Specifying message passing and real-time systems (extended abstract)

Specifying Message Passing and Real-Time Systems
(Extended Abstract)

by

Ron Koymans

86/01

January 1986

COMPUTING SCIENCE NOTES

This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science of Eindhoven University of Technology.
Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review.
Copies of these notes are available from the author or the editor.

Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands

editor: M.L. Potters

SPECIFYING MESSAGE PASSING AND REAL-TIME SYSTEMS

(Extended Abstract)

Ron Koymans

Eindhoven University of Technology
P.O.Box 513
5600 MB Eindhoven
The Netherlands

December 20, 1985

Abstract

Possibilities for a temporal logic based specification methodology for message passing and real-time systems are investigated. Generalizing a result of Sistla et al. to the expressively complete logic studied by Kamp, we show that temporal logic is severely limited in specifying message passing systems. This logical limitation leads us to a study of possible extensions of temporal logic in which messages can be uniquely identified. Furthermore, temporal logic is not suited for hard real-time applications. Nevertheless, we develop a temporal logic based specification methodology overcoming these difficulties and integrating message passing and real-time in a uniform framework.

1. Introduction

This paper investigates possibilities for a uniform framework to specify a large class of systems widely used in practice. In particular, we study message passing systems and real-time systems. The motivation for this choice is supplied by their manifold appearences in practice:

- (asynchronous) message passing is one of the most important means of interprocess communication in distributed systems, either on a high level (e.g. programming telecommunication applications in CHILL [CHILL]) or on a lower level (e.g. an implementation of Ada [Ada]).

- among the many real-time applications (e.g. on-line reservation systems) there are some highly critical systems such as computer controlled chemical plants and nuclear power stations.

As the example of CHILL shows, message passing and real-time can also be combined in one framework.

Because message passing systems are so widely used and the dangers of malfunctioning real-time systems affect most of us (think e.g. of flight control software for civil airplanes), it is of *vital* importance to develop formal techniques for reasoning about them. For message passing this development is actively going on for several years (see e.g. [MC], [B], [NDGO]). For real-time, however, the situation is alarming: theoretical research has almost comnletely ignored real-time aspects (a few favourable exceptions being [BH], [KVR] and [KSRGA]).

To be able to meet the above objectives, we require a general specification methodology to have at least the following properties:

1. it must be rigorous, that is built upon a sound mathematical basis,
2. it must be simple to use,
3. it must support modularity (i.e. hierarchical development) and compositionality (i.e. the specification of the whole system is a function of the specification of its components).

Furthermore, the following property is also desirable:

4. it is abstract: systems are specified in a black box fashion, that is only in terms of their interfaces with the environment.

For computing systems, time is of course a fundamental notion: each step of a computation (i.e. an execution sequence of the system) can be thought of as one tick of some computation clock. Temporal logic, in its classical form often called tense logic, reasons about time sequences in general and allows for the formalization of possible variations in time of a changing (dynamic) situation. It is a simple and elegant extension of classical logic with temporal operators: the classical part is used to specify states, the static situations in which a system can be at any moment, and the temporal operators specify the relation in time between states (describing the dynamic situation, i.e. the evolution of the system over time). In this way, the explicit introduction of states or of time can be avoided. Now, for computing systems such as message passing systems, computations have a definite starting point in time and may have an infinite number of steps. Hence, by specializing time to be like the natural numbers, we get a variant of tense logic, linear time temporal logic, with which we can reason about systems, viewed as generators of execution sequences. In the eight years after the introduction of this logic in the area of program verification ([P]), it has proved to be a most versatile tool for the specification and verification of concurrent systems. It can be used as the basis for a specification methodology fulfilling the four requirements listed above and a lot more as shown in the work of Manna & Pnueli, Lamport, Barringer & Kuiper and many others. Furthermore, it has been applied to specify and verify a wide variety of systems, such as concurrent programs, communication protocols, hardware, VLSI etcetera. It has been used to give axiomatic definitions of concurrent programming languages, and B.Moszkowski even turned his Interval Temporal Logic into a programming language (thereby unifying programs and specifications).

Summarizing, linear time temporal logic seems to be an excellent candidate for the basis of a general specification methodology. However, Sistla et al. were the first to indicate some of its limitations. They proved that certain types of unbounded buffers cannot be specified in linear time temporal logic. Our first result is the generalization of this to a variant of tense logic which, by a result of Kamp, is rather expressive (see the end of section 2). This generalization is the contents of section 2. In section 3.1 we show that the application of the result of Sistla et al. can also be considerably extended: a large class of message passing systems (buffers correspond to special types of message passing systems) can not be specified (now in our variant of tense logic). This gives a theoretical foundation for the fact that researchers using linear time temporal logic used to enrich their formalisms to specify such systems, e.g. by adding certain data structures (queues etc.) or by using history variables. In section 3.1 we explore some possibilities for such additions and investigate their limitations to certain types of message passing systems.

In section 3.2 we treat real-time systems. Clearly, time is still more important in this case than it is for message passing systems. So it seems wise to look for extensions of linear time temporal logic in this case too. However, the changes involved must be of a more fundamental and extensive nature. For one thing, the notion of computation is not appropriate anymore in general: some real-time systems control continuous physical entities like volume, temperature etcetera. If time is discrete, information always gets lost (this is studied in sample theory). Hence, for real-time systems we must suppose time to be dense and have to use tense logic instead of linear time temporal logic. Furthermore, in hard real-time applications, quantitative elements of time are involved (e.g. within three milliseconds). Since tense logic treats time in a qualitative way, it is unable to cope with such situations. To maintain the whole set-up of tense logic (and thereby all its advantages) we should add quantitative temporal operators in this case. Again we study some possibilities for such extensions and investigate their limitations.

2. Tense Logic and a Theorem

In this section we define our variant of tense logic and generalize lemma 4.9 of [SCFM] from linear time temporal logic to this variant. We first define the language used.

Definition: For I an arbitrary set, $L_I(U, S)$ is the language with

vocabulary: atomic propositions $P_i (i \in I)$

logical operators $\neg$, $\wedge$, U, S

formulae: $P_i (i \in I)$

$\neg f_1$, $f_1 \wedge f_2$, $f_1 U f_2$ and $f_1 S f_2$ (f_1, f_2 formulae).

We now turn to the semantics of $L_I(U, S)$. A state is a mapping from I to $\{True, False\}$. S is the set of all states. A model M is a triple $<T, <, D>$ where $<$ is a linear order on T and D a function from T to S. An interpretation is a pair $<M, t>$ where M is a model and $t \in T$. Truth of a formula $f \in L_I(U, S)$ in an interpretation $<M, t>$, notation $M, t \models f$, is inductively defined as follows:

$M, t \models P_i := D(t)(i) = True \quad (i \in I)$

$M, t \models \neg f_1 :=$ *not* $M, t \models f_1$

$M, t \models f_1 \wedge f_2 := M, t \models f_1$ and $M, t \models f_2$

$M, t \models f_1 U f_2 :=$ there exists a $t' \in T$ such that $t < t'$ and $M, t' \models f_2$ and for all $t'' \in T: (t < t''$ and $t'' < t')$ implies $M, t'' \models f_1$

$M, t \models f_1 S f_2 :=$ there exists a $t' \in T$ such that $t' < t$ and $M, t' \models f_2$ and for all $t'' \in T: (t' < t''$ and $t'' < t)$ implies $M, t'' \models f_1$.

We can give our generalization after two preparatory definitions.

Definition: Let $f \in L_I(U, S)$, M be a model, $t \in T$.

Define $[t]_{M,f} := \{g \in SF(f) \mid M, t \models g\}$ where $SF(f)$ is the set of subformulae of f (including f itself).

Definition: Let M be a model and $t_1, t_2 \in T$ such that $t_1 \leqslant t_2$.

Then $M_{t_1}^{t_2}$ is the reduction of M to $T_{t_1}^{t_2} := \{t \in T \mid t \leqslant t_1 \vee t_2 < t\}$.

Theorem: Let $f \in L_I(U, S)$, M be a model and $t_1, t_2 \in T$ such that $t_1 \leqslant t_2$ and $[t_1]_{M,f} = [t_2]_{M,f}$.

Then for all $t \in T_{t_1}^{t_2}$:

$M, t \models f$ if and only if $M_{t_1}^{t_2}, t \models f$.

Proof: By structural induction on f. The details are given in the full paper. As an illustrative example we give a sketch for one of the interesting cases.

Let $f \equiv f_1 U f_2$, M be a model and $t_1, t_2 \in T$ such that $t_1 \leqslant t_2$.

Assume

(i) $[t_1]_{M,f} = [t_2]_{M,f}$.

We are going to show that $M, t \models f$ implies $M_{t_1}^{t_2}, t \models f$ for $t \leqslant t_1$.

Hence assume

(ii) $t \leqslant t_1$ and

(iii) $M, t \models f_1 U f_2$.

We have to prove that $M_{t_1}^{t_2}, t \models f_1 U f_2$.

From (i) and the induction hypothesis we can deduce

(iv) $M, t \models f_1$ implies $M_{t_1}^{t_2}, t \models f_1$ for all $t \in T_{t_1}^{t_2}$.

(v) $M, t \models f_2$ implies $M_{t_1}^{t_2}, t \models f_2$ for all $t \in T_{t_1}^{t_2}$.

From (iii) it follows that

(vi) there exists a $t_0 \in T$ such that $t < t_0$ and $M, t_0 \models f_2$ and $M, t' \models f_1$ for all $t' \in T$ such that $t < t'$ and $t' < t_0$.

We now distinguish two cases:

(a) $t_0 \leqslant t_1$: the result follows in this case immediately from (iv),(v) and (vi)

(b) $t_1 < t_0$: in this case by (vi) we get also $M, t_1 \models f_1 U f_2$.

By (i) it follows that $M, t_2 \models f_1 U f_2$. Hence

(vii) there exists a $t_3 \in T$ such that $t_2 < t_3$ and $M, t_3 \models f_2$ and $M, t' \models f_1$ for all $t' \in T$ such that $t_2 < t'$ and $t' < t_3$.

Because of $t_1 < t_0$ and (vi) we have also

(viii) $M, t' \models f_1$ for all $t' \in T$ such that $t < t'$ and $t' \leqslant t_1$.

Then $M_{t_1}^{t_2}, t \models f_1 U f_2$ by (vii) and (viii). ■

Corollary: A large class of message passing systems can not be specified in $L_I(U, S)$, see section 3.1.

Linear time temporal logic (the case of Sistla et al.) is obtained by taking I finite and $<T, <>$ isomorphic to the natural numbers with its usual ordering (M is then called an ω-model) and noting that their operators next-time, until, last-time and since are all expressible in terms of U and S.

Concerning the expressive power of $L_I(U, S)$: in [K] it is proved that $L_I(U, S)$ with I the natural numbers is expressively complete w.r.t. the class of complete linear orders. For the class of ω-models it is shown in [GPSS] that the operator U already suffices for expressive completeness. In [GPSS] it is furthermore reported that Stavi found two additional operators U' and S' such that U, S, U' and S' are expressively complete w.r.t. the class of all linear orders. The exact definition of U' and S' is not known to us and it would be interesting to find out whether they can be incorporated in the theorem.

3. Specifying Systems

In this section we study the specification of systems by tense logic, first in general and then the special cases of message passing and real-time. As already remarked in section 1, tense logic is a very appropriate tool for specifying possible variations in time of a changing situation. The notions of state and time are implicit on the level of reasoning and are made explicit in the underlying model. The evolution of a system over time can now be directly translated to this formalism. A state of a system is nothing else but a function giving the relevant entities of the system some value. By a development D we then mean a function from T, the time domain with a linear ordering $<$, to S, the set of all states. In this way we get exactly a model $<T, < , D>$ as described in section 2. The choice of S and of $<T, <>$ of course depend on the application.

3.1. Message Passing Systems

As mentioned before, for message passing systems ω-models and the corresponding notion of computation are adequate, that is we can suppose time to be like the natural numbers. A development is then nothing but an infinite sequence of states: $s_0, s_1, \ldots$.

We first describe what kind of message passing systems we consider. Let M be the message alphabet, that is the set of all messages concerned. The interface of the system with its environment consists of two functions only: *in*(m) and *out*(m) for $m \in$ M. By *in*(m) we can give message m to the system and by *out*(m) the system successfully passes the message m to its destination. The way in which messages are handled within the system and the possibility of losing messages are two factors that determine the type of message passing involved. We impose one essential reliability condition on message passing systems: the system does not deliver messages that were not previously given to it (or in other words: at any moment, the bag of delivered messages is *some part* of the bag of accepted messages).

For our examples, we make the following selection of message passing systems:

1. perfect: each message given to the system is eventually delivered at its destination
2. initially perfect: the system behaves like a perfect system until it possibly crashes: it delivers no messages at all anymore
3. the system may loose messages, but for each message the probability of a successful transmission is greater than zero
4. the system looses at least one message but at most $k-1$ messages of each series of k messages ($k \geqslant 2$).

Buffers correspond to type 1 and 2 (buffers of type 1 are called buffers with liveness property in [SCFM]). We call types 1, 2 and 3 potentially perfect: for all n, n messages given to the system *can* result in the delivery by the system of these n messages. The internal structure of the system can influence the order in which messages are delivered. E.g. a simple transmission medium between source and destination corresponds to FIFO (first-in first-out) behaviour. On the other hand, a communication network in which all transmission media are perfect and a message is sent on to an arbitrary node of the network, is itself perfect (by probability theory each message will eventually arrive at its destination) and the messages are delivered unordered, that is in no order at all.

We now show that it is impossible to specify a potentially perfect system by tense logic. Suppose the contrary. Let f be a formula describing the system. The number of subformulae of f is boundede by $2^{|f|}$ where $|f|$ is the length of f. Now choose $n > 2^{|f|}$ and consider the model M consisting of n inputs of the same message in the first n states followed by n outputs of that message in the next n states. This is a possible behaviour of a potentially perfect system and hence f is satisfied in M. Because $n > 2^{|f|}$ there are moments i, j such that $0 \leq i < j < n$ and $[i]_{M,f} = [j]_{M,f}$. Applying the theorem of section 2 we conclude that f is also satisfied in a model in which less than n inputs are followed by n outputs. This violates however our reliability condition for a message passing system. To show the impossibility of specifying systems of type 4 we have to change the above argument slightly. We now consider a model with $k \cdot n$ inputs followed by n outputs and find i and j in the sequence of n outputs. Now according to the theorem a sequence of $k \cdot n$ inputs followed by less than n outputs is also a possible development for systems of type 4. This is however not the case. Note that we needed in the above argument only a singleton set as message alphabet. This means that adding quantification will not help, so our result does also hold for first order tense logic. We did not incorporate quantification because there are some semantical complications concerning interactions between quantification and the temporal operators (cf. [GG] II.6,II.5). The essential problem here is the fact that messages are not unique: two occurrences of a message (given twice to the system) can not be discriminated.

To be able to specify such systems and resolve the problem of message identification, researchers using linear time temporal logic have used additional means e.g. special data structures or auxiliary variables (such as histories). We now review two of these.

Lamport (see e.g. [L]) uses a queue as an additional state component to describe a FIFO transmission medium. We note the following problems with this approach:

1. using an additional internal data structure violates the abstractness requirement (see point 4 in section 1),
2. the behaviour of the additional component is described by an additional formalism such as abstract data types,
3. for different applications we get different additional components (so, in a sense, the method is not general).

Another approach is taken by Hailpern (see e.g. [H]). He uses a partially interpreted temporal logic with history variables (ranging e.g. over sequences of messages) and operations on these variables such as the prefix relation. Our comments on this approach: Histories with the prefix relation are well suited for specifying FIFO behaviour, but awkward for other ordering disciplines (like LIFO, last-in first-out). In general one has to use projections on histories to access individual elements of a history. What one would like to have is a set of operations on histories as a whole such that one can specify each application in terms of this set. So again we have a generality problem.

In the above two approaches the problem of message identification is resolved by implicitly making messages unique, by their place in the queue, respectively the history. In [KR] a third approach can be found in which linear time temporal logic is extended with a past operator and a real-time until operator. The specifications thus remain purely temporal. Having the result above in mind, for the specification of message passing systems, it is assumed on beforehand that all messages can be uniquely identified (e.g. by supplying *conceptual* timestamps). Once having accepted this, we can avoid the problems for the alternative two approaches. The method of [KR] is abstract, needs no additional formalisms and is general: in [KR] it is demonstrated that by slight changes of the specification we can describe different properties of systems (e.g. whether it can loose messages or not). A complication in this approach can be the complexity of the resulting formulae, that is the temporal operators are too low level. This problem was already addressed in [SMV] where higher level temporal operators are introduced.

In the full paper, we investigate the problem of finding a suitable specification methodology for practical message passing systems in more detail.

3.2. Real-Time Systems

As already remarked at the end of section 1, the hypothesis that time is discrete is not adequate for some real-time systems. In general we need a dense linear order. The linearity of time conforms with the absolute time picture of Newtonian physics (and even with local times in relativistic physics), but there time is supposed to be continuous like the real numbers. Philosophically, however, there is a point about observability, and we think that time need not necessarily be continuous. In our opinion, the completeness property of the real numbers is not observable. This means that the minimal choice for our time domain would be the rational instead of the real numbers. So what we assume is: the time domain T contains the rational numbers.

Another typical problem for some real-time systems are the hard real-time constraints (the promptness requirements): everytime A occurs, B must follow within 3 milliseconds (in imperfect message passing systems one can also think of the time-out for receiving an acknowledgement). Obviously, qualitative logics such as tense logic can not cope with such a situation because of lack of quantitative operators. If we still want to base our specification methodology on tense logic, we have to add such operators. For linear time temporal logic this was done in [BH] and [KVR] (further worked out in [KR]). In [BH] only quantitative eventuality operators were introduced, while [KVR] introduces a much more expressive quantitative until operator. We extend the method of [KVR] to tense logic and study expressive completeness issues. For some systems, such as the abstract transmission medium of [KVR], the specification uses quantification over the time domain to express that the medium periodically tries to transmit messages (but with which period is not known). Just as for message passing systems we investigate in the full paper the possibilities for a methodology to specify such real-time systems. In view of the foregoing arguments it would be ideal for our purposes to develop a specification methodology based on tense logic with additional quantitative operators, maintaining all merits of linear time temporal logic and integrating message passing and real-time systems in a uniform framework.

References

[Ada] *The programming language Ada. Reference manual*, LNCS 155, 1983.

[B] S.D. Brookes, *A semantics and proofsystem for communicating processes*, LNCS 164, pp. 68-85, 1984.

[BH] A.Bernstein, P.K. Harter jr. , *Proving Real-Time Properties of Programs with Temporal Logic*, 8th ACM SOSP, pp. 1-11, 1981.

[CHILL] *CHILL Recommendation Z.200 (CHILL Language Definition)*, C.C.I.T.T. Study Group XI, 1980.

[DHJR] T.Denvir, W.Harwood, M.Jackson, M.Ray, *The Analysis of Concurrent Systems*, Proceedings of a Tutorial and Workshop, Cambridge University, September 1983, to appear in LNCS.

[GG] D.Gabbay, F.Guenther, *Handbook of Philosophical Logic, Volume II*, Reidel, 1984.

[GPSS] D.Gabbay, A.Pnueli, S.Shelah, J.Stavi, *On the Temporal Analysis of Fairness*, 7th ACM POPL, pp. 163-173, 1980.

[H] B.T.Hailpern, *Verifying Concurrent Processes Using Temporal Logic*, Ph.D. Thesis, Stanford University, 1980.

[K] J.A.W.Kamp, *Tense Logic and the Theory of Linear Order*, Ph.D. Thesis, University of California, Los Angeles, 1968.

[KR] R.Koymans, W.P. de Roever, *Examples of a Real-Time Temporal Logic Specification*, in [DHJR].

[KSRGA] R.Koymans, R.K.Shyamasundar, W.P. de Roever, R.Gerth, S.Arun-Kumar, *Compositional Semantics for Real-Time Distributed Computing*, LNCS 193, pp. 167-189, 1985.

[KVR] R.Koymans, J.Vytopil, W.P. de Roever, *Real-Time Programming and Asynchronous Message Passing*, 2nd ACM PODC, pp. 187-197, 1983.

[L] L.Lamport, *STL/SERC Problems*, in [DHJR].

[MC] J.Misra, K.M.Chandy, *Proofs of Networks of Processes*, IEEE SE-7 (4), 1981.

[NDGO] Van Nguyen, A.Demers, D.Gries, S.Owicki, *Behavior: a Temporal Approach to Process Modelling*, LNCS 193, pp. 237-254, 1985.

[P] A.Pnueli, *The Temporal Logic of Programs*, 18th FOCS, pp. 46-57, 1977.

[SCFM] A.P.Sistla, E.M.Clarke, N.Francez, A.R.Meyer, *Can Message Buffers Be Axiomatized in Linear Temporal Logic*, Information and Control 63, pp. 88-112, 1984.

[SMV] R.L.Schwartz, P.M.Melliar-Smith, F.H.Vogt, *An Interval Logic for Higher-Level Temporal Reasoning*, 2nd ACM PODC, pp. 173-186, 1983.

COMPUTING SCIENCE NOTES

In this series appeared:

Nr.	Author(s)	Title
85/01	R.H. Mak	The Formal Specification and Derivation of CMOS-circuits
85/02	W.M.C.J. van Overveld	On arithmetic operations with M-out-of-N-codes
85/03	W.J.M. Lemmens	Use of a Computer for Evaluation of Flow Films
85/04	T. Verhoeff H.M.J.L. Schols	Delay-insensitive Directed Trace Structures Satisfy the Foam Rubber Wrapper Postulate
86/01	R. Koymans	Specifying Message Passing and Real-Time Systems