CUPID: A PROTOCOL DEVELOPMENT ENVIRONMENT

Yechiam Yemini and Nihal Nounou

CUCS-59-83

May, 1983

CUPID: A PROTOCOL DEVELOPMENT ENVIRONMENT

Yechiam Yemini & Nihal Nounou,
Computer Science Department,
Columbia University,
New York, NY 10027.

Abstract

This paper describes research conducted towards Columbia's Unified Protocol Implementation and Design (CUPID) environment. CUPID research aims at the integration and automation of protocol design and implementation tools. CUPID uses an algebraic representation of protocols based, in part, upon a variant of Milner's calculus of communicating systems (CCS). Communication behaviors are represented in terms of expressions of a universal algebra. A key notion to the automation of protocol development functions is that of a valuation over the algebra of communication behaviors. A valuation maps communication behaviors to expressions in other algebras (e.g., an algebra of delay distributions used for performance analysis). This allows one to proceed and compute attributes of communication behaviors over the respective algebras using a formal valuation process. We provide a brief introduction to CCS in the context of modelling protocol behaviors. This is followed by a brief summary of how the algebraic valuation mechanism may be used to support the different functions of a protocol design environment: multiple concurrent specifications, automated functional and performance analysis and automated test generation and performance simulation.

1. INTRODUCTION

1.1. BACKGROUND

Software environments for the design of VLSI and software systems are widely accepted as key instruments in addressing the complexities of large scale computing systems. Columbia's Unified Protocol Implementation & Design (CUPID) research aims at the construction of an environment to address issues specific to the design of message-based distributed processing systems. The objectives of this research are to address the design problems of communicating distributed systems and the problems of unifying and automating the functional and performance analysis of such software systems. Protocols, in this context, are not restricted to low-level protocols but are primarily high-level protocols such as those in

the top 3 layers of the ISO hierarchy. CUPID aims at the following goals:

1. Supporting multiple concurrent representations of the protocol. The problem here is that of allowing the user to operate on a variety of possible representations of the protocol, as required by his specific needs. The user can thus maintain a finite-state-automaton description of the protocol in one window over his terminal, a petri-net description of the protocol in another window and a high-level language description in still another window. As any one representation is changed by the user (i.e., by editing), the other representations are respectively changed by the system.

2. Supporting automated assistance in verification. This includes verification of the completeness of the specifications and safety and liveliness properties of the protocol.

3. Supporting automated performance analysis of the protocol. Such performance analysis, while crude, can provide the user with an almost instantaneous performance assessment of the implications of certain changes in the protocol. Such performance assessment may be presented to the user in terms of any one of the protocol representations.

4. Supporting automated simulation-debugging tools; including representation of the simulation model over which to test the protocol, automatic generation of test sequences and statistics gathering.

The development of a canonical semantic model of protocol behaviors is a key element in achieving these goals. A variant of the algebraic model proposed by Milner [5] was chosen as our point of departure. Milner's Calculus of Communicating Systems (CCS) provides a concise algebraic representation of concurrency and communication exchanges. Such representation of communication behaviors is axiomatized in terms of an equational algebra. The use of algebraic specifications of communication protocols has been pursued by Milne [4] and more extensively by G. Holtzman [1, 2, 3] to prove correctness of protocols.

CUPID utilizes CCS as its canonical protocol representation model. Protocol specifications are translated to the algebraic formalism. The algebraic representation may be used by a functional analysis module to verify the protocol, and by a performance analysis module to analyse the performance behavior of the protocol. Furthermore, the algebraic model may be used to generate a simulation model which is used to test/debug the functional behavior and extract statistics of performance behavior. This structure of CUPID is described in figure 1 below. There are 3 categories of tools provided by the environment: tools for the construction of the protocol (specifications layer), tools for the analysis of the behavior of the protocol (analysis layer) and tools for experimental evaluation of the protocol (simulation layer).

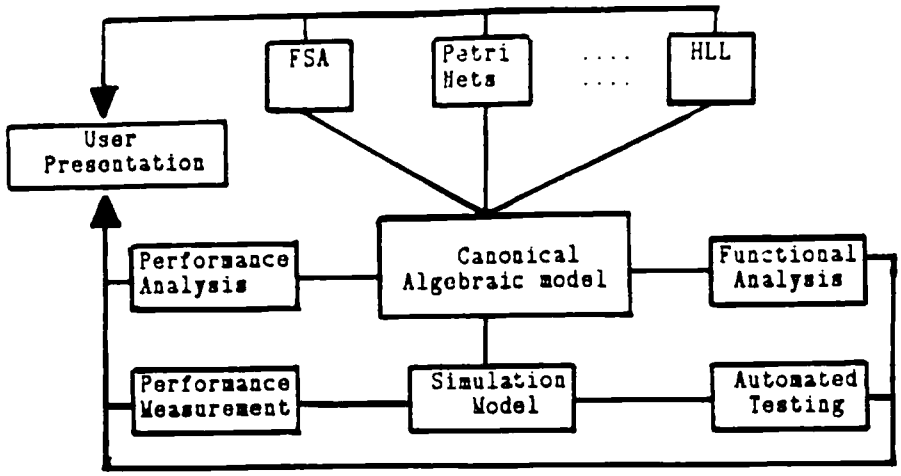


Figure 1: A functional view of CUPID

1.2. CCS AS A UNIFYING PROTOCOL MODEL

1.2.1. AN OVERVIEW OF CCS

CCS aims at providing an algebraic description of communication behaviors. A communication behavior may be described along two dimensions: the flow of communication events (i.e., send and receive) and the transmission of objects as a result of these events.

CCS describes communication events in terms of sending and receiving ports. Ports are denoted by names (for sending ports) such as α, β, msg_0 or co-names (for receiving ports) such as $\bar{\alpha}$, $\bar{\beta}$ and $\overline{msg}_0$. Send and receive ports may thus be associated through a common name to constitute a communication channel. A communication channel is simply a mechanism for processes to pass communication events between a sending port and receiving ports. The communication behavior of a process may be described in terms of the flow of communication events which it generates. CCS provides a few primitive operators to describe the flow of communication events:

- Sequential composition of behavior. For example, the behavior $\alpha.\bar{\beta}$ describes a sending event on port α followed by a receiving event on port $\bar{\beta}$.

+ Non-deterministic choice among behaviors. For example, the behavior $\alpha+\bar{\beta}$ describes a behavior of either a sending event on port α or receiving at port $\bar{\beta}$.

A behavior expression is either the null behavior (no events) *NIL* or is formed from behavior expressions and port names using the above operators. CCS also defines two derived operators: the **restriction** operator "\" and the **composition** operator "|". Both are defined recursively as follows: If $T=\Sigma\mu_i T_i$ and $U=\Sigma\upsilon_j U_j$ are two behavior expressions then:

$$\begin{cases} T\backslash\alpha \triangleq \sum_{\mu_i \neq \alpha} \mu_i(T_i\backslash\alpha) \\ \\ \alpha\backslash\alpha = NIL,\ \bar{\alpha}\backslash\alpha = NIL \end{cases}$$

$$\begin{cases} T|U \triangleq \sum_i \mu_i(T_i|U) + \sum_j \upsilon_j(T|U_j) + \sum_{\mu_i=\overline{\upsilon_j}} \tau(T_i|U_j) \\ T|NIL = T \end{cases}$$

Here the port name τ is used to denote an unobservable **internal transition event** that has no co-port.

The restriction operator is used to suppress communication events that are of no interest. The composition operator is used to compose a description of global communication behavior of a system from the descriptions of the communication behaviors of the local processes constituting the system.

Finally, CCS includes an operator to relabel ports: []. If A is a behavior expression then $A[\alpha\backslash\alpha', \beta\backslash\beta']$ is the behavior expression obtained from A by substituting α' and β' in A for every occurrence of α and β respectively. Relabeling, however, seems to be better considered as a part of the metalanguage of CCS rather than part of the mechanisms to describe communication behaviors.

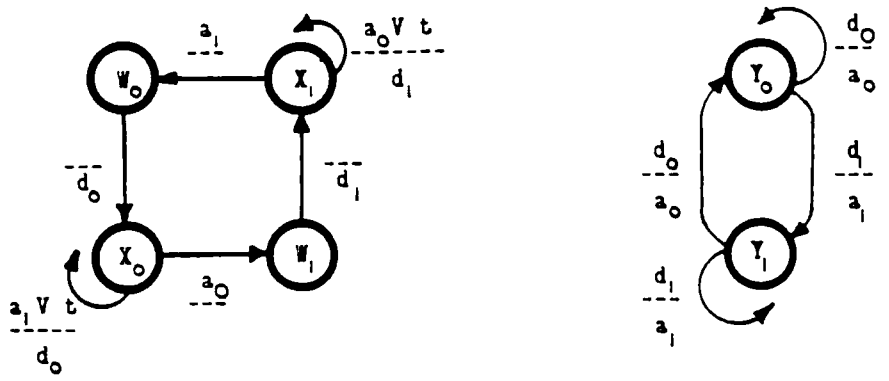
Using behavior expressions formed from operators and ports as above, one may describe arbitrary sequences of communication events in terms of algebraic equations defining their generation. For example, the behavior S described by the equation $S=\alpha(\overline{\beta}+\tau)S$ is one consisting of an infinite repetition of the event of offering a communication on port α followed by either an acceptance of a communication event on port $\overline{\beta}$ or an unobservable internal transition.

CCS proceeds to build an algebraic theory of objects passing along similar lines. Each port has a sort associated with it and may be used to bind values to variables of the sort belonging to the respective processes. The reader should consult [5] for further details concerning value passing. In this paper we only deal with simple protocols whose behavior may be described using behavior expressions involving no value passing (i.e., where the only significant aspect is the sequencing of communication events). This is suitable for connection establishment and simple transfer protocols. More complex protocols such as Stenning's transport protocol [6] require the value-passing expressibility.

Finally, CCS defines equivalence relations over behavior expression to describe several notions of observational similarities of behaviors. A calculus for handling (i.e., simplifying) behavior expressions to observationally congruent expressions is developed. This calculus may be used for automating validation of communication behaviors as demonstrated in [5].

1.2.2. MODELLING PROTOCOLS IN CCS

A protocol may be considered as a mechanism to generate sequences of communication events. Therefore, the behavior of a protocol may be described in terms of a behavior expression that generates the respective sequences. Consider, for example, the alternating-bit (AB) protocol described by the FSA of figure 2, below. The sender behavior S may be generated by following the transitions from the start state and associating sequentially composable behavior expressions with continuations of the FSA executions. The respective behavior expressions are provided in the figure below.



$S = d_0 X_0 d_1 X_1 S$
$X_0 = \bar{a}_0 + (\bar{a}_1 + \tau) d_0 X_0$
$X_1 = \bar{a}_1 + (\bar{a}_0 + \tau) d_1 X_1$

$R = \bar{d}_0 a_0 Y_0 Y_1$
$Y_0 = \bar{d}_1 a_1 + \bar{d}_0 a_0 Y_0$
$Y_1 = R + \bar{d}_1 a_1 Y_1$

Figure 2: The AB protocol

The sender loops between waiting for data at W_0, sending the data d_0, waiting for acknowledgement a_0 at X_0, waiting for next data at W_1, sending it as d_1 and waiting for acknowledgement a_1 at X_1. The receiver loops waiting for d_0 at Y_1, sending an acknowledgement a_0 when it arrives, waiting for d_1 at Y_0 and acknowledging it with a_1 upon arrival. The transitions marked by "t" correspond to a timeout condition. Timeout events might be represented by either an internal event τ or by describing explicitly the timer process and composing it with the sender.

Ignoring the behavior of the communication medium, the overall behavior of the AB protocol is thus described in terms of the composition S|R of the sender and receiver behaviors. The behavior of the medium may be described in terms of a behavior expression as follows. Replace each pair of similarly-named ports at the sender and receiver with a pair of ports for the medium. Substitute names in the sender/receiver so that communications only take place between sender-medium medium-receiver. For example consider the ports d_0 $\bar{d}_0$ in S and R respectively. The medium should be provided with respective ports $\bar{d}_0$ and d'_0. In the R behavior, all occurrences of $\bar{d}_0$ should be replaced with $\bar{d}'_0$. Finally, transmission over the medium of a d_0 event may be described by the behavior expression $\bar{d}_0(d'_0+\tau)$; this describes a succession where the medium accepts a d_0 event and follows by either offering a d'_0 event or an internal transition (corresponding to a message loss). Alternatively, one may imbed the description of the medium in the behavior of the sender and receiver (i.e., the receiver may decide to loose a message).

Let us pursue the AB example to develop partially the expressions for the combined behavior S|R (i.e., assuming that the medium is perfect).

$$S|R = \tau(X_0 d_1 X_1 S|a_0 Y_0 Y_1) + d_0(X_0 d_1 X_1 S|R) + \bar{d}_0(S|a_0 Y_0 Y_1)$$

The first term corresponds to an internal transition of the protocol machine corresponding to a delivery of the message d_0 from the sender to the receiver. The second term corresponds to an offering of a communication on the d_0 port (sending message d_0) of the sender followed

by an interleaving of the rest of the sender behavior with the receiver behavior. The third term has a similar interpretation. These terms may be further developed to pursue the generation of all possible execution sequences. This is best done in terms of a tree description of execution sequences (such synchronization trees describe a model of the CCS algebra [5]). In doing this one can easily see that it is possible for the protocol to generate infinite behaviors looping through the side-lobe loops of the sender an unbounded number of times. This bug in the protocol was first reported in [7].

2. PROTOCOL ANALYSIS

2.1. VALUATIONS

A key concept to the algebraization of protocol analysis relative to the algebraic model is that of a **valuation**. Let $\mathcal{C}$ denote the universal algebra of communication behaviors and let $\mathcal{A}$ be another arbitrary universal algebra. A *valuation* (homomorphism) from $\mathcal{C}$ to $\mathcal{A}$ is a mapping Φ of the elements of $\mathcal{C}$ to elements of $\mathcal{A}$ such that $\Phi(\xi.\psi)=\Phi(\xi)^*\Phi(\psi)$ and $\Phi(\xi+\psi)=\Phi(\xi)\circ\Phi(\psi)$ where * and $\circ$ are two binary operators over $\mathcal{A}$ and such that all equational identities (relative to observation congruence) defining $\mathcal{C}$ are mapped to equational identities on $\mathcal{A}$. The algebra $\mathcal{A}$ will be called **a valuation algebra** of $\mathcal{C}$.

Given a valuation algebra of $\mathcal{C}$ one may proceed and compute attributes of communication behaviors by mapping them to expressions in $\mathcal{A}$ and applying the calculus (identities) of $\mathcal{A}$ to evaluate these expressions. In what follows we describe briefly practical examples of the process.

2.2. AUTOMATED PERFORMANCE MODELLING

To automate performance modeling it is necessary to associate performance behavior with communication behavior expressions. Typically the measure of performance behavior is the distribution of time for a particular communication behavior to be completed. Let $\mathcal{A}$ denote the algebra of time distribution functions equiped with the convolution operation "*" and the sum of functions "+". A valuation from $\mathcal{C}$ to $\mathcal{A}$ may be defined by associating with each port (and co-port) a distribution function describing the delay associated with the generation (reception) of the respective event (i.e., from the time the event is offered until it is delivered to the respective co-port). The *NIL* behavior is naturally mapped to the delta distribution function assigning all mass to zero delay. The sequential composition operation "." is mapped to the convolution "*" (here we assume implicitly that duration of events form independent random variables). The non-deterministic choice "+" is mapped to a convex combination of the respective distributions. Specifically, the distribution associated with the behavior A+B is $F_A \circ F_B = pF_A(t)+(1-p)F_B$, where $F_A(t)$ and $F_B(t)$ are the distribution functions for the time to complete the behaviors A & B respectively and p is the probability that the behavior A will complete before the behavior B. The probability p is given by the expression $p=\int F_A(t)dF_B(t)$.

Using simple algebra of probability distributions, it is possible to compose recursively the delay distributions associated with any communication behavior. In particular, one can easily follow the above definition and associate delay distribution with the parallel composition operator and other derived operators. In computing the performance of a protocol one can follow the generation of the respective execution tree and associate delay distribution and probability of execution with each branch. This process may proceed directly from the recursive definition of the tree in terms of CCS expressions. The equations defining the tree map to equations among distribution functions. These equations may be solved and respective performance measures may be extracted.

An important fact to note is that the mapping defined above is not truely a valuation

with respect to CCS. CCS does not distinguish between different possible internal transitions and uses the same event τ to denote any such transition. The problem is that different internal transitions might involve different delay distributions. Similarly, observation congruence of CCS corresponds to functionally identical behaviors. However, functionally identical behaviors may not exhibit the same performance behavior. Accordingly, a more loose form of congruence for observation equivalence may be necessary. These variations of CCS are currently under study.

2.3. AUTOMATED REPRESENTATION OF PROTOCOLS

Suppose a user wish to interact (e.g., edit, analyze) with a protocol representation in terms of a FSA. What mechanisms would allow automatic translation between the FSA representation and the canonical algebraic representation? The problem may be re-stated as that of mapping expressions of the algebra C into a FSA representation and vice-versa. Consider the algebra A of FSA where the elements of the algebra are marked transition arrows and the operations of the algebra consist of connecting arrows through circular arrows (alternatively, one may use a transition matrix representation of a FSA). A mapping of C onto A may be defined as follows: with each port (co-port) symbol associate a transition with output (input) named with the respective symbol. NIL is mapped to a final state of the FSA. The sequential composition operation "." is mapped to sequential composition of transitions while the operator "+" is mapped to non-deterministic choice. The composition operator "|" is mapped to separate FSAs (for the automata corresponding to the left and the right hand sides). Figure 3, below describes the process. It is easy to verify that this mapping (conversion algorithm) constitutes a valuation (if one defines observation equivalence over FSA adequately, i.e., in terms of the respective equations for FSA). The conversion from FSA to a behavior expression is just as straight forward.

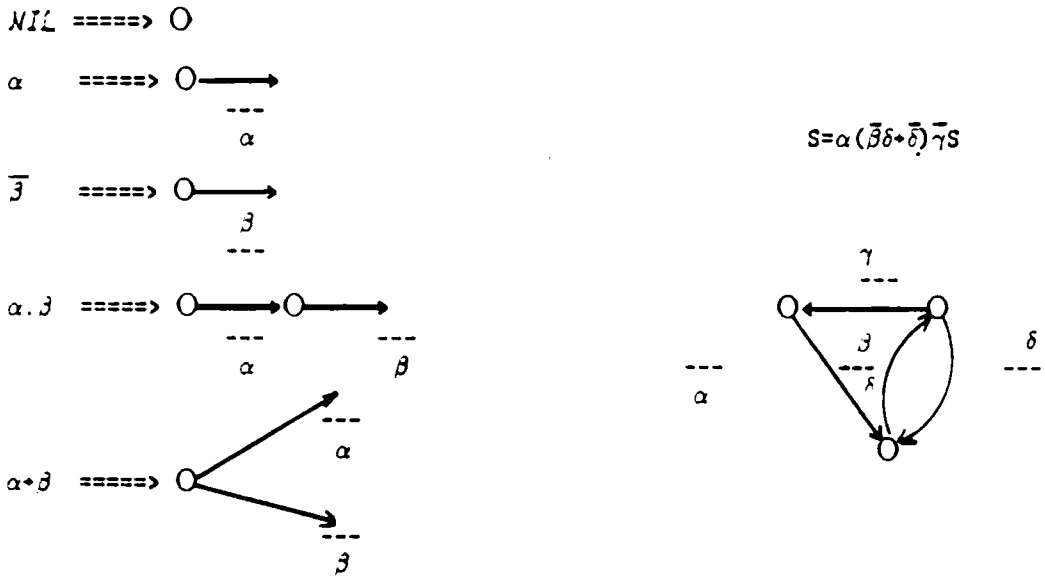
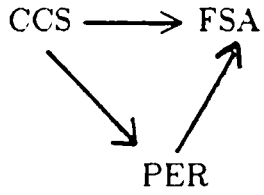


Figure 3: Valuation From CCS to FSA

Similarly, one may define a valuation of C to Petri-nets and other protocol representations.

Finally, given a performance valuation as described in the previous section and a valuation that maps CCS to FSA, it is possible to associate a derived performance valuation with the FSA by a commutative completion of the diagram below:



Here CCS, FSA and PER are the respective algebras. Note the practical significance to the representations of performance behavior to the user in terms of whatever protocol specification method that he may use.

3. NOTES ON AUTOMATED PROTOCOL VERIFICATION, SIMULATION AND TESTING

These subjects deserve more substantive treatment than the one for which this space is adequate. We restrict ourselves to a few comments on the role of valuations in providing the above tools. Milner [5] provides a number of examples of CCS use to verify correctness of communication behaviors. Verification is typically reduced to the proof of equivalence of a behavior to some behavior specifications (described in terms of CCS expressions). Birkhoff completeness theorem for equational theories implies that a proof of an equation (e.g., equivalence of protocol behavior to its specifications) may be obtained using substitutions of equivalent terms. Such proof process is reduced to a process of term rewritting and may be automated using the Knuth-Bendix procedure. Note that the use of equational logic in the specification of communication behaviors simplifies the proof process significantly, compared to the use of general first-order logic.

To generate a simulation of a protocol one should evaluate behavior expressions in terms of an algebra of routines that simulate them. Loosely speaking, each port (co-port) symbol should be mapped to invocation of a respective send (receive) routine. The *NIL* behavior should be mapped to a no-operation. The sequential composition "." should be mapped to sequential composition ";" of the respective invocations of routines. Finally, the non-deterministic composition "+" should be mapped to a choice (conditional on some boolean choice variable) between execution of either of the respective behaviors simulations. To generate an automatic test pattern of the simulator all that is necessary is to define the choice (i.e., by selecting appropriate choice variables) at each application of non-deterministic composition.

4. CONCLUSIONS AND CRITICISM

We have outlined briefly the main ideas underlying an automated algebraic-based protocol development environment. The algebraic approach offers the advantage of addressing complex problems through a uniform mechanism of calculi of respective behaviors. There are a few problem areas where substantive research is required. First, CCS has a number of limitations when it comes to addressing communication behaviors. For example, dynamic establishment and closing of links are not addressed, different modes of communications (e.g., buffered) are not addressed, the notions of equivalence of behaviors are not sufficiently refined to address some of the issues raised (e.g., performance valuation) and the algebraic foundations of the calculus are not completely set. Second, the valuation mechanisms discussed above need to be formalized and respective notions of equivalence explored. Third, the algorithms for performing the valuations need to be experimented with extensively to address practical problems.

References

[1] G.Holzmann.
Algebraic Validation Methods-A Comparison of Three Techniques.
Second International Workshop on Protocol Specification, Testing, and Verification , 1982.

[2] G.Holzmann.
A Theory For Protocol Validation.
IEEE Tranactions on Computers , August, 1982.

[3] G.Holzmann.
The PANDORA System : An Interactive System for the Design of Data Communication Protocols.
Delft University of Technology, Report No.39 , August, 1982.

[4] G. Milne.
A Mathematical Model of Concurrent Computation, Phd Thesis.
Technical Report, Department of Computer Science, University of Edinburgh, 1979.

[5] R. Milner.
A Calculus of Communicating Systems.
Springer Verlag, 1980.

[6] N.Stenning.
A Data Transfer Protocol.
Computer Networks (1):99-110, 1976.

[7] Y. Yemini & J. Kurose.
Towards the Unification of the Functional and Performance Analysis of Protocols, or is the Alternating-Bit Protocol Really Correct?
Second International Workshop on Protocol Specification, Testing, and Verification , 1982.