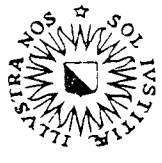
A FORMAL SYSTEM FOR A TELECOMMUNICATION LANGUAGE

Ron Koymans
Jan Vytopil
Willem P. de Roever

RUU-CS-83-5
Februari 1983



Rijksuniversiteit Utrecht

Vakgroep informatica

Princetonplein 5
Postbus 80.002
3508 TA Utrecht
Telefoon 030–53 1454
The Netherlands

A FORMAL SYSTEM FOR A TELECOMMUNICATION LANGUAGE

Ron Koymans
Jan Vytopil
Willem P. de Roever

Technical Report RUU-CS-83-5
Februari 1983

Department of Computer Science
University of Utrecht
P.O. Box 80.002
3508 TA Utrecht
the Netherlands

A FORMAL SYSTEM FOR A TELECOMMUNICATION LANGUAGE

a case study in proofs about real-time programming and asynchronous message passing

by

Ron Koymans & Jan Vytopil
Philips' Telecommunicatie Industrie B.V.,
PTI Hilversum
the Netherlands

Willem P. de Roever
Department of Computer Science,
University of Utrecht
the Netherlands

Part 1. Global motivation section

In recent years an acute need has arisen for a methodology of real-time processes, as occurring in telecommunication systems or process control in general. This paper is an attempt to contribute to the development of such a methodology.

The paper concerns the description of real-time processes and their asynchronous communication by means of signal (message) exchanges in a real-time language, CHILL/D-16. For the characterization of these processes and their communication we develop a real-time temporal logic with two additional operators. By means of these two operators we believe to be able to express the real-time and asynchronous communication aspects in a most natural way. As a consequence, by using a proof system for reasoning

about these operators, one can reason about real-time processes and their communication, i.e. one can prove that a program meets its real-time specifications.

Of course, a complete real-time methodology is still some time off, but we believe that our method applies to real-time processes in general thus providing a possible direction in which such a methodology can develop.

The language we use to study real-time processes and their asynchronous communication is CHILL/D-16, a real-time variant of CHILL (CCITT HIgh Level Language, see [CHILL]), the recommended language of CCITT (Comité Consultatif Internationale de Télégraphique et Téléphonique). CHILL is an interesting subject of study as it is used by many different industries, such as Siemens, NTT(Nippon Telephone and Telegraph), ITT(International Telephone and Telegraph) and PTI(Philips' Telecommunicatie Industrie B.V.) for telecommunication purposes. CHILL/D-16 is a dialect of CHILL used at PTI.

In the area of distributed processing, recent research focusses on languages such as CSP (see [HOARE]) and ADA (see [ADA]), for which safety and lifeness properties etc. have been extensively studied (see e.g. [KUIPER & DE ROEVER], [MANNA & PNUELI], [PNUELI & DE ROEVER], [GERTH]). These languages, however, are based on the rendezvous mechanism, which requires synchronous communication. Furthermore, the research has until now left the real-time aspect almost

completely out of account, although a suggestion has been made for the ADA-case at the end of [PNUELI & DE ROEVER] and a first step towards a proof technique for real-time properties of concurrent programs has been made in [BERNSTEIN & HARTER].

Summarizing, this paper has a twofold purpose:

a) to give a direction of development for a methodology of real-time processes in general,

b) to provide a formal way of reasoning about the communication part of CHILL/D-16 in particular.

Part 2. Technical section

We take the following real-time aspects into account:

I. a time-out mechanism:

in CHILL/D-16 (as in ADA) a timer can be present with the purpose of restricting the period of waiting for a message to some fixed amount of time. Usual temporal logics can not express this as they consider time qualitative and not quantitative. The need for a time-out possibility is obvious when developing telecommunication systems as in a certain amount of time there must always be some progression in the system.

II. a medium real-time property:

we impose the condition that the communication medium is not 'too lazy', that is the medium may not be too long idle as long as there are outstanding signals to be transmitted (again this is an obvious requirement in a telecommunication environment); the formal expression of this property is given furtheron.

The following technical complication concerning 1 occurs: if program control reaches (an occurrence of) an action which can receive a signal (a RECEIVE CASE-action in CHILL/D-16), it must be expressed that a timer will run off within a fixed amount of time. We take care of this by the introduction of a special 'timerstart'-location (this notion is explained and refined below) for each RECEIVE CASE-action. Similar cases of having to introduce 'locations' which do not occur actually in a program in order to reason (prove properties) about that program occur for Brinch Hansen's DP (see [BRINCH HANSEN], [ASTESIANO & ZUCCA]) and ADA (see [GERTH]).

Apart from the real-time property II above, we impose in our formulation furthermore the following conditions for the communication medium: it may create no ghost messages (neither original nor copies of real signals) and it must be fair relative to the transmission of signals (see below).

Furthermore, the medium need not be error-free (but fairness, see below, imposes that it can not be too faulty either).

Fairness of the medium must be imposed indeed, as the properties described hitherto do not guarantee at all that anything will be received. The fairness (liveness) condition imposed on the medium then is classical: if infinitely often a signal of a collection of signals is sent, then at least one signal of the collection will arrive at its destination. (This implies that even infinitely many signals of the collection will arrive at their destination.) Furthermore we decided to impose another fairness condition, namely fairness relative to the receipt of signals: in CHILL/D-16 (as in ADA) it is possible that more than one signal is choosable for receipt at a time and we demand that the choice mechanism is fair (in ADA this fairness is explicitely defined in the language by queues, but this was shown to be superfluous in [PNUELI & DE ROEVER]).

To illustrate the two fairness conditions, we now give a CHILL/D-16 program fragment (for better readability P1 and P2 are placed next to each other):

```
SIGNAL s1 TO p2,
       s2;

p1: PROCESS();
    ...;
    SEND s1;
    DO FOR EVER;
      RECEIVE CASE
        (s2): ...;
              SEND s1;
      ELSE ...;
           SEND s1;
      ESAC;
    OD;
END p1;
```

```
p2: PROCESS() SINGLE;
    ...;
    DO FOR EVER;
      RECEIVE CASE SET source;
      (s1): ...;
            SEND s2 TO source;
      ESAC;
    OD;
END p2;
```

```
p3: PROCESS();
    ...;
    DO FOR EVER;
      RECEIVE CASE
        (s2): ...;
      ELSE ...;
           SEND s1;
      ESAC;
    OD;
END p3;
```

We first explain the syntax of the fragment. In CHILL/D-16 signals can be sent by a SEND-action and received by a RECEIVE CASE-action. Signals are declared in a SIGNAL-definition (see the first two lines above). p1, p2 and p3 are the names of the three PROCESS-definitions of the fragment. They all have an empty parameterlist denoted by (). A PROCESS-definition can have several active process instances at a time. ... stands everywhere in this fragment for an arbritrary sequence of terminating CHILL/D-16 actions.

To each SEND-action a destination, a unique process instance, is associated. This destination can be specified directly in the SEND-action by means of the TO-option (see p2) or alternatively there must be specified a PROCESS-definition name (by means of another TO-option) in the SIGNAL-definition belonging to the SEND-action (see the definition of s1). To guarantee uniqueness of the destination in the latter case this PROCESS-definition must have the SINGLE-attribute (see p2) which means that this PROCESS-definition can have only one active process instance at a time. The DO FOR EVER ... OD; construct has an obvious meaning. Each RECEIVE CASE-action has a number of (in the fragment always one) signal names enclosed in parentheses and followed by a colon and some actions. We say that a signal is receivable for a RECEIVE CASE-action if its signal name is one of the signal names listed in the RECEIVE CASE-action. If program control is at the beginning of a RECEIVE CASE-action there are two possibilities: at least one receivable signal for the RECEIVE CASE-action has arrived or no such signal has arrived. In the first case there will be received a unique signal (of the receivable signals for the RECEIVE CASE-action) and control transfers to the actions specified in the RECEIVE CASE-action after the signal name of the received signal. In the second case (no signal arrived) the undertaken action depends on the presence of an ELSE-part in the RECEIVE CASE-action. If no ELSE-part is present control will stay at the RECEIVE CASE-action until (possibly never!) a receivable signal for

the RECEIVE CASE-action arrives. If on the other hand an ELSE-part is present program control will be transferred to the actions in the ELSE-part. Real-time is introduced in CHILL/D-16 by allowing an additional AFTER-part in the ELSE-part which specifies how long (measured in some real-time unit) there will be waited for the arrival of a receivable signal for the RECEIVE CASE-action before control is transferred to the further actions in the ELSE-part. A further option of the RECEIVE CASE-action is the identification of the sender of a signal. This is done by the SET-option (see p2). It is used here to send a signal back to the sender.

Now we can illustrate by this fragment the two fairness conditions above. In our temporal logic system (of which a version for CHILL/D-16 is already available and in use internally in a development group of PTI) we can deduce the following for this fragment:

1. s1 (both in p1 as in p3) is infinitely often sent,
2. s1 (both coming from p1 as from p3) will infinitely often arrive at its destination, c.q. p2 (this follows from 1. and the fairness condition for the medium),
3. at the RECEIVE CASE-action of p2 infinitely often a signal is received (this follows from 2. and the fact that control will be infinitely often at the RECEIVE CASE-action of p2),

4. s2 will be sent infinitely often (this follows from 3.),

5. at the RECEIVE CASE-action of P2 infinitely often s1 coming from P1 and infinitely often s1 coming from P3 will be received (this follows from 2. and 3. and the fairness condition on the receipt of signals).

The description of the semantics of the CHILL/D-16 communication part consists of an axiom system based on a specialized temporal logic to deal with real-time properties. This specialized temporal logic is obtained by extending linear time temporal logic (see [LAMPORT]) with real-time operators such as $\Diamond_{\leq t_0}$ (eventually within real-time t0 from now), $\Box_{\geq t_0}$ (always from real-time t0 from now on) and the operator (before) referring to the past $\blacklozenge$.

We derive all our real-time operators from the binary real-time operator $\mathcal{U}_{=t_0}$ (strong until in real-time t0). Together with the before operator $\blacklozenge$ this demands for a new temporal logic model. This model has states as in classical temporal logic, but now a state has a real-time component, conceptually representing a kind of global clock. Therefore a necessary condition for a state transition is that time can not decrease (we allow that a state transition leaves this real-time component the same: if program control is at the beginning of a receive signal case action

our axiom system stipulates that the next state transition of this process leaves the real-time component the same and transfers program control to the 'timerstart'-location, see above).

The before operator (which is new in the context of program proving) is used for the formulation of the real-time property of the medium (see furtheron) and furthermore for the classical property that a received signal must have been sent before. In the formulation of the latter property we already recognize the natural way we can express this property by an operator which refers to the past.

To be able to recognize occurrences of signals and actions (in CHILL/D-16) as different we have to refine the notion of location in a program. In Pnueli's (and also Lamport's) works location predicates are introduced, i.e. predicates true in case control resides at a specific location of a program. For the simple kind of programs they consider these predicates are characterized syntactically (e.g. by pointing to a place in the program text). Once (dynamic) process creation is involved this simple syntactic characterization is not sufficient anymore: one can still point to a specific place in the textual definition of a process, but one can not tell the difference between this textual place in one instance of the process and the same textual place in another instance of the process. To identify uniquely every action of sending or receiving, as

is required e.g. in the expression of "every signal received has been sent before", we need a third component (besides syntactic location, i.e. textual place, and process instance). This third component identifies the number of times this specific syntactic location in this process instance has been passed (this is needed because e.g. a send action can textually be placed in a loop and hence it can be passed several times).

As an example of the use of some new operators in our axiom system we now give the formal expression of the real-time medium property (see II. above):

$$\exists c \in \mathbb{R}_{\geq 0}\, \Box \forall s_1 \in SEND\ [[\blacklozenge after(s_1.l) \wedge \neg \blacklozenge arrived(s_1)] \Rightarrow [\exists s_2 \in SEND\ [\neg \blacklozenge arrived(s_2) \wedge \Diamond_{<c} arrived(s_2)] \vee \Box_{\geq c} \neg arrived(s_1)]]$$

We now briefly explain the notations used in this axiom. The domain $\mathbb{R}_{\geq 0}$ represents the non-negative part of the real-time domain. The domain SEND is the semantic domain (as in denotational semantics) corresponding to the SEND-action. s1.l stands for the location belonging to s1 and the predicate after has its usual temporal logic meaning. The predicate arrived indicates if a signal has arrived at its destination (arrived has been defined further in the axiom system).

This axiom reads as follows: there is a non-negative real-time constant c such that whenever there is a sent, but not yet arrived, signal s1 then EITHER there will arrive SOME signal before real-time c has passed OR when real-time

c has passed s1 will not arrive anymore.

The only related work we know of is [BERNSTEIN & HARTER]. They also describe a technique for reasoning about real-time properties of concurrent programs (of asynchronous nature as real-time is involved). However, their work differs in several aspects from ours: Firstly, they consider no dynamic process creation. Secondly, they consider only real-time operators related to the temporal implication operator, while we introduce a general real-time operator $\mathcal{U}_{=t_c}$ from which many more (much needed!) real-time operators can be deduced. Thirdly, their model for program execution is more complicated than ours due to restrictions concerning interleaved execution and indivisibility of operations. Also their modeling of execution time does in no way correspond to the execution time in reality, while our modeling of execution time is directly related to real-time execution indeed. Fourthly, their work concentrates on safety properties while ours is concerned most with liveness properties, since these are what real-time programming is about.

Summarizing, the contributions of this paper consist of:

1. an axiomatic semantics of the communication part of CHILL/D-16, a language used in a nowadays typical concurrent programming environment, the telecommunication industry,

2. a study of asynchronous communication and distributed processing (in the literature of the theory of concurrent programming the emphasis has until now been laid upon synchronous communication, resp. multiprogrammed process execution),

3. the extension of linear time temporal logic to a real-time temporal logic by the introduction of the general real-time operator $U_{=t}$ and the introduction of the extra operator $\blacklozenge$ for facilitating particular kinds of reasoning; this new temporal logic seems to be appropriate to express real-time and communication properties in general in a natural way,

4. an accompanying new model for our temporal logic and its associated refinement of the notion location predicate; the model is simple, but seems nevertheless appropriate enough to be a general model for distributed computation,

5. our modeling of execution time can be directly related to execution time in reality,

6. a general real-time model of an abstract transmission medium.

References

[ADA] "The Programming Language ADA, Reference Manual", Lecture Notes in Computer Science 106, Springer Verlag, New York, 1981.

[ASTESIANO & ZUCCA] Astesiano,E. and Zucca,E., "Semantics of Distributed Processes Derived by Translation", Proceedings of the 11th GI-Jahrestagung, Informatik Fachberichte 50, pp. 78-87, Springer Verlag, New York, 1981.

[BERNSTEIN & HARTER] Bernstein,A. and Harter Jr.,P.K., "Proving Real-time Properties of Programs with Temporal Logic", Proceedings of the 8th Symposium on Operating Systems Principles, pp. 1-11, ACM Special Interest Group on Operating Systems, Vol. 15 No. 5 December 1981, New York.

[BRINCH HANSEN] Brinch Hansen,P. "Distributed Processes: A Concurrent Programming Concept", CACM 21-11, pp. 934-941, 1978.

[CHILL] "CHILL Language Definition (Rec. Z.200)", C.C.I.T.T. Study Group XI, The International Telegraph and Telephone Consultative Committee, March 1982.

[GERTH] Gerth,R.T. "A Sound and Complete Hoare Axiomatization of the ADA Rendezvous", Proceedings of the 9th ICALP, Lecture Notes in Computer Science 140, pp. 252-265, Springer Verlag, New York, 1982.

[HOARE] Hoare,C.A.R., "Communicating Sequential Processes", CACM 21, pp. 666-677, 1978.

[KUIPER & DE ROEVER] Kuiper,R. and de Roever,W.P. "Fairness Assumptions for CSP in a Temporal Logic Framework", IFIP WG2 Working Conference on the Formal Description of Programming Concepts, Garmisch, June 1982.

[LAMPORT] Lamport,L. "'Sometime' is Sometimes 'NOT Never'", A Tutorial on the Temporal Logic of Programs, SRI International CSL-86, January 1979.

[MANNA & PNUELI] Manna,Z. and Pnueli,A. "How to Cook a Temporal Proof System for your Pet Language", Proceedings of the Symposium on Principles of Programming Languages, Austin, Texas, January 1983.

[PNUELI & DE ROEVER] Pnueli,A. and de Roever,W.P. "Rendezvous with ADA - A Proof Theoretical View", Proceedings of the AdaTEC Conference, Crystal City, 1982.