4V−6

Evaluation of LOTOS Execution System with a OSI TP protocol specification

Shingo NOMURA†, Takashi TAKIZUKA†, Toru HASEGAWA† and Ron GREVE††
†KDD R&D Laboratories ††University of Twente

1. Introduction

LOTOS[1] is a Formal Description Technique standardized by ISO. It is useful for specifying distributed systems, such as OSI. Actually some of the OSI protocols have already been specified in LOTOS. When implementing these OSI protocols, one can expect that the use of tools will reduce the implementation costs. We have developed a LOTOS Execution System[2] which can derive an implementation from a specification written in LOTOS. To verify that the system is applicable to build an actual communication system, we have evaluated the system by applying it to a LOTOS specification of part of the OSI Transaction Processing (TP) protocol. It is proved that the system is able to generate a correct program for this OSI TP protocol, and that the implementation costs are smaller than those in case of manual programming. This paper describes the results of our implementation experiment in detail.

2. LOTOS Execution System

Our system consists of a translator from LOTOS specifications to C programs, and a scheduler library used for LOTOS process execution. Since the translator transforms a LOTOS behaviour expression into a sequence of scheduler library calls, the generated program closely resembles the architecture of the original LOTOS specification. The scheduler library executes LOTOS processes in parallel, and provides LOTOS basic functions such as synchronization, choice and disabling. At the moment, the scheduler library is designed for execution on the VAX/VMS operating system.

A restriction of our system is that there is almost no support for data type compilation, although data type descriptions, defining sorts and operations, are included in the specification.

3. Implementation

In order to evaluate our system, we translated a LOTOS specification of part of the OSI TP protocol. After translation, the generated code was complemented with manually written code in order to derive a complete implementation. Manual coding included both the coding of the data type operations and the interface with the environment.

3.1 OSI TP protocol and its LOTOS specification

The OSI TP protocol[3] is one of the application layer protocols of the OSI Reference Model. It is used to support transaction processing in a distributed environment, and allows communication between more than two partners.

The LOTOS specification[4] is developed within Lotosphere, one of the European ESPRIT II projects. Its structure conforms to the ISO Application Layer Structure and its size is 3,870 lines: 2,630 lines data type part and 1,240 lines behaviour part. Basically, the following Functional Units (FUs) of OSI TP are specified;

- Kernel FU (without error and abort services),
- Commit FU (without rollback and recovery),
- Shared control FU.

Approximately, the protocol described in this specification corresponds to a state machine with 19 states and 23 input primitives.

3.2 Translation

Our translator requires the user to modify the specification, before it can be translated. Since the translator does not handle any axioms for data type operations, all axiom descriptions have to be removed from the specification. Also some operation names have to be redefined in order to resolve naming ambiguities. In our case these modifications reduced the specification from 3,870 to 1,611 lines: 381 lines data type part and 1,240 lines behaviour part.

As a result of translating the specification, a C program is generated which consists of 4,407 lines (see Table 1).

Table 1 Implementation result

input specification		1,611 lines
generated code		4,407 lines
manual coding	operations	2,582 lines
	interface	723 lines

3.3 Manual coding

(1) Coding of data type operations

For each data type operation, the user has to implement the body of a function whose prototype is generated by the translator from its LOTOS

LOTOS実行系を用いたOSIトランザクションプロトコルの実装実験
野村 真吾, 瀧塚 孝志, 長谷川 亨, Ron GREVE
国際電信電話株式会社 研究所, University of Twente

description. Our system uses so called descriptors (C structures) for representing data values [2]. These descriptors are a key feature for the user to implement operations.

We needed 2,582 lines of C code (see Table 1) for implementing 203 operations. Although the size of the code is very large, its complexity is relatively low. The operations can be thought of as grouped into 39 categories. Operations in each category have a similar structure. For example, there are 69 operations to identify the various service primitives (e.g. IsTP__DATAreq) and their structure is almost the same.

(2) Coding of interface

The implementation has to run as a separate process on VAX/VMS and has to communicate with its environment via mail boxes. In LOTOS, the specification communicates with its environment via external gates. For each of these external gates, our system generates a prototype function, of which the body has to be coded by the user. The 'gate functions' read from and write to mail boxes and convert service primitives into an internal representation using descriptors and vice versa. In order to handle the mail box interface easily, primitive routines for using a mail box have been prepared in our library.

The total number of lines necessary to describe the interface was 723 lines (see Table 1).

4. Evaluations

(1) Program execution experiment

In order to check the correct working of our TP implementation (TP Entity), we built a simple TP-user program for sending service primitives to and receiving service primitives from the TP Entity. Another program called lower layer simulator was built for conveying the output of one TP Entity to another (see Fig. 1).

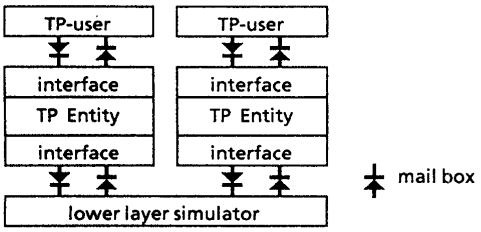

Fig. 1 Execution test

The implementation is evaluated using VAX station 3100. In the configuration shown in Fig. 1, we executed the TP service primitive sequence as listed below :

TP-BEGIN-DIALOGUEreq/ind, TP-DATAreq/ind,
TP-DEFERRED-END-DIALOGUEreq/ind,
TP-COMMITreq/ind,
TP-CONTINUE-COMMITreq,
TP-COMMIT-RESULTind,
TP-DONEreq, and TP-COMMIT-COMPLETEind.

The execution time was 750 ms and 144 LOTOS processes were created.

Since 1,400 LOTOS processes can exist simultaneously on our system[2] and 86 processes are created for each transaction branch, 14 transaction branches can be processed at the same time.

(2) Implementation costs

Table 2 shows the costs of the implementation experiment. This experiment was performed by a novice user of our system, who is familiar with LOTOS, the OSI TP protocol and its specification. In case the user knows our system well, costs of both manual coding and debugging are likely to be decreased.

It is thus possible for a novice user of our system to implement the TP protocol within 32 days. Not so much effort was required from the user. Although the user needed to write code for data type operations and for the interface with the environment, this task was relatively small. Also the similarity between the architecture of the generated code and that of the specification helped the user to debug the code.

Table 2 Implementation costs

training		5 days
modification of specification		4 days
manual coding	operations	8 days
	interface	5 days
debugging		10 days

5. Conclusion

This paper presented the result of the evaluation of our LOTOS Execution System. The result shows that the system can generate a program for an actual communication system like OSI TP, and that this program can execute correctly after some additions to the code.

The implementation costs were small enough to conclude that our system is suitable for prototyping. For a novice user of our system it is possible to implement the OSI TP protocol within 32 days.

The authors would like to thank Dr. K. Ono, Director, Dr. Y. Urano, Deputy Director of KDD R&D Labs, Dr. K. Suzuki, Manager of OSI Systems Group and Mr. K. Konishi, senior research engineer of KDD R&D Labs, for their helpful suggestions.

References

[1]: ISO 8807, "LOTOS - A formal description technique based on the temporal ordering of observational behaviour", Feb. 1989.

[2]: S. Nomura, T. Takizuka, T. Hasegawa, "Preliminary Evaluation of LOTOS Translation Method", Conf. of IPSJ, 6L-09, Mar. 1992.(in Japanese)

[3]: DIS 10026-2, "Information technology - Open Systems Interconnection - Distributed transaction processing - Part 3: Protocol specification", Mar. 1990.

[4]: R. Greve, I. Widya, "TP Protocol Version 2.0: a two-way synchronization version", Lotosphere project ESPRIT 2304 Lo/WP3/T3.1/UT/N0016/V3, UT, PTT Research, 1992.