PROTOCOL SPECIFICATION AND VERIFICATION WORK AT USC/ISI
Summary Report
August 1982

Carl Sunshine

1. INTRODUCTION

For the past three years, several projects at USC/ISI, particularly the Internet Concepts Project, have been studying formal protocol specification and verification. This work is now coming to an end, and we would like to present here a summary of results obtained. More complete information is available in a research report [1]. A complete list of references to earlier outputs is also included here.

Section 2 presents a brief outline of the major activites undertaken and their outputs. Sections 3-5 summarize the results obtained in each major area. Conclusions are discussed in Section 6.

2. MAJOR ACTIVITIES

Our work has been divided into three major areas: survey, in-depth studies, and standards activities.

As might be expected, the work began in the survey area in 1979, building on a study (jointly sponsored by ARPA and NBS) completed while the author was still at the Rand Corp [2]. Extensions of this work resulted in two journal publications [3,4]. A later and entirely new survey was completed in 1981 [5], and again the results were widely distributed in conferences, journals, and books [6,7,8,9,10]. As part of this later survey effort, major papers in the field were selected and collected into a reprint volume, published by Artech House [11].

We have also helped to organize a number of workshops bringing together researchers in the field to discuss current state of the art, problems, and promising future directions. The first was sponsored by ARPA in 1979 [12,13]. The next was hosted by ISI in 1980 [14]. A major effort was made to broaden participation in 1981 with a workshop sponsored by IFIP WG 6.1 and organized jointly with the British National Physical Laboratory [32]. The success of this effort has led to a series of annual IFIP workshops on this topic, organized and hosted by us in 1982 with a published proceedings [15].

We have also edited or organized several special sections on formal modeling of protocols in journals [4,16,17].

Our in-depth studies have focused on the appplication of general software engineering techniques to protocols, and particularly on experiments with automated tools for verification. The majority of our work has been done with AFFIRM, a system based on abstract data types with axiomatic specifications developed at ISI. Experience with several protocols and with various aspects of the methodology (safety, liveness, multi-level specifications) have been reported in technical reports [18,19,20], a conference [21], a PhD thesis [22], and a journal [18].

To test other approaches, we have worked with three other automated verification systems more recently. These were the Ina Jo system [Ina Jo is a registered trademark of the System Development Corp.] based on abstract machine notions, the Gypsy system based on a buffer history approach, and state deltas which employ symbolic execution. Some preliminary results in these were reported in [23,24,25], while the major results are in [26,1].

Part of our work has involved participation in standards activities in order to promote wider use of the more rigorous specification methods we and others have been developing. This work has included development of specification standards within ISO SC 16 (for use with Open Systems protocols and services), collaboration with the System Development Corp. in their development of new IP, TCP, and Telnet protocol specifications [27,28,29], and comments on the transport protocol being developed for the National Bureau of Standards by Bolt Beranek and Newman [30,31].

3. SURVEY WORK

Our early work in this area focused on clarifying the meaning of specification and verification in the context of communication protocols. This included developing the now widely accepted notion that complete protocol specifications must include separate definitions of (1) the service to be provided to the protocol's users, (2) the mechanisms used within the protocol itself to accomplish those services, and (3) the services requried from the lower layer(s) used by the protocol [2,3,4].

Ideally, the protocol itself (item 2) should be specified in an abstract fashion so that its "design" may be verified (shown to provide the desired service), while still leaving as much freedom as possible for implementation. Of course, this means that any implementation must also be shown to properly implement the abstract design (in the more conventional program verification sense).

Protcol specification methods have traditionally developed from either a state machine or program language point of view. State machine models

view protocols as accepting inputs and producing outputs based on an explicit current state which serves to summarize the relevant previous history of the system. These models include basic finite state automata (FSA), their extension into abstract machines (with additional state variables, not necessarily bounded), abstract data types, and graph models such as Petri nets and their extensions or the UCLA graph model.

The simpler forms of these have a variety of powerful analysis tools that are avialable such as state exploration, derivation of invariants by linear algebra, or reduction methods. Unfortunately, they also lack the expressive power to completely model protocols of real world complexity. The extended methods are more successful as specification tools, but make the analysis problem more difficult.

Program language models view protocols as just another kind of algorithm that may be specified using a choice of several high level programming languages. Verification is then possible by developing appropriate assertions that reflect desired properties of the protocol, and proving them by conventional assertion proof methods. Since protocols involve interaction among several concurrent modules, techniques that can handle parallel execution are required.

Traditional program proof methods have focused more on safety properties (only good things can happen) than liveness (good things will eventually happen). More recent work to incorporate temporal logic into these methods facilitates the direct expression and proof of liveness properties.

A third group of methods has developed in an effort to eliminate the appearance of explicit state information in specifications, and to focus more directly on the input/output behavior of the protocol. Formal languages and sequencing expressions are methods to specify the allowed sequences of events directly. Of course these have well known relations to FSA that recognize or generate the same language. Buffer or event histories are another technique used in several systems to facilitate direct expression of I/O behavior.

Of these different approaches, state machine methods seem to be most widely used due to their wider understandability and the existence of automated tools for their manipulation. Surprisingly useful results have been obtained from the simpler models such as FSA and Petri nets, with applications to such protocols as X.25, X.21, and local net token passing documented in the literature [4,11,15].

The greater expressive power of abstract machines makes them popular for more complete or complex protocol specifications. Such methods have been proposed within ISO, CCITT, NBS, and DoD as favored approaches to specifying their protocols.

Other methods, particularly temporal logic and sequencing expressions, are receiving much attention in the research community for their ability to remedy particular shortcomings, but are still in earlier stages of development.

4. IN-DEPTH STUDIES

Our in-depth studies have focused on the application of existing automated verification systems to communication protocols. Our main goal has been to assess the current state of the art in automated verification and determine the potential for more widespread use of these techniques in protocol development.

A common set of example protocols were emplyed with each system. These were the well-known Alternating Bit protocol (in a form including arbitrary message loss and retransmission), and the "three-way handshake" connection establishment protocol from the DARPA TCP. The former served to test capabilites of the systems to handle "data transfer" functions, while the latter served to test "control" functions. Since these protocols are quite mature, our results were mainly methodological, identifying strengths and shortcomings of each system, rather than uncovering protocol bugs (although we did discover an obscure error in TCP).

Our major interest throughout this work has been on design verification rather than code or implementation verification. Hence we have attempted to develop "abstract" specifications for the services and entities of a given protocol layer, and to prove that the combined operation of the entitities plus the lower layer service has certain properties, or meets some service specification. We have been less interested in the (more traditional) problem of verifying that a specific program or code correctly implements a protocol entity.

Affirm

Our deepest study has been of the Affirm system. Affirm includes a specification language based on the theory of abstract data types, a verification condition (VC) generator, and an interactive theorem prover for proving properties of specifications or of programs. There is also a library of already specified types (e.g., queues, sets) and their properties that may be used in building new types.

Thanks in part to collaboration with the Program Verification Project at ISI (developers of Affirm), our experience with Affirm has been quite successful. We were able to model abstract machines in Affirm easily, to perform multilevel proofs (that a protocol meets its service), and to prove some other significant properties of several protocols, including progress properties and finding an obscure bug in TCP [18,19,20].

Ina Jo

The other three systems were chosen to explore some particular features, and received less attention. Ina Jo is specifically intended to model hierarchies of abstract machines, with mappings from higher to lower layers defined. The system also includes a specification language, a VC generator, and an interactive theorem prover.

Abstract machine type protocol specifications were very easy to write in Ina Jo, although absence of data types and sometimes cryptic syntax were shortcomings. But the hierarchical proofs we had hoped to perform proved impossible due to limitations in Ina Jo's ability to handle nondeterministic mappings between layers (needed to model protocols with message loss). The theorem prover was also not as convenient or flexible, especially in the handling of lemmas, driving us to carry some proofs into the Affirm system where they could be developed more easily, and then using the results to continue in Ina Jo.

Gypsy

The primary feature of Gypsy that interested us was its reliance on buffer histories rather than state transitions for specifying process behavior. The specification language focuses on the external or input/output behavior of processes, with constructs to refer to the history of messages read and written by each process in each buffer.

Gypsy's buffer history orientation proved to be a mixed blessing. When properties to be specified directly concerned relations between sequences of messages, Gypsy's buffer history techniques were quite powerful and convenient. The Alternating Bit protocol falls in this category, and was essentially proved by transitivity of subsequence relationships.

For the three-way handshake, designers clearly think in terms of the state of an entity when defining its behavior, and it was difficult to construct meaningful external specifications of the entities.

Instead the proof was essentially carried out at the code level where reference to internal state variables and a conventional abstract machine could be used. In this case, the Gypsy methodology seemed to get in the way.

State Deltas

The Concurrent State Deltas system was an outgrowth of the Microcode Verification Project at ISI and was still in an early stage of development. The basic unit of specification is a "state delta" stating that if a certain precondition is ever met, then eventually a given postcondition will become true. A set of CSDs for several processors is

symbolically executed from a given initial state to determine whether a desired final state will necessarily be reached.

Unlike the previous proof systems that are interactive, the symbolic execution is completely automatic and requires no user aid. In practice, however, system resources are quickly exhausted for specifications of any complexity, and the user must provide some appropriate intermediate goals to force pruning of the proof tree. Thus proofs of the simplest cases of the three way handshake (no loss or retransmission) were easiest with CSDs since they were completed totally automatically, but it was difficult to see how to extend these to more complex cases.

The CSD system is also the only one to include specific time bounds. In simple examples (e.g., specifying that retransmission will not occur unless no reply will arrive) time bounds effectively simplified the proof, but including the time information makes the symbolic execution more complex and hence was not always practical. In more general protocols, such simple time constraints cannot be assumed anyway.

5. STANDARDS

Protocol development is now underway in many forums, and each of these must select some method for the specification of protocols being developed. Throughout this research period we have attempted "technology transfer" by participating in several of these outside efforts.

The System Development Corp. (SDC) has been under contract to the Defense Communications Engineering Center to produce more rigorous specifications of the DARPA IP, TCP, and Telnet protocols. We participated in the development of their specification methodology which is based on abstract machine notions [27], and helped review the application of this methodology to specific protocol and service specifications [28,29].

Bolt Beranek and Newman (BBN) has been under contract to the National Bureau of Standards (NBS) to develop a protocol specification technique and several specific protocols for Federal standards. We participated in the review of the general methodology, and its application to the transport layer protocols and services in particular [30,31].

The International Standards Organization (ISO) SC 16 has developed the now widely known Open Systems Interconnection Architecture, and is now in the process of specifying protocols and services for the various layers. A subgroup on Formal Definition Techniques has been in operation within WG 1 for about two years, and we have participated in their development of guidelines to be used by the other groups actually developing protocols. These include a specification language based on

abstract machine notions much like those in use by BBN, SDC, and others. The linear form has Pascal language syntax in many places, while a graphical form based on the CCITT SDL language is likely to be adopted.

6. CONCLUSIONS

In the area of specification, it is clear that abstract machine models are realtively mature, and are in wide use by more ambitious specification projects. In addition to their benefit in simply defining a protocol, there are also a number of automated tools that can check for correct syntax, completeness, and partially validate the interactions of such specifications, and produce partial implementations. This technology appears to be ready for more widespread use, and indeed has already been applied in the work on DoD protocol standards mentioned above.

Shortcomings of abstract machine methods in the areas of insufficient abstraction and handling progress as well as safety are being tackled by several methods that are in earlier stages of development, such as sequencing expressions and temporal logic. We expect this work will have to continue for several years before the results are ready for more widespread use.

In the area of verification, based on our experiments with four systems, we can report that none of the systems has all the features desired, and none of them is ready for routine and/or mechanical application to real-world protocols. Affirm was by far the more polished system, but even there the proof process remained very tedious.

Surprisingly (to us at least), the major contribution of automated verification systems does NOT seem to be in reducing the amount of human ingenuity required to accomplish a proof. Rather, they do seem to increase the certainty of correctness. If the user has the ingenuity to formulate the problem in a tractable fashion, and the stamina to follow through all the tedium, the formally verified conclusion does seem to be far more reliably correct than with hand proofs.

Beyond user interfaces and robustness, which certainly need attention in all but Affirm, each system is lacking some key abilities such as composition of independent modules, handling progress properties, redoing portions of proofs, supporting hierarchical verification, or more automatic proof discovery. Several years' work and a second generation of verification systems incorporating the best features of all will be necessary before formal verification of realistic protocols can be accomplished by other than expert researchers with very large investments of time.

REFERENCES

[1] Sunshine, C., "Automated Protocol Verification", USC Information Sciences Inst., Research Report, September 1982.

[2] Sunshine, C., "Formal Methods for Communciation Protocol Specification and Verification," Rand Corp., N-1429-ARPA/NBS, November 1979.

[3] Sunshine, Carl A., "Formal Techniques for Protocol Specification and Verification", Computer 12, 9, September 1979.

[4] Bochmann, G., and C. Sunshine, "Formal Methods in Communication Protocol Design", IEEE Trans. on Communication, COM-28, 4, April 1980.

[5] Sunshine, C., "Formal Modeling of Communication Protocols", USC Information Sciences Inst., RR-81-89, March 1981.

[6] Sunshine, C., "Formal Modeling of Communication Protocols", in Proc. Conference on Communication in Distributed Data Processing Systems, Technical University Berlin, January 1981.

[7] Sunshine, C., "Formal Protocol Specification", State of the Art Report on Network Architectures, C. Solomonides, editor, Pergamon Infotech, 1982.

[8] Sunshine, C., "Local Network Protocols", Proceedings of the Local Networks and Distributed Office Systems Conference, London, England, Online Conferences Ltd., May 1981.

[9] Sunshine, C., "Protocol Verification", talk presented at Nordic Universities Networking Conference (Nordunet), Copenhagen, Denmark, June 1981.

[10] Sunshine, C., Formal Modeling of Communication Protocols", Computer Networks and Simulation II, S. Schoemaker, editor, North-Holland Publishing, September 1982.

[11] Sunshine, C., editor, "Communication Protocol Modeling", Artech House, Dedham, Massachusetts, 1981.

[12] Sunshine, C., "The Meaning of Protocol Specification and Verification", ARPA Protocol Verification Workshop, March 1979.

[13] Postel, J., "Issues in Protocol Verification", ARPA Protocol Verification Workshop, March 1979.

[14] Sunshine, C., "Problem Areas in Protocol Specification and Verification", ISI Internal Memo, July 1980.

[15] Sunshine, C., editor, Proc. 2nd Int. Workshop on Protocol Specification, Testing, and Verification, North-Holland Publishing, May 1982.

[16] Sunshine, C., editor, special issue on Protocol Specification, Testing, and Verification, to appear in Computer Networks, early 1983.

[17] Sunshine, C., editor, special issue on Protocol Specification, Testing, and Verification, to appear in IEEE Trans. on Communications, December 1982.

[18] Thompson, D., C. Sunshine, R. Erickson, S. Gerhart, D. Schwabe, "Specification and Verification of Communication Protocols in AFFIRM Using State Transition Models", USC Information Sciences Institute, RR-81-88, March 1981. Also to appear in IEEE Transactions on Software Engneering, September 1982.

[19] Schwabe, D., "Formal Specification and Verification of a Connection-Establishment Protocol", USC Information Sciences Institute, RR-81-91, April 1981.

[20] Berthomieu, B., "Algebraic Specification of Communication Protocols", USC Information Sciences Institute, RR-81-98, December 1981.

[21] Schwabe, D., "Formal Specification and Verification of a Connection Establishment Protocol", Proc. 7th Data Communications Symp., Mexico City, Mexico, IEEE, October 1981.

[22] Schwabe, D., Formal Techniques for the Specification and Verification of Protocols, Report No. CSD-810401, UCLA, (PhD Thesis), April 1981.

[23] Sunshine, C., "The Restaurant Example Revisited," USC Information Sciences Institute, Affirm Memo 52, September 1981.

[24] Sunshine, C., "Experience with Four Automated Verification Systems", Proc. 2nd Int. Workshop on Protocol Specification, Testing, and Verification, C. Sunshine, editor, North-Holland Publishing, May 1982.

[25] Overman, W., and S. Crocker, "Verification of Concurrent Systems: Function and Timing", Proc. 2nd Int. Workshop on Protocol Specification, Testing, and Verification, C. Sunshine, editor, North-Holland Publishing, May 1982.

[26] Overman, W., "Verification of Concurrent Systems: Function and Timing", PhD thesis, UCLA, 1981.

[27] Simon, G.A., "DCEC Protocols Standardization Program/ Protocol Specification Report," System Development Corp., TM-7038/204/00, July 1981.

[28] Bernstein, M., "DCEC Protocols Standardization Program: Proposed DoD Internet Protocol Standard", Systems Development Corp., TM-7038/205/01, December 1981.

[29] Bernstein, M., "DCEC Protocols Standardization Program: Proposed DoD Transmission Control Protocol Standard", Systems Development Corp., TM-7038/207/01, December 1981.

[30] Sunshine, C., "Comments on the NBS Transport Protocol Proposal", USC Information Sciences Inst., IEN-195, August 1981.

[31] Sunshine, C., "Comments on 'Formal Service Specification of the Transport Protocol Services'" (April 1982 draft by Bolt Beranek and Newman for the National Bureau of Standards), USC/ISI memo, June 1982.

[32] Sunshine, C., "Protocol Workshop Report", Computer Communication Review, Special Interest Group on Data Communication, ACM, July 1981. Also in Computer Networks, V.5, N.4, July 1981.