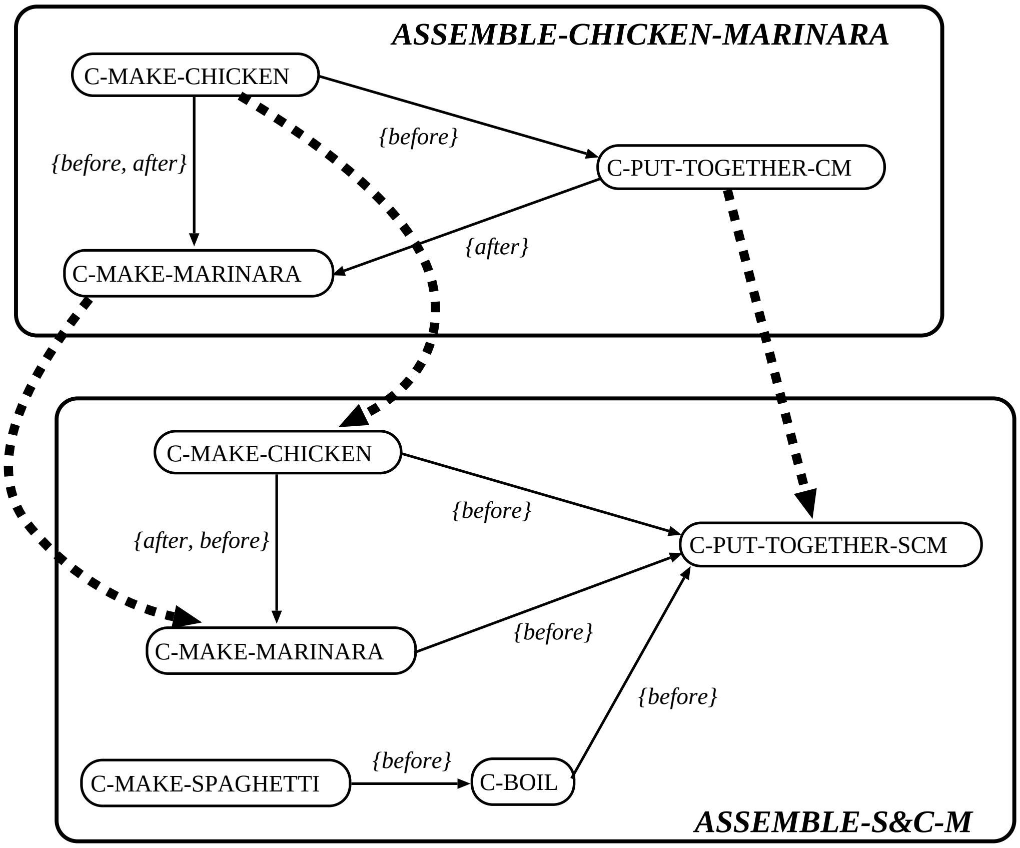

Figure 10: Indirect Optionality

There is an inverse subsumption mapping from OBS-4 to PLAN-Y, so PLAN-Y is directly possible. Although it cannot be mapped to PLAN-X, OBS-4 *can* be extended to instantiate PLAN-Y in such a way that it also instantiates PLAN-X, e.g.:

- OBS-5: MAKE-MARINARA34 {*before*} MAKE-SPAGHETTI35 {*before*} BOIL36

On the other hand, OBS-6 can be extended to instantiate PLAN-Y but not to instantiate PLAN-X:

- OBS-6: MAKE-MARINARA34 {*before*} MAKE-FETTUCINI37

The relationship between PLAN-X and PLAN-Y illustrates a general situation where plan *P1* does not subsume plan *P2*, yet certain instantiations of *P2* will also be subsumed by *P1*. In these situations we *sometimes* want to infer that *P1* is indirectly optional via *P2*. One solution is for T-REX to ensure that there exist plan(s), e.g., *P3*, subsumed by both *P1* and *P2* such that *P1* is indirectly optional via *P2* just in case *P3* is directly optional. T-REX can always create such plan(s) as required. Therefore, we will assume throughout that the library has been so augmented, i.e., the *augmented complete library assumption*. After introducing the required machinery in Section 4.3.2, we will specify how this is accomplished in Section 4.3.3.

Now, we can formally define *potential subsumption* with respect to a plan library:

Definition 6 *Plan network* P potentially subsumes *observation network* O *under perfect observation iff (1) there exists an inverse subsumption mapping from* O *to* P*, or (2) there exists a plan* P' *such that* P *subsumes* P' *and* P' *potentially subsumes* O*.*

Under the complete library assumption, whenever *P* actually subsumes *O*, one of the two clauses of this definition must be true. Potential subsumption expands the notion of actual subsumption, i.e., subsumption entails potential subsumption but the converse is not true. That potential subsumption precisely captures our idea of possibility is stated in the following theorem (proved in Appendix C):

Theorem 3 *Under the complete library, single plan and perfect observation assumptions, a plan is possible iff it potentially subsumes the observations.*

Given OBS-3, we can now see that our recognition methodology will partition the plan taxonomy in Figure 7 into the following modalities:

Directly Optional:	HEAT-NOODLES
	HEAT-SPAGHETTI
	BOIL-NOODLES
	MAKE-PASTA-DISH
	MAKE-SPAGHETTI-PESTO
	MAKE-SPAGHETTI-MARINARA
	BOIL-SPAGHETTI
	ASSEMBLE-SPAGHETTI-MARINARA
	ASSEMBLE-S&C-M
Indirectly Optional:	MAKE-MEAT-DISH
	MAKE-MEAT-MARINARA
	ASSEMBLE-CHICKEN-MARINARA
Impossible:	MAKE-FETTUCINI-ALFREDO

Note that the recognition of MAKE-FETTUCINI-ALFREDO as impossible depends crucially on the complete library and single plan assumptions. Of course, given a wide range of cooking plans, OBS-3 alone would render many of them impossible.

If observation network OBS-3 is extended to include an instance of C-BOIL occurring after MAKE-SPAGHETTI5, the following plans change from directly optional to necessary: HEAT-NOODLES, HEAT-SPAGHETTI, BOIL-NOODLES and BOIL-SPAGHETTI. By contrast, if OBS-3 is instead extended to include an instance of MAKE-CHICKEN (temporally unconstrained), then HEAT-NOODLES, HEAT-SPAGHETTI, BOIL-NOODLES, MAKE-PASTA-DISH, MAKE-SPAGHETTI-MARINARA, BOIL-SPAGHETTI, and ASSEMBLE-SPAGHETTI-MARINARA change from directly optional to indirectly optional, while MAKE-SPAGHETTI-PESTO changes from directly optional to impossible.

Our recognition methodology is not limited to plans. Definitions 5 and 6 simply specify that terminological constraint network *T2* satisfies a subnetwork of *T1*. Likewise, Definitions 5 and 6 apply generally to any pair of terminological constraint networks, P and O. It is interesting to note that our methodology could also partition a regular K-Rep concept taxonomy into modalities *vis à vis* a K-Rep instance as its definition is extended.

4.3.2 Monotonic Observations

Now we permit imprecise observations, including action instances of arbitrarily abstract type and/or disjunctive temporal constraints, along with refinement of prior observations. The type of an action instance in the observation network may be refined to a more specific type [6]. Similarly, an observed temporal constraint may be refined to a subset of its disjuncts.

This framework poses more of a challenge. An action instance in the observation network which is not subsumed by a certain action in a plan network may later be refined to the point that it becomes subsumed by that action, and similarly for temporal constraints. For motivation, consider the following pair of plan networks, neither of which subsumes the other (as defined in Appendix A and illustrated in Figure 7):

- BOIL-NOODLES: C-MAKE-NOODLES {*before, meets*} C-BOIL
- HEAT-SPAGHETTI: C-MAKE-SPAGHETTI {*before*} C-HEAT

Also consider the following pair of observation networks:

- OBS-7: MAKE-NOODLES6 {*before, meets*} BOIL7
- OBS-8: MAKE-SPAGHETTI8 {*before*} HEAT9

Notice that OBS-7, which is subsumed by BOIL-NOODLES, would become subsumed by HEAT-SPAGHETTI if MAKE-NOODLES6 was refined to be of type C-MAKE-SPAGHETTI

[6] The *type* of an instance is the conjunction of the concepts which subsume it.

and the temporal constraint was refined to $\{$*before*$\}$. Conversely, OBS-8, which is subsumed by HEAT-SPAGHETTI, would become subsumed by BOIL-NOODLES if HEAT9 was refined to be of type C-BOIL.

The possibility of refinement forces us to expand the conditions under which a plan is deemed possible. A plan is possible if the observations are consistent with it or may become so. Potential subsumption of the observation network by a plan network under monotonic observation depends on *compatibility* of actions and temporal constraints. We formalize this notion with respect to structural subsumption and our completeness assumptions by the following series of definitions.

Definition 7 *A pair of concepts (constraints) are* compatible *iff there exists a concept (constraint) which they both subsume.*

Thus C-HEAT is compatible with C-BOIL and *vice versa.* Recall that subsumption is reflexive.

Definition 8 *An instance* I *and a generic concept* G *are* compatible *iff the type of* I *is compatible with* G.

Thus HEAT27 is compatible with C-BOIL and conversely.

Definition 9 *Temporal constraints are* compatible *iff the intersection of their disjuncts is non-empty.*

The constraint $\{$*before, during*$\}$ is bidirectionally compatible with $\{$*during, after*$\}$.

Of course, when an observed action or constraint is incompatible with a certain action or constraint in the plan library, their incompatibility is impervious to future refinement of the observation (recall our assumption that the action taxonomy is complete).

Compatibility for a pair of terminological constraint networks, such as a plan network and an observation network, can be decided as follows:

Definition 10 *A pair of nodes (arcs) are* compatible *iff the associated concepts (constraints) are compatible.*

Definition 11 *There is a* compatibility mapping *from terminological constraint network* T1 *to terminological constraint network* T2 *iff every node of* T1 *is compatible with a distinct node of* T2*, such that every arc between a pair of nodes in* T1 *is compatible with the arc between the corresponding nodes in* T2*.*

Finally, we can give a more general definition for potential subsumption of an observation network by a plan in isolation:

Definition 12 *Plan network* P potentially subsumes *observation network* O *under monotonic observation iff (1) there exists a compatibility mapping from* P *to* O*, or (2) there exists a compatibility mapping from* O *to* P*.*

As before, potential subsumption with respect to a complete plan library entails a level of indirection:

Definition 13 *Plan network* P potentially subsumes *observation network* O *under monotonic observation iff (1) there exists a compatibility mapping from* O *to* P*, or (2) there exists a plan* P' *such that* P *subsumes* P' *and* P' *potentially subsumes* O*.*

Intuitively, a plan network is possible if the observation network can be extended and/or refined so that it is subsumed by the plan network. Definition 13 is formally justified by the following (proved in Appendix C):

Theorem 4 *Under the complete library, single plan and monotonic observation assumptions, a plan is possible iff it potentially subsumes the observations.*

Returning to our motivating example, Definition 13 shows that BOIL-NOODLES and HEAT-SPAGHETTI potentially subsume observation networks OBS-7 and OBS-8. In particular, the full partitioning of the plan taxonomy in Figure 7 given OBS-7 is:

Necessary: HEAT-NOODLES
BOIL-NOODLES

Directly Optional: HEAT-SPAGHETTI
MAKE-PASTA-DISH
MAKE-FETTUCINI-ALFREDO
MAKE-SPAGHETTI-PESTO
MAKE-SPAGHETTI-MARINARA
BOIL-SPAGHETTI
ASSEMBLE-SPAGHETTI-MARINARA
ASSEMBLE-S&C-M

Indirectly Optional: MAKE-MEAT-DISH
MAKE-MEAT-MARINARA
ASSEMBLE-CHICKEN-MARINARA

In contrast, the partitioning resulting from OBS-8 is:

Necessary:	HEAT-NOODLES HEAT-SPAGHETTI
Directly Optional:	BOIL-NOODLES MAKE-PASTA-DISH MAKE-SPAGHETTI-PESTO MAKE-SPAGHETTI-MARINARA BOIL-SPAGHETTI ASSEMBLE-SPAGHETTI-MARINARA ASSEMBLE-S&C-M
Indirectly Optional:	MAKE-MEAT-DISH MAKE-MEAT-MARINARA ASSEMBLE-CHICKEN-MARINARA
Impossible:	MAKE-FETTUCINI-ALFREDO

Under the single plan assumption and monotonic observation, the set of plans that are optional (directly or indirectly) decreases monotonically as observations occur. In that case, the effect of each new observation is to change the status of zero or more plans to necessary or impossible. For example, if MAKE-NOODLES6 in OBS-7 is refined to be of type C-MAKE-SPAGHETTI and the temporal constraint refined to {*before*}, the plans HEAT-SPAGHETTI and BOIL-SPAGHETTI change from directly optional to necessary, while MAKE-FETTUCINI-ALFREDO changes from directly optional to impossible. If HEAT9 in OBS-8 is refined to be of type C-BOIL, the plans BOIL-NOODLES and BOIL-SPAGHETTI change from directly optional to necessary.

As before, our recognition methodology for monotonic observations applies to any type of terminological constraint network, not just plans. Moreover, our methodology could also partition a regular K-Rep concept taxonomy into modalities *vis à vis* a K-Rep instance as its definition is monotonically updated.

4.3.3 Augmenting the Plan Library

As noted in Section 4.3.1, we must account for cases where plan *P1* is indirectly optional via plan *P2*, but *P1* does not subsume *P2*. Motivated by our desire to curtail inferencing during plan recognition, the solution we have implemented *augments* the plan library by creating additional plans for the internal use of T-REX, so that a plan is indirectly optional just in case it subsumes a directly optional plan.

Observe that we can compute a compatibility mapping from one plan to another, just as we do between a plan and the observations. Such a mapping establishes *structural compatibility*. In general, we augment the library as needed to ensure that there exists a plan *P3* for every compatibility mapping from some plan *P1* to another plan *P2*, where *P1* does not subsume *P2*, yet an instantiation of *P2* may also be subsumed by *P1*. *P3* is created by specializing *P2* according to a compatibility mapping from *P1* so every constituent (action, temporal or equality constraint) *C2* of *P2* mapped from constituent *C1* of *P1* is replaced by the conjunction of *C1* and *C2*. In the example from Section 4.3.1, C-MAKE-SPAGHETTI and C-BOIL of PLAN-X are mapped to C-MAKE-NOODLES and C-BOIL of PLAN-Y, respectively, along with the corresponding intervening temporal constraints, to derive:

- PLAN-Z: C-MAKE-MARINARA {*before*} C-MAKE-SPAGHETTI {*before*} C-BOIL

In PLAN-Z, the second action and second temporal constraint of PLAN-Y have been specialized. Note that *P2* always has at least as many nodes as *P1*. In case *P1* and *P2* have the same number of nodes, all compatibility mappings between them are symmetric, so we need not consider mappings from *P2* to *P1* separately. Throughout this paper, we assume that the plan library has been augmented in the way we have described. It need only be done once, after the plan library is defined and before plan recognition commences. We are studying how to minimize the number of plans that must be added overall.

4.3.4 Unrestricted Observations

T-REX actually provides for arbitrary modification and retraction of observations. 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 subsumption under monotonic observation still applies. When allowing nonmonotonic observations, however, 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. Nonmonotonic observation could have unfortunate performance consequences. We must effectively be able to undo any constraint propagation in the observation network, since the justification may cease to exist. Retraction in the observation network is currently done by recomputation. Presumably it could also be supported via truth maintenance, but the cost of tracking dependencies may not be worthwhile.

4.3.5 Simultaneous Plans

When the single plan assumption is violated, T-REX accounts for the eventuality 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, 1991], observed actions can be shared among plans.

A set of plans accounts for all observed actions iff there is a compatibility mapping from the observation network to their multiple plan network. T-REX also needs a way to explore the set of possible plan combinations. Cardinality assumptions seem essential to constrain the combinations for reasonable performance. Also, in their absence, 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. 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, 1991]. As a first cut at improvement, T-REX only considers those multiple plan networks that have a compatible action for every observed action.

As an example, consider observation network OBS-9:

- OBS-9: MAKE-FETTUCINI10 {*before*} MAKE-NOODLES11 {*before*} MAKE-ALFREDO12 {*before*} MAKE-ALFREDO13

Since no single plan can account for the observations, T-REX infers that three possible sets of two (interleaved) plans can account for OBS-9:

1. {MAKE-FETTUCINI-ALFREDO, MAKE-FETTUCINI-ALFREDO}
2. {MAKE-FETTUCINI-ALFREDO, MAKE-PASTA-DISH}
3. {MAKE-PASTA-DISH, MAKE-PASTA-DISH}

T-REX searches for combinations of *end plans* only; if T-REX were told that the MAKE-PASTA-DISH plan is not an end in itself, only the first of these combinations would be inferred.

4.4 Inferences with Metric and Coreference Constraints

This section describes recent T-REX extensions to handle metric temporal constraints and coreference constraints. Our presentation here is somewhat less formal than in preceding portions of Section 4. As noted above, we are now in the process of formalizing this work.

4.4.1 Plan Subsumption and Instantiation

We have recently extended our algorithm on page 19 to compute plan subsumption in light of metric constraints. First, the potential images of a node must now respect constraints on its duration as well as its associated action. Metric constraint subsumption follows from containment on the real number line, so STEP2 of the first plan below (repeated from Section 4.1.3) subsumes S2 of the second plan:

```
(defplan DEMO-METRIC-CONSTRAINTS
  ((step1 c-make-noodles)
   (step2 c-heat))
  :metric-constraints
   ((5 <= left step2 - right step1 <= 5)
    (6 <= right step2 - left step2 <= 9)))

(defplan DEMO-METRIC-CONSTRAINTS-SUBSUMEE
  ((s1 c-make-spaghetti)
   (s2 c-boil))
  :metric-constraints
   ((5 <= left s2 - right s1 <= 5)
    (6 < right s2 - left s2 <= 8)))
```

Second, when a subsumption mapping is extended by associating node *N2* from *T2* with node *N1* from *T1*, the metric constraints between the starting and ending points of *N2* and those of all previously selected nodes from *T2* must continue to respect the corresponding metric constraints from *T1*. That is, temporal constraint subsumption between a pair of nodes entails subsumption between a pair of Allen constraints and four pairs of metric constraints. Suppose that given the two preceding plans, our subsumption algorithm has already mapped STEP1 of DEMO-METRIC-CONSTRAINTS to S1 of DEMO-METRIC-CONSTRAINTS-SUBSUMEE. Subsequently mapping STEP2 to S2 entails verifying these four metric constraint subsumption relations:

- LEFT STEP2 - LEFT STEP1 *subsumes* LEFT S2 - LEFT S1
- LEFT STEP2 - RIGHT STEP1 *subsumes* LEFT S2 - RIGHT S1
- RIGHT STEP2 - LEFT STEP1 *subsumes* RIGHT S2 - LEFT S1
- RIGHT STEP2 - RIGHT STEP1 *subsumes* RIGHT S2 - RIGHT S1

Observation assumptions aside, metric constraints in plan instances are just like those in plans, so determining plan instantiation with respect to metric constraints is identical to plan subsumption with respect to them.

We have also recently extended our algorithm to compute plan subsumption in light of coreference constraints. At present, T-REX computes a subsumption mapping from plan *T1* to plan *T2* with respect to their actions and temporal constraints, then checks to see if this mapping respects their equality constraints. If not, it seeks another subsumption mapping. Plan *T1* subsumes plan *T2* with respect to their equality constraints (after normalization) just in case every equality constraint *E1* in *T1* subsumes some equality constraint *E2* in *T2*. That is the case when for each operand specifier in *E1* there is a corresponding operand specifier in *E2* such that their role-names are the same (there is no role hierarchy) and their labels are bound together under the current mapping. The latter property ensures that every operand specifier in *E1* is mapped to a distinct operand specifier in *E2* (as always, steps in a plan are assumed to be disjoint). Also, note that the ordering of operand specifiers within an equality constraint is immaterial since equality is commutative. As an example, it can be seen that the SOLO-BOIL-SPAGHETTI plan on page 14 subsumes the SOLO-MAKE-SPAGHETTI-MARINARA plan which follows it.

T-REX must also be able to determine whether some plan instance satisfies every coreference constraint specified by a given plan description. For each operand specifier in a given coreference constraint, T-REX collects the corresponding role filler from an action instance in the plan instance (with respect to the putative subsumption mapping), and applies the operator — currently only equality — to those operands. For example, the following plan instance would instantiate the SOLO-BOIL-SPAGHETTI plan on page 14 just in case its two action instances had identical fillers for their AGENT roles and for their OBJECT roles.

- MAKE-SPAGHETTI98 {*meets*} BOIL99

4.4.2 Plan Recognition

We have recently extended our plan recognition algorithm to account for metric constraints. We will consider the case of monotonic observation, since it includes perfect observation as a special case. First, nodes are compatible if both their actions and their durations are compatible. Durations, like all metric constraints, are compatible if their intersection is non-empty, e.g., the durations of INTERVAL1 and INTERVAL2 defined as follows:

- 4 <= RIGHT INTERVAL1 - LEFT INTERVAL1 <= 8
- 7 <= RIGHT INTERVAL2 - LEFT INTERVAL2 <= 9

Second, temporal constraints between nodes are compatible if the Allen constraints between the associated intervals are compatible, and the metric constraints among the corresponding starting and ending points are compatible. That is, temporal constraint compatibility between a pair of nodes entails compatibility between a pair of Allen constraints and four pairs of metric constraints.

When equality constraints are applied to plan instances under monotonic observation, the K-Rep instances identified by the operand specifiers must be of *compatible* types. When only part of a plan has been observed, T-REX may find a compatibility mapping from the plan instance to the plan description. Then, some operands of an equality constraint may not be present in the instance (they might still be observed in the future). We need only check the compatibility of those operands which are present in the observation network. For example, the following observations would be compatible with the SOLO-MAKE-SPAGHETTI-MARINARA plan on page 14 when the agent of MAKE-SPAGHETTI998 is a particular human, say JOE997, and the agent of BOIL999 is only known to be an individual of type C-HUMAN. This is because, under monotonic observation, we may later discover that the agent of BOIL999 is in fact JOE997.

- MAKE-SPAGHETTI998 {BEFORE} BOIL999

5 Other Related Work

In the previous section we saw how our research to date builds upon the work in terminological knowledge representation and temporal reasoning described in Section 3. We now highlight the recent research by others most strongly related to our own, first in extending terminological reasoning, then in plan recognition.

5.1 Terminological Reasoning

Terminological reasoning with compositions of concepts has been investigated by others in three specialized domains: plans, temporal concepts, and production rules. We now summarize each of them.

5.1.1 Plans

Previous work on plan subsumption allowed plans that were either atemporal and used for plan synthesis [Wellman, 1990] or restricted to the relationship of temporal sequence and used for information retrieval [Devanbu and Litman, 1991]. There is also contemporaneous work on state-based reasoning with plans limited to simple sequences [Heinsohn *et al.*, 1992]. After considering each of these systems in turn, we summarize their comparability in Table 1. Proposed but unimplemented features of T-REX are parenthesized.

SUDO-PLANNER Wellman studied the formulation of tradeoffs in the context of medical therapy [Wellman, 1990]. He proposed an architecture for a constraint-posting 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 dominated plan class from the search space. Wellman's plans are composed of actions represented in a terminological hierarchy, but his plans are entirely atemporal. We outline some ideas for integrating Wellman's work with temporal planning in Appendix B.

CLASP T-REX's plan subsumption is similar in spirit to CLASP [Devanbu and Litman, 1991], which described plans as action sequences by means of regular expressions. However, by using Allen's temporal logic, T-REX supports simultaneous actions. T-REX also captures finer sequential relations than CLASP, which, for example, makes no distinction between *before* and *meets*. In [Devanbu and Litman, 1991], a plan instance with n steps can only be subsumed by plans with exactly n steps. Our system has no such restriction. CLASP has no coreference constraints between actions. 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. On the other hand, CLASP models preconditions and effects of actions and plans, and it fully supports disjunction and looping. Recently, PROTODL [Borgida, 1992] introduced a framework for extending terminological systems with customized language constructs. This methodology was demonstrated by reconstructing CLASP in PROTODL.

RAT The RAT system [Heinsohn *et al.*, 1992] is used in the WIP project at the German Research Center for Artificial Intelligence (DFKI) 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 actions. However, RAT focuses on the representation of complex state descriptions which 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 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 uses 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 which constitute formal parameters, as well as (2) preconditions and (3) postconditions, both of which are conjunctions of attribute restrictions, agreements and disagreements (role value maps with equality and inequality operators). Plans in RAT are defined by a set of parameters, an action sequence, and equality constraints among the plan's parameters and constituent actions.

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. Prospects for incorporating state information in T-REX are addressed in Section 6.3.3.

	SUDO-PLANNER	**RAT**	**CLASP**	**T-REX**
Application	plan synthesis	multimedia explanation	information retrieval	plan recognition
Temporal Language	none	simple sequences	regular expressions	constraint networks
Concurrent Actions	n/a	no	no	yes
Disjunction	no	no	yes	(restricted to single action)
Repetition	no	no	loop	(arbitrary, single action)
Subplans	no	no	yes	yes
Coreference Constraints	no	equality, inequality	no	equality (inequality)
Plan Instances	no	no	number of actions must agree w/ plan	unrestricted
States	no	yes	yes	no

Table 1: Comparison of Terminological Plan Systems

5.1.2 Temporal Concepts

Schmiedel [Schmiedel, 1990] has described an ambitious attempt to extend terminological logic with temporal semantics by integrating both Allen's temporal logic and Shoham's [Shoham, 1987]. Unlike T-REX, his representation supports concepts such as "former car owner". His temporal concept definitions include a set of temporal variables, along with temporal constraints among the variables. Although he offers no algorithm, Schmiedel does suggest a few "preliminary hints" (his words), including a definition of subsumption which corresponds to ours. His work did not consider temporal constraint networks as first class entities to be reasoned with in their own right, nor did he address either recognition or the notion of potential subsumption.

5.1.3 Production Rules

The CLASP system of [Yen *et al.*, 1991][7] 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 a knowledge engineering perspective, the rule taxonomy provides a principled basis for selecting rules to fire under the commonly used specificity criterion. This compares favorably with ad-hoc specificity measures used in production systems such as OPS5 [Brownston *et al.*, 1985]. We observe that the subsumption task we face is rather like the one described in [Yen *et al.*, 1991]. The antecedents of CLASP rules are composed of unary predicates (corresponding to concepts) and binary predicates (corresponding to roles). Thus they can be viewed as constraint networks. We are encouraged by Yen's analysis which found their algorithm's complexity to be polynomial in "normal" cases [Yen, 1990].

5.2 Plan Recognition

5.2.1 Kautz

Our plan recognition work is most closely related to that of Kautz [Kautz, 1991]. Our plan recognition technique, like Kautz's, is deductive, and incorporates the use of a plan abstraction taxonomy (as well as the traditional hierarchy decomposing plans into constituent actions). Both approaches are also restricted compared to other techniques in that they do not chain on state information (e.g. preconditions and effects), and have strong assumptions such as plan library correctness and completeness. Kautz's landmark work produced a formal theory of plan recognition based on

[7] Not to be confused with the homonymous CLASP system of [Devanbu and Litman, 1991].

circumscription, along with more practical algorithms that approximate his theory. A major contribution was his logical characterization of the completeness assumptions. In contrast, we have not focused on formalizing our own view of plan recognition.

There are several reasons to prefer T-REX to Kautz's implementation. Unlike Kautz's approach, we extend work in TKR to formalize and automate the organization of the plan taxonomy. Moreover, we directly exploit the library's definitional nature to guide plan recognition. Kautz's system uses a temporal language for relating actions that is more restricted than the one we use via MATS. We also use an underlying TKR system, K-Rep, to represent and reason with the actions and objects that are the building blocks of plans, whereas atomic actions and objects in [Kautz, 1991] and many other approaches lack defined semantics. Thus our approach allows the plan recognition system to share the advantages of existing terminological ontologies. We permit observation of actions at an abstract level, as well as revision 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 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.

5.2.2 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 which 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 which further refine the extracted relations. However, it is not clear how or 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 inverse subsumption mappings in our framework).

Song and Cohen proposed an algorithm to infer strong temporal constraints between a plan and its substeps. 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 are now implementing 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 with 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.

5.2.3 Intention-based 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, 1991] and [Song, 1991]. For reference, discussions of intention-based plan recognition are contained in [Kautz, 1991] and [Song, 1991].

6 Future Directions

This section presents some important avenues for continuing our work. We group them (somewhat arbitrarily) into four classes: semantics, algorithmics, plan language extensions, and additional inferences.

6.1 Semantics

Two superficially obvious alternatives for organizing plan hierarchies are by *part-of* relations and by *is-a* relations. It can be important to establish these relations because they may license useful inferences. For example, the *is-a* relation is key because it sanctions inheritance of information. Unfortunately, it is not yet clear how to handle inheritance with respect to our structural plan subsumption. Various part-of relations also license particular inheritance inferences, e.g., location can be inherited along *physical part-of* relations. Our structural plan subsumption seems to combine aspects of *is-a* and *part-of*, and we are eager to study their interaction in an integrated plan subsumption framework.

Let us consider how *is-a* and *part-of* relate to our structural plan subsumption. Figure 11 shows how two plans, MAKE-PASTA-DISH and ASSEMBLE-CHICKEN-MARINARA, both subsume plan ASSEMBLE-S&C-M, which describes a way to prepare a spaghetti and chicken marinara dish. Dashed lines indicate the subsumption mappings from nodes in the subsuming plans to nodes in the subsumed plan. This is indeed an interesting structural relationship, similar to those found in concept languages, in that each subsumer describes a different portion of the subsumee. Moreover, these portions partially overlap. Whether we choose to say that the subsumee *is-a* MAKE-PASTA-DISH, or *is-a* ASSEMBLE-CHICKEN-MARINARA, or both, or neither, may be a matter of taste as much as anything else. At any rate, in our framework a subsumer describes part but not necessarily all of a subsumee, and perhaps in a generalized way by means of more general action types and/or temporal constraints. As we have seen, this analytical relationship is a powerful tool for organizing a plan library in service of plan recognition. However, we still seek a better characterization of its meaning. We will revisit the semantics of our structural plan subsumption inference when we consider inheritance in Section 6.4.4.

Knowledge representation researchers commonly assign meaning to a formalism by specifying its model theoretic semantics, i.e., by stating the set theoretic denotations of its syntax. Doing this for T-REX would be an interesting and perhaps useful exercise.

6.2 Algorithmics

This section is concerned with speeding the computation of subsumption, classification and recognition inferences defined earlier. This goal might be achieved by

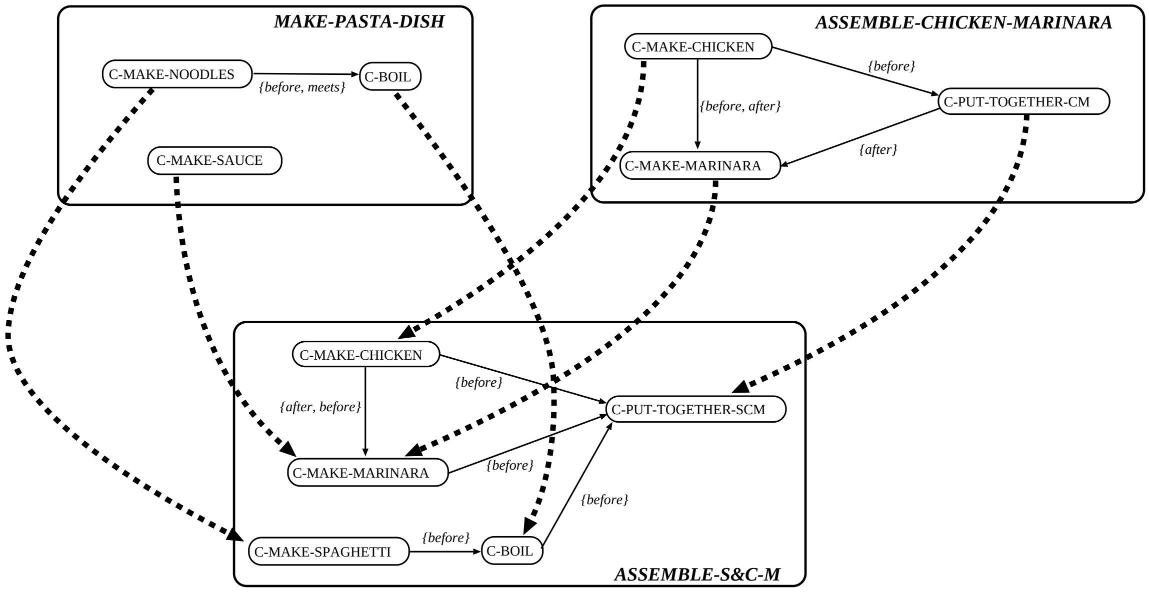

Figure 11: Structural Subsumption

designing improved algorithms and/or by restricting the problem to simpler cases.

6.2.1 Subsumption Algorithms

The search for a subsumption mapping has a combinatorial nature because nodes are matched according to the semantics of their associated concepts. As noted earlier, constraint network subsumption exemplifies the constraint satisfaction problem. CSP has been carefully studied, and a wide variety of techniques have been proposed. Their relative merits should be studied in the context of our application. For example, we will experiment with the tradeoffs involved in interleaving search to achieve (partial) arc consistency with the backtracking, in using intelligent (e.g., dependency-directed) backtracking, and in exploiting domain-specific techniques.

Here, we will point out just a few of the heuristic opportunities. A node *N1* representing an action *A1* in plan network *T1* can only subsume those nodes in network *T2* having actions subsumed by *A1*. Often this will be a small subset of the nodes in *T2*. Similarly, an arc in *T1* with an associated constraint *C1* can only subsume those arcs in *T2* whose constraints are subsumed by *C1*. Moreover, the constraints relating node *N1* to other nodes in temporal network *T1* may further restrict the nodes in *T2* which might be subsumed by *N1*. Many of the arcs in *T1* may carry trivial constraints; these can be safely ignored. Designing a superior algorithm to take best advantage of such heuristic information in most cases is a central goal of our future work.

There is considerable overlap in the expressive power of Allen and metric temporal constraints, hence the Allen 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 which verified the first constraint need not also verify the second. Minimizing redundant tests is a desirable goal. However, we must balance the cost of duplicated effort against the cost of identifying the duplication. For example, the current T-REX subsumption algorithm verifies Allen constraints before metric constraints. Thus it could ignore all linear inequalities whose numeric operand is zero or infinity. Comparing metric constraints, however, is quite inexpensive so it is unclear that this strategy would be worthwhile.

6.2.2 Classification Algorithms

Classification can be accomplished by plugging our subsumption algorithm into a "plain vanilla" classification procedure. Of course, when we are classifying, macro expansion and network closure only need to be performed once per plan network. The plain vanilla approach has the merit of conceptual clarity, however it precludes the possibility of reducing or eliminating redundant computations across multiple plan subsumption tests. Performance considerations may dictate a more sophisticated approach. For example, when K-Rep installs a new concept in the concept taxonomy, it restricts testing to the local differences between existing concepts and their parents. Doing this with T-REX plans may be tricky, however, in cases where there is more than one subsumption mapping from a parent plan to a child plan. One might also cache the results of various computations. We intend to explore this idea as an avenue to speedier plan classification. We have implemented an initial plan classification algorithm and are gaining experience with it in practice.

6.2.3 Incremental Plan Recognition Algorithms

We expect to develop a strategy for efficient incremental plan recognition that takes maximum advantage of the plan library's terminological nature. Our thoughts are inspired by traditional concept classifiers which compute a set of most specific subsumers and a set of most general subsumees of the concept being classified. It must be emphasized that the analogy is not direct, because (1) we must consider the assumptions along with the structural relationship between the observations and the plans, (2) the structural relationship in question is potential, not just actual subsumption, and (3) plan recognition is an incremental process. Hence, we are not simply classifying the observation network within the plan taxonomy. Instead, to efficiently recognize plans on an incremental basis, we track the most specific necessary

(MSN) plans and the most general optional (MGO) plans with respect to the current observations.

Definition 14 *A plan is a MSN if it is necessary and none of its children are necessary.*

Definition 15 *A plan is a MGO if it is optional and none of its parents are optional.*

Recall Figure 9. The MSN and MGO sets jointly delineate the border between the necessary and optional plans. Neither set alone is sufficient to pinpoint the border since the children of MSNs may not be MGOs and the parents of MGOs may not be MSNs. Figure 12 illustrates this situation.

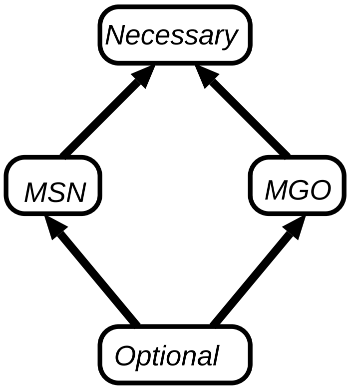

Figure 12: Disjoint MSN and MGO

The initial MSN set is {PLAN} and the initial MGO set contains the immediate descendants of PLAN. Similarly, we track a second (lower) frontier between the Most Specific Optional (MSO) and Most General Impossible (MGI) plans. The set of optional plans is thus sandwiched between the two frontiers.

It would be simple to explicitly associate a modality with each plan in the library. These modalities would be initialized as previously indicated. After updating the MSNs and MGOs to reflect new observations, and similarly for the second border, we can efficiently update the modalities of other affected plans through marker propagation. This approach seems likely to be cost effective for large taxonomies, but it must be acknowledged that maintaining the frontiers is not entirely trivial.

6.2.4 Simultaneous Plans

We should look for a better way to search for sets of plans that account for the observations when the single plan assumption proves unjustified. It might be possible to make use of information about the modalities of the single plans. An issue related to simultaneous plans is when and how substeps can be shared among several plans. This is discussed in Section 6.4.5.

In general, we may wish to find a minimum *cost* set of plans that potentially subsumes 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 should be considered in this context.

6.2.5 Restricted Plan Languages

For years, workers in the field of terminological knowledge representation endeavored to identify a terminological language that was both useful and tractable, e.g., [Brachman *et al.*, 1983, Patel-Schneider, 1984]. This effort now appears to have been a noble failure [Nebel, 1990]. Nonetheless, it seems worthwhile to consider restrictions on our plan representation language to see when and if performance advantages might accrue. The intractability of constraint network subsumption mapping follows from the combinatorics of the matching process, characterized by its reducibility from directed subgraph isomorphism (see Section 4.2.2). That is, the number of putative subsumption mappings that must be explored can be exponential in the size of the constraint network. However, some special cases of subgraph isomorphism are tractable, e.g., subtree isomorphism and problems with graphs satisfying a fixed degree bound. Might there be useful analogues to such special cases in plan subsumption? We also note that Vilain and Kautz derived a subset of Allen's interval calculus from a point-based representation which limits the disjunction within temporal constraints. In their restricted framework, complete closure of temporal networks is achieved in polynomial time. Might this restriction engender easier subsumption testing as well?

Due to the restricted nature of plan instances under assumptions such as perfect observation, we hope that subsumption-related processes carried out on plan instances during plan recognition will prove to be computationally easier. This remains an unexplored idea at present.

6.3 Plan Language Extensions

6.3.1 Integration of CLASP Operators

As pointed out in Section 5.1.1, T-REX's ability to express temporal relations among actions compares favorably with CLASP in some ways, but unfavorably in others. To our knowledge, CLASP [Devanbu and Litman, 1991] is the only plan subsumption system whose plan language can express temporal information that T-REX cannot. We now consider the prospects for bridging this gap.

CLASP composes actions via regular expressions, so it supports arbitrary disjunction and looping (recursively). 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.

Disjunction First, we note that when several concepts representing action types form a *cover* of a concept representing a more general action type, we may use the more general concept to indicate that any of the more specific action types is acceptable. For instance, given our sample action taxonomy and a closed-world assumption, a C-HEAT action type effectively sanctions either C-BOIL or C-BAKE. Moreover, it appears straightforward to support explicit disjunction of atomic actions in T-REX, e.g., a single action within a plan might be expressed as (OR MAKE-FETTUCINI MAKE-SPAGHETTI MAKE-ZITI). A subsumption mapping could map this to any action which expresses a subset of its disjuncts. An inverse subsumption mapping can map to this action just in case it could map to any of its disjuncts. Soon, we hope to take advantage of extensions to K-Rep that provide a full closed-world treatment of disjunction and negation [Dionne, *et. al.*, in progress].

It would be nice to express plans via disjunction over compositions of actions, but it does not seem possible to normalize such a language. In principle, one could iteratively generate the possible expansions and consider them individually, but this takes us far from our constraint network paradigm. Moreover, this prospect seems combinatorially daunting.

Looping Semantically, it appears possible for T-REX to support a repetition operator on atomic actions, denoted by an asterisk, e.g.:

```
(defplan MAKE-NOODLES*
  ((s1 (* c-make-noodles))))
```

Let us consider the effect on computing potential subsumption, by comparing the previous plan network with the following:

```
(defplan MAKE-SPAGHETTI-AND-MAKE-FETTUCINI*
  ((s1 c-make-spaghetti)
   (s2 (* c-make-fettucini))))
```

Note that that there is no temporal constraint between its two steps. We can see that MAKE-NOODLES* subsumes MAKE-SPAGHETTI-AND-FETTUCINI*. The potential images of (* C-MAKE-NOODLES) include all actions subsumed by C-MAKE-NOODLES and all repetitions of those actions. Moreover, potential subsumption mappings can now be *one-many*, as long as the images respect the constraints on the domain element. In keeping with the temporal constraint network paradigm, we simply require that each of the mapped-to nodes respect any temporal constraints on the node they are mapped from. Our * operator allows overlapped actions, so in that sense it is more general than CLASP's loop operator. While it would not be difficult to augment our plan subsumption algorithm for one-to-many mappings, the combinatorial impact on performance could be drastic.

In the case of plan recognition, we would have to support *many-one* mappings from the observation network to a plan network. For instance, the following observation network instantiates the above MAKE-NOODLES* plan:

- OBS-10: MAKE-FETTUCINI14 {*before*} MAKE-SPAGHETTI15 {*meets*} MAKE-ZITI16

Sequencing, Repetition, and Conditional Action There are several constructs in CLASP [Devanbu and Litman, 1991] which can easily be added to the T-REX plan language as syntactic sugar via macros, e.g., *sequence* and *repeat*. Their system also supports conditional actions e.g.:

```
(case (c-state1 c-action1)
      (c-state2 c-action2))
```

For atomic actions, we may be able to handle this construct as in CLASP. Conditionalizing subplans appears troublesome in a manner similar to disjunctive subplans (see Section 6.3.1).

6.3.2 Extended Coreference Constraints

For some applications it may be useful to support additional coreference constraint operators such as $\neq$, $\subset$ or $\supset$. We might also extend the role portion of coreference 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.

6.3.3 Preconditions, Goals and States

Plan networks serve to describe the "bodies" of plans. As proposed in [Devanbu and Litman, 1991], we might separately classify plans according to concepts describing their circumstances and/or purpose. These plan classifications and the plan classification via plan bodies we have discussed heretofore are orthogonal to one another. One interesting direction would be to explore their interaction in the context of plan recognition (see Section 6.4.6). At any time, one can consider executing the set of plans whose initial conditions subsume the current state. Similarly, to achieve certain conditions, one can seek to execute some plan whose goal is subsumed by those conditions.

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.

The problem becomes far more complicated if we similarly associate conditions with actions. As in [Allen, 1991], each action might have a set of associated conditions related to it by some temporal structure. In the terminological framework, these actions could be represented by temporal concepts similar to those of [Schmiedel, 1990]. Like RAT [Heinsohn *et al.*, 1992], we should reconcile the sets of conditions associated with all the actions comprising a plan with one another, and with the conditions of the plan itself. RAT addressed plans which are simple sequences of actions having preconditions and postconditions. 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 a difficult problem 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 terminological plan recognition.

6.3.4 Plan Roles

Plans might be given roles, just like the roles of standard concepts, which must be factored into subsumption. Plan roles could be used to represent parameters of a plan such as *agent*. 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/her underlings. Naturally, coreference constraints on a plan might correlate parameters of the plan with parameters of its actions. Here is an example, with a list of roles shown between the plan's name and its steps. In this case there is only an agent role. Note that the coreference constraints employ role chains and the special label *:self* refers to the plan itself.

```
(defplan EXECUTIVE-BOIL-SPAGHETTI
  ((agent executive-chef))
  ((s1 c-make-spaghetti)
   (s2 c-boil))
  :allen-constraints ((s1 (before meets) s2))
  :coref-constraints ((equal (agent :self)
                             (supervisor agent s1))
                             (supervisor agent s2)))
```

Ideally, one would simply extend the K-Rep concept language to encompass plan concepts. Then the body of the plan would simply be another role with a distinguished name.

6.4 Additional Inferences

As we have discussed, the plan language extensions contemplated in Section 6.3 would require corresponding extensions to our plan subsumption and recognition inferences. This section goes on to examine further inferences.

6.4.1 Forward Chaining

T-REX could support forward-chaining inference rules similar to those of CLASSIC [Borgida *et al.*, 1989]. For instance, it is perhaps reasonable to declare that spaghetti is always boiled. Then, recognition of a trivial plan description consisting solely of a C-MAKE-SPAGHETTI action could trigger a rule that immediately adds a subsequent instance of C-BOIL to the observation network. The early presence of a C-BOIL "observation" may permit earlier recognition of more complex plans containing a C-BOIL action. [8] Of course, preemptive recognition of this sort would have to be reconciled with actual observations later on.

[8] Indeed, a program that analyzes the plan library might notice that every plan which contains a C-MAKE-SPAGHETTI action also contains a subsequent C-BOIL action. It might generate the aforementioned rule automatically, justified by the complete plan library assumption.

6.4.2 Consistency with Domain Theory

T-REX checks the consistency of plan definitions by verifying that the stated temporal constraints are satisfiable (within the competence of the propagation algorithm) and that the coreference constraints are satisfiable. It cannot further verify the plausibility of plans without some knowledge of the domain. Since the T-REX architecture is domain-independent, we will build an inference rule facility for users to represent domain-specific integrity constraints on plans. When a new plan is defined, it would be verified against the set of inference rules. These rules operate at the *terminological level* on plan descriptions, whereas the rules of Section 6.4.1 operate at the *assertional level* on instances. Suppose that in our cooking domain, we wish to limit use of the oven to one action at a time. Concepts describing actions which use the oven are subsumed by the following K-Rep concept:

```
(defconcept c-use-oven
  (and c-action
       (all instrument c-oven)))
```

The following rule states the constraint we have in mind:

```
IF ((act1 c-use-oven) (act2 c-use-oven))
THEN ((act1 (before after) act2))
```

The antecedent of this rule matches all pairs of USE-OVEN actions (using our existing plan subsumption code) while the consequent asserts an additional temporal constraint between those actions. Consequently, domain rules may further refine plan definitions. If an inconsistency results, the plan is ill-formed with respect to the domain theory.

6.4.3 Feedback between Plan Library and Observed Constraints

Under the complete library, single plan and monotonic observation assumptions, the candidate plans will have temporal constraints and constraints on their actions which are the intersection of the constraints in the observations with constraints in the plan library. For example, consider the plan library in Appendix A and the following observations:

- MAKE-CHICKEN17 {*before, overlaps*} MAKE-SAUCE18

One can deduce that the actual events which transpired must be:

- MAKE-CHICKEN17 {*before*} MAKE-MARINARA18

It is not yet clear how to identify these new constraints precisely, however we should ideally enhance T-REX to propagate them through the observation network, improve the plan library partitioning if it can, and repeat the cycle until arriving at a fixpoint. This may require T-REX to coordinate inferences in K-Rep and MATS.

6.4.4 Inheritance

Today, there is no inheritance of information in the T-REX plan taxonomy. Although inheritance is an intuitively appealing benefit of *is-a* hierarchies, it is far from obvious when and how information can be inherited in our framework. This somewhat surprising state of affairs stems from the fact that our structural plan subsumption relationship, while clearly useful for characterizing plans at varying levels of abstraction, is different from an is-a relationship (as discussed in Section 6.1). For example, we cannot automatically inherit preconditions or effects along structural plan subsumption links. Consider that while the MAKE-SPAGHETTI-MARINARA plan may require certain ingredients beforehand, they need not be preconditions of a GO-SHOPPING-THEN-MAKE-SPAGHETTI-MARINARA plan, even though the latter is subsumed by the former. More thought must be devoted to understanding the interaction of structural plan subsumption and inheritance inferences.

6.4.5 Conditional Substep Sharing

In many applications, when more than one plan is required to account for the observations, we may need to decide on a case-by-case basis whether it is appropriate to share a particular observed action instance among the plans, i.e., by mapping it to actions from different plans within a multiple plan network. For example, two sequential cooking plans may require a freshly cleaned frying pan. The pan need only be removed from the cabinet once, but it will still need to be cleaned twice. This is a very difficult problem in general, since it entails solution of the notorious *frame problem* [McCarthy and Hayes, 1968]. Hence this problem will not be an early emphasis, but we imagine a domain-specific inference rule facility to provide limited guidance.

6.4.6 Intentional Plan Recognition

When several alternative plans are possible with respect to the observations, our deductive plan recognition methodology provides no basis for favoring one over another. If we incorporate state information in plan networks, along with preconditions and effects of actions and plans, then it should be feasible to integrate our deductive plan recognition with intention-based approaches such as those mentioned in Section 5.2.3. This would entail chaining on preconditions and goals of actions and plans. For example, we might use intention-based reasoning to select preferences among a set of optional plans identified by our methods. [9] As an example, suppose we have recognized MAKE-SPAGHETTI-PESTO and MAKE-FETTUCINI-ALFREDO as the possibilities. If there is a goal to avoid garlic, then the latter plan should be preferred.

6.5 Continuous Plan Recognition

In some applications, e.g., intelligent user interfaces to operating systems, users may carry out many plans over an extended period of time. These applications create a need for *continuous plan recognition*. In this setting, we would need to move beyond the minimal cardinality assumption which quickly becomes inadequate to control searching. 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. More drastically, once a single plan has been recognized, we might remove that entire portion of the observations. 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., the plan recognition would be incomplete but it would remain sound.

[9] Probabilistic reasoning would be another way to choose among alternatives. Although we do not propose to integrate probabilistic reasoning, there is nothing about our approach which would rule it out. This would require probabilities associated with actions and/or plans to indicate the likelihood of their occurrence. For this to be effective, however, we need complete and accurate information about the probabilities. Such information is usually difficult to come by.

6.6 Systems Issues

So far, our discussion has dealt with a passive form of plan recognition, namely the ability to answer queries regarding the possibility or necessity of a plan. In the capacity of user interface tool, T-REX should move towards *active plan recognition* which takes the initiative to report or act when the modality of a plan changes. This should be a straight-forward implementation matter. We should provide an escape to the host language (currently Common Lisp) which is triggered by designated modality changes on particular plans. Such *demons* might perform a service for the user of a software system, e.g., to prefetch a large file over the network when it is recognized that the user will act on it.

6.7 Orthogonal Constraint Networks

The constraint satisfaction problem characterizes many important problems in AI and computer science at large [Kumar, 1992]. Often it is formulated in terms of constraint networks. Our methods apply whenever it is useful to reason about structural subsumption between constraint networks or to recognize partial instances of constraint networks via potential subsumption. We now sketch an application 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], arbitrary 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-RECTANGLE in 2-dimensional space:

- C-SQUARE ({*before, after*}, {*before, after*}) C-RECTANGLE

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 which is properly contained in the C-SQUARE then it is spatially disjoint from the C-RECTANGLE.

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 in each dimension. Thus we 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-RECTANGLE subsumes C-SQUARE, etc.

Our formulation of potential subsumption also extends directly to multiple dimensions. Spatial subsumption and potential subsumption may be useful for computer vision and graphics tasks. Potential subsumption can recognize spatial configurations of objects described by library entries from partial observations recorded in orthogonal observation networks.

7 Potential Applications

As one concrete application for our work, we have been considering use of T-REX to enhance the capabilities of the FAME expert system [Apte *et al.*, 1992]. FAME supports financial marketing of IBM mainframe computer systems. Its principle problem solvers are a mainframe equipment planner (MEP) and a financial analyzer. The MEP searches for suitable strategies to acquire, deacquire and upgrade products (from IBM or a competitor) to meet a user's computing needs over an extended time period. The financial analyzer helps a marketing representative to evaluate different proposals for financing the purchase of computer equipment. FAME's problem solving components and user interface are all constructed on top of, and integrated through K-Rep. FAME is particularly hospitable to T-REX since all of FAME's input and output is already done via presentation and acceptance of K-Rep concepts. However, FAME's present user interface strongly directs the interaction in a top-down manner, minimizing the opportunity for useful plan recognition. Less restrictive paradigms have been considered. For example, the interaction might be geared towards bottom-up construction of an *argument graph* showing that (1) the customer will need additional computing resources and (2) the IBM proposal is more attractive than its competition. A bottom-up, user-directed problem solving control strategy would allow the user greater flexibility in working on a problem, and would provide a rich setting for plan recognition. Unfortunately, while redesign of FAME's interaction paradigm is interesting in its own right, it could well be beyond the scope of this work.

Alternative applications to be considered include (1) clinical information systems, (2) travel consultation, (3) interactive, adaptive computer help systems [Selker, 1989], and (3) equipment maintenance and repair. We are also aware that corporations are increasingly interested in rigorous analysis of business processes. T-REX can model processes just as it models plans. One application in this area might be completion, submission and approval of business forms [10].

8 Evaluation

Our work must be evaluated in terms of both utility and performance.

An excellent way to demonstrate the utility of T-REX would be to create a successful plan-based interface to a system which is already useful in its own right, as envisioned in Section 7. This would be a landmark achievement, since to our knowledge, no deployment of a generic plan recognition system in a practical application has ever been reported.

While we are optimistic that our ideas can be deployed, extraneous factors may stand in the way. That is, we may lack both access to a suitable existing application and the time and resources required to build a fully deployed application for the purpose of demonstrating our research. In that case, we would propose to build a substantial demonstration application, in a domain other than cooking, that is sufficient in size and scope to fully exercise T-REX and convincingly suggest its potential. Besides classifying a library of plans and illustrating the recognition of candidate plans from observations, we would demonstrate that the system can provide helpful services to the user, e.g., by taking preemptive actions and making recommendations.

In a different direction, it would also be possible to evaluate T-REX as a theoretical system by formally specifying a syntax and semantics for plans, then characterizing the plan recognition problem in terms of a model theory and a proof theory. This could be compared with the work of Kautz [Kautz, 1991].

In terms of performance, we conjecture that our terminological plan inferences can be computed with acceptable speed. Obviously, a successful application would go a

[10] Thanks to Eric Siegel for mentioning this application.

long way towards justifying this claim. However, we wish to study the performance of our algorithms more methodically. To this end, we anticipate conducting an empirical analysis of plan classification and recognition techniques similar to our current study of terminological concept classification (with Eric Mays, in preparation). We now briefly outline that work to suggest its relevance here.

The recent wave of discouraging tractability results, combined with feedback from users [Doyle and Patil, 1991], suggests renewed focus on terminological algorithms which perform well in practice. Eric Mays and I are working on an empirical study of classification methods. Mays has designed a new classifier architecture which supports compile-time selection and combination of techniques for computing subsumption and classification. We are studying the effectiveness of the techniques, alone and in combination, with respect to the varying characteristics of knowledge bases. Unfortunately, we do not have many large knowledge bases available, nor is it practical to construct them by hand for our study. Instead, I have implemented a "workbench" consisting of a knowledge base synthesizer, a knowledge base analyzer, and an experiment manager. The synthesizer generates artificial knowledge bases according to settings of its "knobs". However, these knobs do not correspond exactly to the set of attributes we use to characterize knowledge bases. Thus, a separate analyzer statistically measures KB properties. By examining numerous points in the space of possible KBs, we can determine how particular techniques effect performance, and which combinations of methods are appropriate in given circumstances.

9 Conclusion

We extend the scope of TKR by showing how to compute structural subsumption relationships among constraint networks such as temporal networks used in plan representation. In the case of plans, we use K-Rep to compute subsumption on structured action concepts and we also compute subsumption on temporal constraints and coreference constraints. This allows us to automatically organize a plan library into a definitional taxonomy, thereby easing search and maintenance tasks. We further exploit the plan library's terminological nature in a new and promising approach to plan recognition that partitions the plan library by modality. Our framework supports arbitrary revision of prior observations. We have explored our ideas in T-REX, a system whose modular architecture utilizes state of the art components: K-Rep for standard TKR and MATS for temporal reasoning. Our ideas apply to constraint networks in general, and we have proposed a representation, subsumption and recognition facility for configurations of objects in N-dimensional space. There are many interesting and challenging directions along which to further develop this work.

9.1 Contributions

For handy reference, we now list what we see as the principal contributions of our research (Section 6 suggests possible future contributions):

- Synthesis of terminological knowledge representation with constraint network reasoning
- Treatment of temporally rich plans in a terminological framework
  - Plan representation (metric, Allen and coreference constraints)
  - Integrity constraints
  - Plan subsumption algorithm
  - Plan library classification
- Terminological approach to plan recognition
  - Partitioning of plan taxonomy by modality
  - Potential subsumption inferences under varying assumptions regarding the observations
    * Inverse subsumption mapping
    * Compatibility mapping
  - Efficient and incremental algorithms
  - Revision of prior observations
  - Simultaneous plans
- Integration of state of the art systems
  - K-Rep (Mays, Dionne and Weida)
  - MATS (Kautz and Ladkin)
- Subsumption, classification and recognition of orthogonal constraint networks

9.2 Agenda

To complete my dissertation research, I propose to accomplish at least the following things, in the following order. Time estimates are given for each task.

1. Formally extend definitions and theorems concerning plan subsumption and plan recognition to encompass metric temporal constraints and equality constraints. [2 additional weeks]

2. Design and implement an incremental plan recognition algorithm which reduces the number of comparisons between plans and the observation network in two ways. First, whenever the observation network is updated, exploit the previous partitioning of the plan library to compute the new one. Second, propagate the consequences of comparing one plan with the observations to other plans. [1 additional week]

3. Provide a facility for ensuring domain-specific integrity constraints on plan definitions via inference rules, as suggested in Section 6.4.2. The antecedant of an inference rule has the same form as a plan, so a rule matches a plan just in case its antecedant subsumes the plan. This is determined by T-REX's existing plan subsumption algorithm, which returns the appropriate bindings for matches. The rule's consequent simply refines the plan, i.e., by adding action, temporal and/or coreference constraints, quite similar to when the plan was first defined. Whenever a new plan is defined, the rules will be applied (repeatedly) to monotonically refine constraints in the plan. This process continues until either no more rules are applicable or an inconsistency arises. Note that I am not promising more than a trivial algorithm for selecting rules to fire. [2 weeks]

4. Implement a *demon* facility which executes specified Lisp code when the status of a particular plan changes from a given modality and/or to a given modality. This facility will permit triggering of demons to be conditioned on the results of queries to K-Rep. The incremental plan recognition algorithm which I have implemented already changes the explicit modality associated with a particular plan when and if appropriate. [2 weeks]

5. Construct a demonstration library of approximately fifty plans which fully illustrates the T-REX plan language. This entails construction of an underlying K-Rep concept taxonomy. The plan library will be subjected to domain-specific integrity constraints and several of the plans will be refined accordingly. [3 weeks]

6. Specify a representative set of demons in conjunction with the demonstration library. When triggered, these demons will display messages to indicate useful and appropriate actions which could be taken if T-REX were employed by the user interface of an application system in the chosen domain. [1 week]

7. Show the potential utility of terminological plan recognition through automatic invocation of these demons during the course of approximately twenty observation network sequences. The sequences will include examples conforming to both the *perfect observation* and *monotonic observation* assumptions. [1 week]

8. Empirically compare the performance of T-REX's incremental and non-incremental (naive) plan recognition algorithms under the *complete library* and *single plan* assumptions for each of these sequences. [2 weeks]

9. Build a metering facility which can time plan library classification and/or count basic inferences carried out during classification (e.g., concept subsumption queries to K-Rep). [1 week]

10. Illustrate the basic capabilities of the metering facility by applying it to the demonstration library. [1 week]

11. Design and implement a plan library synthesizer which can use action concepts in the demonstration concept taxonomy to randomly generate sets of plan libraries for the purpose of performance analysis. Input parameters will control several plan library characteristics, including the number of plans in the library and the average number of actions per plan. In this effort, I will benefit from similar work I have already done to synthesize K-Rep concept taxonomies. The goal of this task and the next one is confined to empirically measuring the performance of T-REX. I do not propose to advance the state of the art in performance analysis methodology. [3 weeks]

12. Apply the metering facility to a suite of plan libraries generated by the synthesizer, so as to evaluate the impact of varying different plan library characteristics. These characteristics will include the number of plans in the library and the average number of actions per plan. This step will help to evaluate how well the T-REX plan subsumption algorithm scales up. [2 weeks]

The work described above totals 21 weeks in my conservative estimation, to which I add 2 weeks for unforeseen contingencies. In addition, I estimate the writing component as follows: 8 weeks for the first draft of the thesis, 3 to initially revise it with respect to my advisor's comments, and 3 to address further comments from my entire committee. Of course, I will have been seeking the advice and counsel of my committee on an ongoing basis throughout the remainder of my thesis work. This allows 23 weeks for research and 14 weeks for writing, yielding a grand total of 37 weeks.

We now more fully categorize future research directions by priority to guide our future work. We have endeavored to strike a balance between matters of theoretical interest and matters of practical import, with the expectation that our priorities will probably evolve according to the availability of a suitable application system (and also in consultation with our dissertation committee). References are made to the pertinent portions of Section 6.

9.2.1 High Priority

- Formally extend definitions and theorems concerning plan subsumption and plan recognition to encompass metric temporal constraints and coreference constraints.
- Build a substantial application or demonstration system (Section 7).
- Fully design and implement an incremental plan recognition algorithm (Section 6.2.3).
- Performance analysis of implemented plan subsumption and plan recognition algorithms (Section 8).
- Provide a facility for ensuring integrity constraints via inference rules (Section 6.4.2).
- Study the applicability of our ideas in settings which call for continuous recognition (Section 6.5).
- Endow plans with roles (Section 6.3.4).

9.2.2 Medium Priority

- Investigate alternative plan subsumption algorithms (Section 6.2.1).
- Seek better ways to explore plan combinations when no single plan suffices to account for the observations (Section 6.2.4).
- Integrate disjunction and looping to the extent feasible (Section 6.3.1).
- Refine recognition of plans via feedback between constraints in the plan library and observed constraints (Section 6.4.3).
- Formal semantics of plan subsumption and recognition (Section 6.1).
- Discern the interrelationship among our structural plan subsumption, *is-a* and *part-of* (Section 6.1).
- Investigate the place of inheritance in structural plan subsumption (Sections 6.1 and 6.4.4).

9.2.3 Low Priority

- Explore alternative plan classification algorithms (Section 6.2.2).
- Analyze the complexity of subsumption and/or recognition with restricted plan languages (Section 6.2.5).
- Integrate additional CLASP operators (Section 6.3.1).
- Extend expressiveness of coreference constraints (Section 6.3.2).
- Integrate state information into the plan representation, including preconditions and effects of actions and plans; augment subsumption and recognition algorithms accordingly (Section 6.3.3).
- Integrate our plan recognition methodology with intentional plan recognition techniques (Section 6.4.6).
- Implement forward chaining from recognized patterns (Section 6.4.1).
- Conditionalize substep sharing in plan recognition (Section 6.4.5).
- Further develop the use orthogonal constraint networks in spatial reasoning (Section 6.7).

A Sample Plan Library

The following definitions were used by T-REX to construct the plan taxonomy in Figure 7. As used in this appendix, the function "defplan" takes three arguments. Informally, the first argument names the plan type being defined. The second argument is a list of plan steps. Each step specifies a label for the step and a constraint on its action type. Any action instance satisfying the constraint must be subsumed by the action type. The third (keyword) argument is a list of Allen's temporal constraints. Each constraint specifies a disjunction of Allen's 13 temporal relationships between the temporal intervals associated with the two designated plan steps. For example, the first definition states that any instantiation of the plan HEAT-NOODLES satisfies the following constraints: it contains an action instance of type C-MAKE-NOODLES; it contains an action instance of type C-HEAT; the temporal interval associated with the first action instance is *before* or *meets* the temporal interval associated with the second. Also note the use of the plan BOIL-SPAGHETTI as a macro action in the definition of ASSEMBLE-SPAGHETTI-MARINARA.

```
(defplan HEAT-NOODLES
  ((s1 c-make-noodles)
   (s2 c-heat))
  :allen-constraints ((s1 (before meets) s2)))

(defplan BOIL-NOODLES
  ((s1 c-make-noodles)
   (s2 c-boil))
  :allen-constraints ((s1 (before meets) s2)))

(defplan HEAT-SPAGHETTI
  ((s1 c-make-spaghetti)
   (s2 c-heat))
  :allen-constraints ((s1 before s2)))

(defplan BOIL-SPAGHETTI
  ((s1 c-make-spaghetti)
   (s2 c-boil))
  :allen-constraints ((s1 before s2)))

(defplan MAKE-PASTA-DISH
  ((s1 c-make-noodles)
   (s2 c-boil)
   (s3 c-make-sauce))
  :allen-constraints ((s1 (before meets) s2)))

(defplan MAKE-SPAGHETTI-MARINARA
  ((s1 c-make-spaghetti)
   (s2 c-boil)
   (s3 c-make-marinara))
  :allen-constraints ((s1 (before meets) s2)))

(defplan ASSEMBLE-SPAGHETTI-MARINARA
  ((bs boil-spaghetti)
   (s3 c-make-marinara)
   (s4 c-put-together-sm))
  :allen-constraints (((s2 bs) (before meets) s4)
                      (s3 (before meets) s4)))

(defplan MAKE-SPAGHETTI-PESTO
  ((s1 c-make-spaghetti)
   (s2 c-make-pesto)
```

```
   (s3 c-boil))
   :allen-constraints ((s1 (before meets) s3)))

(defplan MAKE-FETTUCINI-ALFREDO
  ((s1 c-make-fettucini)
   (s2 c-make-alfredo)
   (s3 c-boil))
   :allen-constraints ((s1 (before meets) s3)))

(defplan MAKE-MEAT-DISH
  ((s1 c-make-meat)
   (s2 c-make-sauce)))

(defplan MAKE-MEAT-MARINARA
  ((s1 c-make-meat)
   (s5 c-make-marinara)))

(defplan ASSEMBLE-CHICKEN-MARINARA
  ((step1 c-make-chicken)
   (step2 c-make-marinara)
   (step3 c-put-together-cm))
  :allen-constraints
    ((step1 (before after) step2)
     (step1 (before) step3)
     (step3 (after) step2)))

(defplan ASSEMBLE-S&C-M
  ((ms c-make-spaghetti)
   (b c-boil)
   (mc c-make-chicken)
   (mm c-make-marinara)
   (pt c-put-together-scm))
  :allen-constraints ((ms before b)
                      (b before pt)
                      (mc before pt)
                      (mm before pt)
                      (mc (after before) mm)))
```

B Plan Synthesis

Although beyond the scope of the proposed research, it is natural to wonder how temporal plan subsumption might be used in plan generation. For a temporal planner such as Allen's [Allen, 1991], it would seem obviously useful to classify a library of plans which serve as *macro operators*.

Can we accomplish something more ambitious? Following is an extremely speculative abstract for hypothetical future work. It suggests a combination of Allen's temporal planner [Allen, 1991] and Wellman's dominance-proving planner [Wellman, 1990] which uses subsumption to organize a search space of atemporal plans. We call this vaporware system T-SYN, for Terminological Plan Synthesizer.

T-SYN is a dominance-proving temporal planner which integrates and extends ideas of Allen and Wellman in a new planning methodology based on terminological reasoning with temporally rich plans. T-SYN is a companion to T-REX, a terminological plan recognition system. Like Allen's temporal planner, T-SYN reasons about actions and persistence assumptions using an explicit temporal logic. Like Wellman's SUDO-PLANNER, it derives multiple plan classes (partial plans) by posting constraints at varying levels of abstraction. Thus it can accommodate partially satisfiable goals. In T-SYN, constraints of the following types may be posted:

- Introduce an action in a plan.
- Constrain the type of an action.
- Constrain the temporal relationship between a pair of actions.
- Assert a coreference constraint among roles of one or more concepts.

T-SYN, like the atemporal SUDO-PLANNER, organizes partial plans into a taxonomy representing the explored portion of the search space, thereby preventing redundant search. However, T-SYN exploits temporal plan subsumption and classification technology introduced in T-REX. Wellman defines a plan class P1 to "dominate" another, P2, if for every plan in P2, there is a plan in P1 which is preferred or indifferent according to some preference relation. He then shows how dominance can be propagated in a plan taxonomy. T-SYN adopts Wellman's dominance-proving control strategy to focus search. Similarly, T-SYN can explore the search space with respect to Allen-style temporal persistence assumptions by adapting his notion of conditional dominance relations. In short, T-SYN combines the advantages of Allen's planner and Wellman's planner in a single system. It should be possible to share conceptual ontologies between T-SYN and T-REX. Indeed, they might serve together in applications that call for both planning and plan recognition. However, there is a not-exactly-trivial catch to applying these ideas. As Wellman notes, "The interesting task for the dominance prover is to come up with meaningful conditions that imply useful dominance relations."

C Proofs of Theorems

Theorem 1 *Terminological constraint network* T1 *subsumes terminological constraint network* T2 *iff there exists a subsumption mapping from* T1 *to* T2.

If: Clearly, the subsumption mapping demonstrates that any instance in the extension of *T2* is also in the extension of *T1*. □

Only If: We assume that closure of *T2* is complete. When there is no subsumption mapping from *T1* to *T2*, we will see that *T2*'s extension is not a subset of *T1*'s extension, so the notion that *T1* subsumes *T2* is contradictory. There are two cases to consider. First, the nodes of *T2* may not permit a mapping from *T1* with a distinct subsumee for each node in *T1*. Then *T1* contains at least one node without a counterpart *T2*. Second, the nodes of *T2* may permit such a mapping, but not so that every arc between a pair of nodes in *T1* subsumes the corresponding arc in *T2*. Then *T1* contains at least one arc which forbids a relationship sanctioned by its counterpart in *T2*. In either case, there clearly exists an instantiation of *T2* which is not an instantiation of *T1*, hence the contradiction. □

Theorem 2 *Subsumption mapping between terminological constraint 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 and no more than n^2 arc subsumption tests. It is also trivial to check that no two nodes in the subsumer are mapped to the same node in the subsumee.

There is a polynomial time transformation from directed subgraph isomorphism, which is NP-complete [Garey and Johnson, 1979], to subsumption mapping between terminological constraint networks. Digraphs $G1 = (V1, E1)$ and $G2 = (V2, E2)$ are transformed into terminological constraint networks *T1* and *T2*, respectively, as follows:

- Associate the primitive concept C-VERTEX with each element of *V1* and each element of *V2*.

- Associate the primitive concept C-EDGE with each element of *E1* and each element of *E2*.

Then, it is evident that *G1* contains a subgraph isomorphic to *G2* just in case there is a subsumption mapping from *T2* to *T1*. □

A directed graph is *complete* if there exists an edge from every vertex to every other vertex. For example, Allen-style temporal networks are complete following constraint propagation. Although subgraph isomorphism is trivial when both *G1* and *G2* are complete digraphs, subsumption mapping between a pair of complete terminological constraint networks can nonetheless be reduced from the general directed subgraph isomorphism problem:

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

Proof: The proof is similar. The transformation from *G1* to *T1* also adds an element $<u1,\ v1>$ to *E1* for every pair of elements *u1* and *v1* in *V1* such that $<u1, v1>$ is *not* in *E1*, and associates with it the primitive concept C-NON-EDGE. The transformation from *G2* to *T2* is analogous. Concepts C-EDGE and C-NON-EDGE are defined to be disjoint. Again, it can be seen that *G1* contains a subgraph isomorphic to *G2* just in case there is a subsumption mapping from *T2* to *T1*. □

Theorem 3 *Under the complete library, single plan and perfect observation assumptions, a plan is possible iff it potentially subsumes the observations.*

We refer here to potential subsumption as in Definition 6 on page 26.

If: There are two cases of potential subsumption. In the first case, there is an inverse subsumption mapping from observation network *O* to plan network *P*. If there is also a subsumption mapping from *P* to *O*, then by Theorem 1, *P* subsumes *O*, making *P* necessary and hence possible. Otherwise, it is clear that *O* could be extended to instantiate *P*, i.e., with nodes instantiating the remaining nodes of *P* as well as corresponding arcs instantiating the remaining arcs of *P*. Since *P* can thus become necessary, *P* is possible. In the second case, some plan *P'* is possible by the criterion just discussed, and *P'* is subsumed by *P*. By definition of subsumption, the extension of *P'* is a subset of the extension of *P*. Therefore, whenever *O* comes to instantiate *P'*, it must also instantiate *P*, and *P* will become necessary whenever *P'* becomes necessary. It follows that *P* is possible whenever *P'* is possible. □

Only If: According to the complete library and single plan assumptions, there is one particular plan in the plan library that is, in fact, being instantiated. Let us define this as the *actual plan*:

Definition 16 *Under the complete library and single plan assumptions, the* actual plan *is the single most specific plan in the plan library which is, in fact, being instantiated.*

Under perfect observation, it is easy to see that there must always be an inverse subsumption mapping from the observation network to the actual plan. Thus, only those plans which enjoy an inverse subsumption mapping from the observation network can be the actual plan. Hence, assuming that the plan library has been augmented as in Section 4.3.3, only those plans and their subsumers are possible, i.e., exactly the set of plans which potentially subsume the observations according to Definition 6. □

Theorem 4 *Under the complete library, single plan and monotonic observation assumptions, a plan is possible iff it potentially subsumes the observations.*

We refer here to potential subsumption as in Definition 13 on page 29. We generalize the proof of Theorem 3 to accommodate compatibility.

If: There are two cases of potential subsumption. In the first case, there is a compatibility mapping from observation network O to plan network P. Given the compatibility mapping, it is clear from the definitions of node and arc compatibility that O can be refined so there is also an inverse subsumption mapping from O to P. Then, as in the proof of Theorem 3, either P will subsume O or O will be extensible such that P can later subsume O. Since P is either necessary or can become so, P is possible. In the second case, some plan *P'* is possible by the preceding criterion, and *P'* is subsumed by P. By definition of subsumption, the extension of *P'* is a subset of the extension of P. Therefore, whenever O comes to instantiate *P'*, it must also instantiate P, and P will become necessary whenever *P'* becomes necessary. It follows that P is possible whenever *P'* is possible. □

Only If: Again, according to the complete library and single plan assumptions, there is an actual plan that is, in fact, being instantiated. Under monotonic observation, it is easy to see that there must always be a compatibility mapping from the observation network to the actual plan. Thus, only those plans which enjoy a compatibility mapping from the observation network can be the actual plan. Hence, assuming that the plan library has been augmented as in Section 4.3.3, only those plans and their subsumers are possible, i.e., exactly the set of plans which potentially subsume the observations according to Definition 13. □

References

[Allen and Perrault, 1980] J. F. Allen and C. R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15(3):143–178, 1980.

[Allen, 1983] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[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.

[Apte *et al.*, 1992] C. Apte, 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 1992.

[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.

[Beck and Gala, 1989] H. W. Beck and S. B. Gala, S. K.and Navathe. Classification as a query processing technique in the candide semantic data model. In *Proc. Fifth International Conference on Data Engineering*, pages 572–581, Los Angeles, CA, 1989.

[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 *et al.*, 1983] R. J. Brachman, R. E. Fikes, and H. J. Levesque. Krypton: Integrating terminology and assertion. In *Proceedings of AAAI-83*, pages 31–35, Washington, D.C., 1983. American Association of Artificial Intelligence.

[Brownston *et al.*, 1985] L. Brownston, R. Farrell, E. Kant, and N. Martin. *Programming Expert Systems in OPS5.* Addison Wesley, Reading, MA, 1985.

[Carberry, 1990] S. Carberry. *Plan Recognition in Natural Language Dialogue.* MIT Press, Cambridge, MA, 1990.

[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.

[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 *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.

[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.

[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.

[Heinsohn *et al.*, 1992] J. Heinsohn, D. Kudenko, B. Nebel, and H. J. Profitlich. Rat: Representation of actions using terminological logics. DFKI, 1992.

[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, 1991] 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.

[Koomen, 1989] J. Koomen. Localizing temporal constraint propagation. In *First International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 198–202, Toronto, Ontario, Canada, 1989.

[Kumar, 1992] V. Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, 13(1):32–41, 1992.

[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.

[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 and Bates, 1987] R. MacGregor and R. Bates. The loom knowledge representation language. Technical Report ISI/RS-87-188, USC/Information Sciences Institute, Marina del Ray, CA, 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, 1991] 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.

[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.

[Mark, 1981] W. Mark. Representation and inference in the consul system. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 375–381, Vancouver, BC, 1981.

[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.

[McCarthy and Hayes, 1968] J. McCarthy and P. J. Hayes. *Some Philosophical Problems from the Standpoint of Artificial Intelligence*, pages 463–502. Edinburgh University Press, Edinburgh, England, 1968.

[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, 1990] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

[Patel-Schneider, 1984] P. F. Patel-Schneider. Small can be beautiful in knowledge representation. In *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, pages 11–16, Denver, CO, 1984.

[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.

[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*.

[Robins, 1986] G. Robins. The nikl manual. Technical report, USC/Information Sciences Institute, Marina del Ray, CA, 1986.

[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 Lipkis, 1983] J. G. Schmolze and T. A. Lipkis. Classification in the kl-one knowledge representation system. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 330–332, Karlsruhe, West Germany, 1983.

[Selker, 1989] T. Selker. Cognitive adaptive computer help (coach). In *Proceedings of the International Conference on Artificial Intelligence*, pages 25–34. IOS, 1989.

[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.

[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.

[von Luck *et al.*, 1987] K. von Luck, B. Nebel, C. Pelatson, and A. Schmiedel. The anatomy of the back system. Technical Report KIT- Report 41, Technische Universitat Berlin, Berlin, 1987.

[Weida and Litman, 1992] R. Weida and D. J. 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, 1991] R. A. Weida. Object-centered knowledge representation and reasoning with definitional taxonomies. Technical Report CUCS-047-91, Columbia University Department of Computer Science, 1991.

[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. First appeared as: Harvard University/Center for Research in Computing Technology, Technical Report no. TR-19-90.

[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.