Real-Time Logic Revisited

Stephen Paynter[1]
[1]Matra BAe Dynamics (UK) Ltd, Filton, Bristol, UK

August 24, 2000

Abstract

This paper redefines RTL within classical many-sorted logic with arithmetic. In doing so, RTL is generalised in a number of ways. In particular, functionality is handled through the use of timed variables. Various models of time for RTL are discussed, and it is argued that, providing events satisfy a finite occurrence property, time in RTL can be continuous. A number of useful RTL theorems are stated, and it is shown that RTL can naturally express all the usual temporal requirements that are placed on real-time systems. RTL is compared with other timed logics.

1 Introduction

Real-Time Logic (RTL) was one of the first logics proposed for the specification and description of real-time systems, [JaM86]. Since then RTL has been used in the definition of the semantics of a number of graphical notations for designing real-time systems, including: Modecharts, [JLM88], [JAM94], [MSJ96]; Statecharts, [ArB96], [Arm98]; Real-Time Mode-Machines, [Pay96], and ADL, [PAH00]; and in defining the semantics of a family of communication protocols, [Sim00], and Kernels, [FoW96] and [FoW97]. Various decidable fragments of RTL have been investigated, [JaM87], [Mil92], and RTL has been integrated with untimed formalisms to form more expressive notations, [Fid92]. Extensions to RTL have been proposed to make it applicable to hybrid systems, [HaL96].

RTL was originally presented as classical unsorted first-order logic with Presburger arithmetic and an uninterpreted function symbol, [JaM86]. It used functions from state predicates to events to integrate functionality into the basic RTL timed event model. Here RTL is presented as many-sorted logic (MSL) with real and integer arithmetic, and functionality is handled by timed variables.

The main reasons for this reformulation are that it enables time to be modeled by the real numbers (as opposed to the natural numbers used in [JaM86]), and it enables a wider range of functional behaviour to be integrated into the RTL timed event model. In reformulating RTL, some of the restrictions adopted in previous definitions of RTL are removed and RTL is generalised. Table 1 summarises the main differences between the version of RTL presented here and previous presentations of RTL.

	First-Order Logic	MSL with arithmetic
Time	Discrete	Continuous
Events	only event constants	quantified event variables
Occurrence Arithmetic	Decidable (Presburger)	Full (Peano)
Actions	Positive duration	May be instantaneous
Functionality	State predicates	Predicates over Timed Variables

Table 1: RTL Generalisations

Briefly, the reasons for these changes are:

- Real-time is now usually understood to be continuous (or at least dense), and the use of dense time means that the problem of deciding *a priori* what the basic time unit should be is avoided;
- Quantification over events is often natural, for example, in defining the RTL axioms;
- Having real arithmetic for time undermines the point of restricting arithmetic for occurrences;
- Instantaneous actions are sometimes a useful fiction, see for example, [Sim00]; and Predicates over timed variables integrates the property being specified into the logic, allowing functionality to be reasoned about within RTL.

Each of these topics is treated more fully below.

Some of the restrictions in earlier presentations of RTL are hard to justify, because, in spite of them, RTL has been shown to be undecidable, [JMS88]. Hence, while acknowledging the usefulness of the decidable fragments of RTL explored in [JaM87] and [Mil92], the position in this paper is that full RTL should be made as expressive as needed to support the natural specification and analysis of real-time systems[1].

The rest of the paper is organised as follows. Section 2 describes the basic RTL model. Section 3 discusses the model of time in RTL, and argues that time may be continuous, providing event occurrences are appropriately countable. Section 4 reviews how actions, states, and variables are handled in RTL. Section 5 defines RTL within many-sorted logic. Section 6 catalogues a number

[1] Of course, expanding RTL to include, for example, full natural number arithmetic, entails the loss of completeness as well as decidability, and the inability to prove consistency, [Göd31]. However, using a result of Gentzen's, expanding RTL even further, such as by embedding it in a formal system which supports transfinite induction (up to ε_0, the first ordinal greater than $\omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}$), would make it possible to prove the consistency of RTL arithmetic, [Kle52]. Kleene hypothesises in [Kle52] that transfinite induction up to even higher ordinals might make it possible to prove the consistency of real analysis (i.e. second-order arithmetic, Z_2). Currently, the consistency of the $\prod^1_2 - CA$ fragment has been demonstrated, [Rat95].

of RTL theorems which are invariably needed when performing proof in RTL. Section 7 shows how RTL can be used to formalise a typical temporal requirements on real-time systems. Section 8 shows a shallow embedding of RTL in the PVS logic, [OSR99a]. Section 9 compares RTL with other timed logics, including the DC and ERTL.

2 The Basic RTL Model

The basic RTL concept is of *timed events*: RTL events occur at specific times and they have no duration. RTL events can re-occur at different times. An RTL event is therefore a *class* of actual event occurrences. Re-occurrences of an event must occur at later times than earlier occurrences. RTL therefore does not support the super-dense "micro-step" model described in [MaP93]. RTL has no concept of events which occur at an instant being ordered or causally related.

The RTL syntax associates together an event, a time, and the number of occurrences of that event up to that time. There is redundancy in this, as occurrence numbers can be derived from the history of event occurrences. Nevertheless, as will be discussed later, this makes RTL expressions particularly succinct and readable.

The original RTL syntax used an "@" function, which returned the time of a particular event occurrence.

$$@ : Event \times Occ \rightarrow Time$$

where *Event*, *Occ*, and *Time* are the types of events, occurrence numbers, and times, respectively. *Occ* is typically taken to be N^+, the set of all positive integers, although there are cases where N is more useful, either because it allows "0" to model a special start-up "initial" occurrence of an event, e.g. in [Sim00], or because the theorem prover being used to support RTL makes induction over the naturals easier than induction over the positive naturals. A discussion of the type *Time* is postponed until the next section.

$@(e, i)$ is understood to return the time of the i^{th} occurrence of e.

Although "@" has been used in some later RTL papers, e.g. [JaM94], a proposal in [JMS88] advocated the use of an occurrence relation, θ, instead.

θ is a predicate with the following signature:

$$\theta : Event \times Occ \times Time \rightarrow \mathcal{B}$$

where $\mathcal{B}$ is the Boolean sort, and $\theta(e, i, t)$ is to be read as asserting that event e occurs for the i^{th} time at time t.

A strong argument in favour of θ over @, is that $\theta(e, i, t)$ is a total function, unlike $@(e, i)$, which is undefined if there is not an i^{th} occurrence of event e. The restriction to total functions significantly simplifies the logic, and allows

classical theorems provers, such as PVS [OSR99b], to be used to reason about RTL. θ has been used in [ArB96], [Arm98], [PAH00] and [Sim00].

The basic properties of RTL events, namely, that occurrence numbers of an event increase monotonically over time, and later occurrences of an event occur at later times, are captured in the following two axioms:

Ax 1 $\forall\, e : Event, i : Occ, t_1, t_2 : Time \bullet\ \theta(e, i, t_1) \wedge\ \theta(e, i, t_2) \Rightarrow\ t_1 = t_2$

Ax 2 $\forall\, e : Event, i : Occ, t_1 : Time \bullet$
$\theta(e, i+1, t_1) \Rightarrow\ \exists\, t_2 : Time \bullet\ \theta(e, i, t_2) \wedge\ t_2 < t_1$

The above axioms which distinguish events, occurrences, and times, and which involve arithmetic of occurrence numbers, and ordering relations over times, suggests that RTL can be naturally formalised in classical many-sorted logic (MSL) extended with arithmetic, [Man93], or in classical typed higher-order logic. The definition of RTL in many-sorted logic is pursued in Section 5.

3 Time in RTL

In earlier presentations of RTL, for example [JaM86] and [JMS88], time is taken to be discrete. More precisely, *Time* has been taken to be N^+, the set of positive integers. However, since RTL was defined in 1986, real-time formalisms have come to be understood to be those which use the real numbers to represent time. That is, "real-time" is now understood to be continuous.

It is argued here that RTL can be modified to use continuous time. However, it is not possible simply to set *Time* to be $R_{\geq 0}$, the positive reals including "0". This can be seen when an event which occurs at every time is considered. It would occur more often than could be counted by an occurrence number, the cardinality of N being strictly smaller than $R_{\geq 0}$. The same argument does not disallow dense time, as Q, the rationals, has the same cardinality as N. However, the systems modeled in rational time can still exhibit so-called "Zeno" behaviour, where time advances but is bounded.

Since [AbL91] it has been recognised that it is not necessary to impose an *a priori* lower bound on the gap between time points to prohibit Zeno behaviour. The DC, [ZHR91], for example, allows continuous time, but requires a "finite variability" constraint to ensure integrability. Similarly, a "finite occurrence density" could be added to RTL disallowing Zeno behaviour, and allowing events in continuous time to have an natural number occurrence.

The finite occurrence density (FOD) criterion is defined using a new occurrence predicate, ψ, which is true whenever a particular event occurs at a particular time. That is, $\psi(e, t)$ says event e occurs at time t.

The FOD criterion is:

Ax 3 $\forall\, e : Event, t : Time \bullet\ \psi(e, t) \Rightarrow\ \exists\, i : Occ \bullet\ \theta(e, i, t)$

This ensures that there is always a finite number of occurrences of an event up to any (finite) time. There is no contradiction in assuming time is continuous if events obey the FOD criterion.

Clearly, $\psi(e,t)$ will also satisfy the following property:

Ax 4 $\forall\ e : Event, i : Occ, t : Time \bullet\ \theta(e,i,t) \Rightarrow\ \psi(e,t)$

It is noted that all finite event occurrence properties could be be expressed using ψ instead of θ. For example, the fact that an event e occurs for the third time at time t, i.e. $\theta(e,3,t)$, would be:

$$\begin{array}{l}\psi(e,t) \wedge\ \exists\ t_1, t_2 : Time \bullet \\ \quad \psi(e,t_1) \wedge\ \psi(e,t_2) \wedge\ t_1 < t \wedge\ t_2 < t \wedge\ t_1 \neq\ t_2 \wedge \\ \quad \neg\ \exists\ t_3 : Time \bullet\ t_3 < t \wedge\ \psi(e,t_3) \wedge\ t_3 \neq\ t_1 \wedge\ t_3 \neq\ t_2.\end{array}$$

Obviously, the use of occurrence numbers are more readable and succinct, especially for higher occurrence numbers.

An infinite occurrence property that cannot be expressed using ψ is that an event, e say, occurs an unbounded number of times. That is:

$$\forall\ i : Occ \bullet\ \exists\ t : Time \bullet\ \theta(e,i,t).$$

It is noted that one implication of using $R_{\geq 0}$ for time is that time becomes totally ordered. Lamport, however, has argued convincingly that in some applications, this is a stronger property than is required, [Lam86]. However, the development of a weakly monotonic RTL is beyond the scope of this paper. Another implication of using $R_{\geq 0}$ is that the basic elements of time in RTL are point instances - the real numbers. This is in contrast with interval logics such as DC, [ZHR91], where the basic elements of time are intervals. See [Haj95] for a discussion of various ways point and interval temporal structures can be axiomatised.

In the author's experience, many proofs about systems described in RTL proceed by induction over the occurrence numbers of events, and make no use of the denseness or discreteness of time.

One property which only holds with discrete time is the following relationship between occurrence numbers and times, [Arm99], (assuming the type system allows occurrences and times to be compared). :

$$\forall\ e : Event, i : Occ, t : Time \bullet\ \theta(e,i,t) \Rightarrow\ i \leq\ t$$

4 Actions, States and Variables in RTL

In RTL actions are characterised by two events: a start and a stop event. Two functions, $\uparrow$ and $\downarrow$, from actions to events return start and stop events, respectively.

In previous RTL papers, a property of actions which distinguished them from events was that they had duration. In particular, that:

$$\forall\ a : Action, i : Occ, t_2 : Time \bullet$$
$$\theta(\downarrow a, i, t_2) \Rightarrow\ \exists\ t_1 : Time \bullet\ \theta(\uparrow a, i, t_1) \wedge\ t_1 < t_2$$

A minor but important generalisation is to allow RTL actions to be instantaneous. It is occasionally useful to be able to exploit such actions, for example, when using actions to model parts of an algorithm which may not be executed, [Sim00].

Ax 5 $\forall\ a : Action, i : Occ, t_2 : Time \bullet$
$$\theta(\downarrow a, i, t_2) \Rightarrow\ \exists\ t_1 : Time \bullet\ \theta(\uparrow a, i, t_1) \wedge\ t_1 \leq t_2$$

In modeling actions which are only executed by sequentially, the following property holds:

Ax 6 $\forall\ a : Action, i : Occ, t_2 : Time \bullet$
$$\theta(\uparrow a, i+1, t_2) \Rightarrow\ \exists\ t_1 : Time \bullet\ \theta(\downarrow a, i, t_1) \wedge\ t_1 \leq t_2$$

It is noted that this does not prevent different types of actions from occurring concurrently.

States can be handled in a similar way to actions, using functions which return the entering and leaving of a state. Examples of the use of RTL to describe state-machines can be found in [Arm98] and [PAH00].

RTL, as described in [JaM86] and [JMS88], had functions from "state predicate" identifiers to events, denoted by "$:= T$" and "$:= F$". The events they returned were defined to occur when the state predicate became true or false. For example, $\theta(\alpha := T, i, t)$ asserts that state predicate α became true for the i^{th} time at time t, where α is defined outside of RTL.

One limitation of an approach which fails to integrate the state predicate language into RTL can be seen when another state predicate, β say, is considered, such that, in an appropriate logic: $\alpha \vdash \beta$. Knowing $\psi(\alpha := T, t)$ one would wish to be prove within RTL that:

$$\exists\ t_1 : Time \bullet\ t_1 \leq\ t \wedge\ \psi(\beta\ := T, t) \wedge$$
$$\neg\ \exists\ t_2 : Time \bullet\ t_1 \leq\ t_2 \leq\ t \wedge\ \psi(\beta\ := F, t).$$

This cannot be deduced within RTL as previously presented. Another problem arises with a (presumably) untimed state predicate language, namely, that it is not possible to describe the predicates over variables at multiple times, for example, to constrain a variable to values which increase monotonically over time.

An alternative approach advocated here is the direct use of predicates over timed variables. That is, variables should be modeled as functions from time to their value domain. For example, a variable v of type T becomes:

$$v : Time \rightarrow\ T$$

This enables single time predicates to be defined, such as "$v_1(t) = v_2(t)$," as well as multiple time predicates, such as, "$v_1(t_1) > v_2(t_2)$".

It is noted that such predicates can be easily related to the core RTL timed event model, such as, for example, by the following:

$$\forall\ t : Time \bullet\ v_1(t) = v_2(t) \Rightarrow\ \psi(e,t).$$

However, relating a single time predicate (P say, over a set of variables, V) to events when it becomes true and false is a common enough requirement that it is helpful to define the following "schemas":

$$\begin{array}{l}
\nearrow:\ PREDICATE \times Event \times Time \rightarrow \mathcal{B} \\
\nearrow (P,e,t) \triangleq \psi(e,t) \Leftrightarrow \\
\quad (P(V(t)) \wedge\ (t = 0 \vee \\
\qquad \exists\ t_1 : Time \bullet\ t_1 < t \wedge\ \forall\ t_2 : Time \bullet\ t_1 < t_2 < t \Rightarrow\ \neg\ P(V(t_2)))) \vee \\
\quad (\neg\ P(V(t)) \wedge\ (t = 0 \vee \\
\qquad \exists\ t_1 : Time \bullet\ t_1 > t \wedge\ \forall\ t_2 : Time \bullet\ t < t_2 < t_1 \Rightarrow\ P(V(t_2))))
\end{array}$$

$$\begin{array}{l}
\searrow:\ PREDICATE \times Time \rightarrow \mathcal{B} \\
\searrow (P,e,t) \triangleq \psi(e,t) \Leftrightarrow \\
\quad (\neg\ P(V(t)) \wedge\ (t = 0 \vee \\
\qquad \exists\ t_1 : Time \bullet\ t_1 < t \wedge\ \forall\ t_2 : Time \bullet\ t_1 < t_2 < t \Rightarrow\ P(V(t_2)))) \vee \\
\quad (P(V(t)) \wedge\ (t = 0 \vee \\
\qquad \exists\ t_1 : Time \bullet\ t_1 > t \wedge\ \forall\ t_2 : Time \bullet\ t < t_2 < t_1 \Rightarrow\ \neg\ P(V(t_2))))
\end{array}$$

Here $\nearrow$ ("becomes true") and $\searrow$ ("becomes false") are schemas, or templates which should be expanded when encountered. Turning them into functions or operators in the logic would require RTL to be made a full secord or higher order logic, and not only one which exploits second order concepts in the definition of real and integer arithmetic. The definitions of $\nearrow$ and $\searrow$ are based on almost identical definitions in ERTL, [LeH95].

Other helpful schemas for RTL are the ones which assert that:

- a predicate holds over a period;
- a predicate will hold until a particular time; and
- a predicate started to hold at a particular time.

The following are definitions of these schemas:

$$\begin{array}{l}
hold_over :\ PREDICATE \times Time \times Time \rightarrow \mathcal{B} \\
hold_over(P,t_1,t_2) \triangleq \forall\ t : Time \bullet\ t_1 \geq\ t \geq\ t_2 \Rightarrow\ P(V(t)).
\end{array}$$

$$\begin{array}{l}
hold_until :\ PREDICATE \times Time \times Time \rightarrow \mathcal{B} \\
hold_until(P, now, then) \triangleq \\
\quad hold_over(P, now, then) \wedge\ \exists\ e : Event \bullet\ \psi(e, then) \wedge\ \searrow (\mathrm{P}, e, then)
\end{array}$$

$hold_since: \ PREDICATE \times Time \times Time \rightarrow \mathcal{B}$
$hold_since(P, then, now) \triangleq$
$\quad hold_over(P, then, now) \wedge \ \exists \ e : Event \bullet \ \psi(e, then) \wedge \ \nearrow (\mathrm{P}, e, then).$

There are many other possible schemas, although two which are likely to be useful are schemas which assert:

- the time a predicate has held within a period; and
- the number of times a predicate has switched within a period.

An example definition of the first schema is given below, under the assumption that predicates over variables obey the finite variability condition, see [ZHR91].

$duration: \ PREDICATE \times Time \times Time \times Time \rightarrow \mathcal{B}$
$duration(P, begin, end, t) \triangleq$
$\quad (hold_over(P, begin, end) \Rightarrow \ t = end - begin) \wedge$
$\quad (\exists \ t_1 : Time \bullet \ begin \leq \ t_1 < end \wedge \ hold_over(P, begin, t_1) \Rightarrow$
$\qquad t = t_1 - begin + duration(P, t_1, end)) \wedge$
$\quad (\neg \ (\exists \ t_1 : Time \bullet \ begin \leq \ t_1 \wedge \ hold_over(P, begin, t_1)) \wedge$
$\qquad \exists \ e : Event, t_2 : Time \bullet \ begin \leq \ t_2 < end \wedge \ \nearrow (P, e, t_2) \wedge$
$\qquad\quad \neg \ (\exists \ e_1 : Event, t_3 : Time \bullet$
$\qquad\qquad begin \leq \ t_3 < t_2 \wedge \ \nearrow (P, e_2, t_3))) \Rightarrow \ t = duration(P, t_2, end) \wedge$
$\quad (\neg \ (\exists \ t_1 : Time \bullet \ begin \leq \ t_1 \wedge \ hold_over(P, begin, t_1)) \wedge$
$\qquad \neg \ \exists \ e : Event, t_2 : Time \bullet \ begin \leq \ t_1 < end \wedge \ \nearrow (P, e, t_2)) \Rightarrow \ t = 0$

It is noted the above definition is recursive, and hence depends upon the finite variability condition for termination. It is further noted that "*duration(P, begin, end, t)*" is similar to the Duration Calculus expression: "$t = \int_{begin}^{end} P$", [ZHR91]. It is noted that using this definition of "duration" means that RTL is not as expressive as MTL-$\int$, [LaH94].

5 Many-Sorted Logic

Many-Sorted Logic (MSL), [Man93], modifies classical first-order logic by:

- identifying sorts (or types) which each constant and variable;
- identifying sorted (typed) signatures with each function; and by
- requiring quantification to range over a particular sort.

MSL is strictly no more expressive than unsorted logic, but for many purposes the notation is more compact and readable. MSL has been advocated as a unifying logic in which other logics should be defined, [Man96].

5.1 The MSL Language

The basic symbols of the version of MSL used here are: { (,), $\neg$, $\wedge$, $\forall$, :, $\bullet$, = }, along with an infinite set of sort symbols, {$\mathcal{B}$, S_1, S_2, S_3, ...}, where $\mathcal{B}$ is the "Boolean" sort; an infinite set of variables, { x_1, x_2, x_3, ..., y_1, y_2, ... }, where a sort is associated with each variable; and an infinite set of function symbols of each arity, from zero upwards, {F_1^0, F_2^0, F_3^0, ..., F_1^1, F_2^1, ... }, where F_j^n is the j^{th} function F with arity n. Associated with each function symbol, F_j^n is its signature which defines the sorts of the parameters and result of the function, $F_j : S_1 \times S_2 \times ... \times S_n \rightarrow S$. The arity of a function is its number of parameters. Functions with an arity of zero are known as *constants*. Functions of result sort $\mathcal{B}$ are known as *predicates*.

The well-formed sentences of MSL are defined by the following grammar:

S_k-**Var ::=** v, where the sort of v is S_k;

S_k-**Term ::=** S_k-Var | $F_j^n(t_1, t_2, ..., t_n)$, where F_j^n is a function with signature: $S_1 \times S_2 \times ... \times S_n \rightarrow S_k$ and each t_i, i in $1..n$, is an S_i-term;

S ::= '(' $S_{\mathcal{B}}$ -Term ')' | '('$\neg$' S ')' | '('S '$\wedge$' S ')' | '(' S_k-Term '=' S_k-Term ')' | '(' '$\forall$' S_k-Var ':' 'S_k' '$\bullet$' S ')';

The usual abbreviations and conventions are typically used in MSL, in particular:

- the arity indicator is dropped from function symbols;
- more meaningful function, sort, and variable names are used where this aids readability;
- the other propositional connectors and the existential quantifier are defined as usual;
- brackets are dropped when the assumption of the usual precedence rules produces the desired sentence; and
- quantifiers are not repeated when multiple variables are quantified by the same quantifier, and sorts are not repeated when variables are of the same sort. For example, "$\forall\ v_1, v_2 : S_1, v_3 : S_2 \bullet\ P(v_1, v_2, v_3)$" is short hand for "$\forall\ v_1 : S_1 \bullet\ \forall\ v_2 : S_1 \bullet\ \forall\ v_3 : S_2 \bullet\ P(v_1, v_2, v_3)$"

5.2 An MSL Proof System

A natural deduction proof system for MSL is given in Table 2.

Introduction Rules	Elimination Rules
$\wedge$-i $= \dfrac{\vdash \alpha;\ \vdash \beta}{\vdash \alpha\ \wedge\ \beta}$	$\wedge - e_1 = \dfrac{\vdash \alpha\ \wedge\ \beta}{\vdash \alpha}$ $\wedge - e_2 = \dfrac{\vdash \alpha\ \wedge\ \beta}{\vdash \beta}$
$\vee - i_1 = \dfrac{\vdash \alpha}{\vdash \alpha\ \vee\ \beta}$ $\vee - i_2 = \dfrac{\vdash \alpha}{\vdash \beta\ \vee\ \alpha}$	$\vee$-e $= \dfrac{\vdash \alpha\ \vee\ \beta;\ \alpha\ \vdash \gamma;\ \beta\ \vdash \gamma}{\vdash \gamma}$
$\neg\neg$-i $= \dfrac{\vdash \alpha}{\vdash \neg\neg\alpha}$	$\neg\neg$-e $= \dfrac{\vdash \neg\neg\alpha}{\vdash \alpha}$
$\Rightarrow$-i $= \dfrac{\alpha\ \vdash \beta}{\vdash \alpha\ \Rightarrow\ \beta}$	$\Rightarrow$-e $= \dfrac{\vdash \alpha;\ \vdash \alpha\ \Rightarrow\ \beta}{\vdash \beta}$
$\Leftrightarrow$-i $= \dfrac{\vdash \alpha\ \Rightarrow\ \beta\ \wedge\ \beta\ \Rightarrow\ \alpha}{\vdash \alpha\ \Leftrightarrow\ \beta}$	$\Leftrightarrow$-e $= \dfrac{\vdash \alpha\ \Leftrightarrow\ \beta}{\vdash \alpha\ \Rightarrow\ \beta\ \wedge\ \beta\ \Rightarrow\ \alpha}$
$\neg$-i $= \dfrac{\alpha\ \vdash \bot}{\vdash \neg\alpha}$	$\neg$-e $= \dfrac{\vdash \alpha;\ \vdash \neg\alpha}{\vdash \bot}$
$\bot$-i = There is no such rule.	$\bot$-e $= \dfrac{\vdash \bot}{\vdash \alpha}$
$=$-i $= \dfrac{}{\vdash t = t}$	$=$-e $= \dfrac{\vdash t_1 = t_2;\ \vdash \alpha\ [t_1/x]}{\vdash \alpha\ [t_2/x]}$
$\forall$-i $= \dfrac{x_0 \vdash \alpha\ [x_0/x]}{\vdash \forall\ x : S \bullet\ \alpha}$	$\forall$-e $= \dfrac{\vdash \forall\ x : S \bullet\ \alpha}{\vdash \alpha\ [t/x]}$
$\exists$-i $= \dfrac{\vdash \alpha\ [t/x]}{\vdash \exists\ x : S \bullet\ \alpha}$	$\exists$-e $= \dfrac{\vdash \exists\ x : S \bullet\ \alpha; x_0,\ \alpha\ [x_0/x] \vdash \beta}{\vdash \beta}$

Table 2: A Natural Deduction System for MSL

5.3 MSL Models

The structure, St, in which MSL sentences are interpreted, contains a set of domains, D. There is a domain d_{S_i} of values for each sort, S_i, of the language. St also contains a set of functions, F, which contains a function, $f_{F_j^n}$, for each function in in the language, and a set of variable names, V. An assignment, A, is mapping from $V \rightarrow D$, where each v of each sort S_i gets assigned a value from d_{S_i}. An special equality function Eq is defined as expected over values in the elements of D.

The interpretation function for MSL, I, is defined by the following rules:

1. For every variable, v, $I(v) = A(v)$
2. For every function, F_j^n, $I(F_j^n(\alpha_1, ..., \alpha_n) = f_{F_j^n}(I(\alpha_1), ...I(\alpha_n))$.
3. For every sentence, s, $I(\neg\ s) = 1$ if $I(s) = 0$, otherwise $I(\neg\ s) = 0$.
4. For every pairs of sentences, s_1 and s_2, $I(s_1 \wedge\ s_2) = 1$ if $I(s_1) = 1$ and $I(s_2) = 1$, otherwise $I(s_1 \wedge\ s_2) = 0$.
5. For every pair of expressions, e_1 and e_2, $I(e_1 = e_2) = 1$ if $Eq(I(e_1, e_2) = 1$, otherwise $I(e_1 = e_2) = 0$.
6. $I(\forall\ v : S_i \bullet\ S) = 1$ if for every value, x, in d_{S_i}, $I(S[x/v]) = 1$, otherwise $I(\forall\ v : S_i \bullet\ S) = 0$

A model of MSL is a tuple: $[St, I]$.

5.4 RTL in MSL

RTL can be defined in MSL by adding sorts: *Events*, *Time*, *Occ* and *Action*, the functions θ, ψ, $\uparrow$, $\downarrow$ and the normal arithmetical operations, $+$, $-$, $*$, $/$, $<$, $>$, $\leq$, and $\geq$. θ, ψ, $\uparrow$ and $\downarrow$ are axiomatised using the axioms given above. These axioms constrain the standard MSL model that the interpretation function I defines for RTL.

Clearly, embedding RTL in an 'MSL' requires the logic to be more expressive than first-order logic, if full natural number or real arithmetic is exploited. Therefore, although the RTL syntax does not explicitly support secord-order predicates and quantification, it implicitly does by exploiting arithmetic: RTL is not a first-order logic.

6 RTL Theorems

This section includes a number of simple but useful RTL theorems, each of which the author has proven using the PVS theorem prover, [OSR99b], and the shallow embedding of RTL in PVS given in Section 8.

Earlier occurrences of events occur at earlier times.

Th 1 $\forall\, e : Event, i, j : Occ, t_1, t_2 : Time \bullet$
$\theta(e, i, t_1) \wedge\ \theta(e, j, t_2) \wedge\ i < j \Rightarrow\ t_1 < t_2$

Only one occurrence of an event occurs at a time.

Th 2 $\forall\, e : Event, i, j : Occ, t : Time \bullet\ \theta(e, i, t) \wedge\ \theta(e, j, t) \Rightarrow\ i = j$

If an event has occurred for the i^{th} time, it has occurred for all earlier occurrences (at earlier times).

Th 3 $\forall\, e : Event, i : Occ, t_1 : Time \bullet$
$\theta(e, i, t_1) \wedge\ i > 0 \Rightarrow$
$(\forall\, j : Occ \bullet\ j < i \Rightarrow\ \exists\, t_2 : Time \bullet\ \theta(e, j, t_2) \wedge\ t_2 < t_1)$

An earlier occurrence of an event has an earlier occurrence number.

Th 4 $\forall\, e : Event, i, j : Occ, t_1, t_2 : Time \bullet$
$\theta(e, i, t_1) \wedge\ \theta(e, j, t_2) \wedge\ t_1 < t_2 \Rightarrow\ i < j$

An earlier or equal occurrence of an event has an earlier or equal occurrence number. (This is a simple corollary of theorems 2 and 4.)

Th 5 $\forall\, e : Event, i, j : Occ, t_1, t_2 : Time \bullet$
$\theta(e, i, t_1) \wedge\ \theta(e, j, t_2) \wedge\ t_1 <= t_2 \Rightarrow\ i <= j$

7 Typical Real-Time Requirements in RTL

This section has two aims: one, to provide a formal statement of a number of common real-time requirements, and two, to provide some validation of the RTL notation, by showing that it is capable of expressing simply and clearly these common requirements.

One of the commonest properties of real-time systems (or assumptions about their environments) is that certain events occur periodically. Formally,

$periodic :\ Event \times Time \rightarrow \mathcal{B}$
$periodic(e, period)\ \underline{\triangle}$
$\forall\, i : Occ, t_1, t_2 : Time \bullet\ \theta(e, i, t_1) \wedge\ \theta(e, i+1, t_2) \Rightarrow\ t_2 = t_1 + period.$

Another is that an event is sporadic (that is, aperiodic, with a minimum inter-arrival time or *miat*). Formally,

$sporadic :\ Event \times Time \rightarrow \mathcal{B}$
$sporadic(e, miat)\ \underline{\triangle}$
$\forall\, i : Occ, t_1, t_2 : Time \bullet\ \theta(e, i, t_1) \wedge\ \theta(e, i+1, t_2) \Rightarrow\ t_2 \geq\ t_1 + miat.$

A common requirement of a real-time computation is that one event (typically an input or triggering event) causes another event (typically an output) within a lower and upper time bound of the first event occurring. Formally,

$$deadline : \ Event \times Event \times Time \times Time \rightarrow \mathcal{B}$$
$$deadline(e_1, e_2, l, u) \ \underline{\triangle}$$
$$\forall \ i : Occ, t_1 : Time \bullet \ \theta(e_1, i, t_1) \Rightarrow \ \exists \ t_2 : Time \bullet \ \theta(e_2, i, t_2) \wedge \ t_1 + u \geq \ t_2 \geq \ t_1 + l.$$

Another common requirement is that an (input or output) event must fall within a time window relative to a periodic event (such as a clock 'tick').

$$window : \ Event \times Event \times Time \times Time \rightarrow \mathcal{B}$$
$$window(e, clock, l, u) \ \underline{\triangle}$$
$$\forall \ i : Occ, t_1 : Time \bullet \ \theta(e, i, t_1) \Rightarrow$$
$$\exists \ j : Occ, t_2 : Time \bullet \ \theta(clock, j, t_2) \wedge \ t_2 + u \geq \ t_1 \geq \ t_2 + l.$$

A more interesting real-time requirement to formalise is *jitter*, because of the lack of consensus on its definition. Here a definition is given that the author has found useful. A jitter requirement is a positive and negative timing constraint on when one (nominally periodic) event may occur relative to a periodic event. To avoid the proliferation of events, it is helpful to allow the jitter constraints to be offset from the periodic event.

$$jitter : \ Event \times Event \times Time \times Time \times Time \rightarrow \mathcal{B}$$
$$jitter(e, clock, x_1, x_2, offset) \ \underline{\triangle}$$
$$\forall \ i : Occ, t_1, t_2 : Time \bullet \ \theta(e, i, t_1) \wedge \ \theta(clock, i, t_2) \Rightarrow$$
$$t_2 + offset + x_2 \geq \ t_1 \geq \ t_2 + offset - x_1.$$

A related requirement to jitter, but one which it is necessary to distinguish, is a temporal constraint on consecutive occurrences of a (nominally periodic) event. The author refers to this as a *Consecutive Occurrence Bound* or COB.

$$COB : \ Event \times Time \times Time \rightarrow \mathcal{B}$$
$$COB(e, max, min) \ \underline{\triangle}$$
$$\forall \ i : Occ, t_1, t_2 : Time \bullet \ \theta(e, i, t_1) \wedge \ \theta(e, i + 1, t_2) \Rightarrow \ t_1 + max \geq \ t_2 \geq \ t_1 + min.$$

The formalisation of these requirements have arisen out of work with the BAE SYSTEMS Dependable Computing System Centre on capturing real-time requirements using real-time transactions, [Hav97] and [Pay99].

8 A Shallow Embedding of RTL in PVS

This section presents an encoding of RTL events and actions in the PVS logic, [OSR99a]. It is similiar to other encodings, e.g. [FoW97].

```
rtl :  THEORY
BEGIN
```

```
 Event:  NONEMPTY_TYPE
 Time:  TYPE = real
 Occ:  TYPE = nat
 Action :  NONEMPTY_TYPE

 th :  [Event, Occ, Time -> bool]
 psi :  [Event, Time -> bool]
 up :  [Action -> Event]
 dwn :  [Action -> Event]

------------ RTL Axioms

 RTLax1:  AXIOM
  FORALL (e :  Event, i :  Occ, t1, t2 :  Time):
   th(e, i, t1) AND th(e, i, t2) IMPLIES t1 = t2

 RTLax2:  AXIOM
  FORALL (e :  Event, i :  Occ, t1 :  Time):
   th(e, i + 1, t1) IMPLIES EXISTS (t3 :  Time):  th(e, i, t3) AND t3 < t1

 RTLax3 :  AXIOM
  FORALL (e :  Event, t :  Time):
   psi(e, t) IMPLIES EXISTS (i:  Occ):  th(e, i, t)

 RTLax4 :  AXIOM
  FORALL (e :  Event, i :  Occ, t :  Time):
   th(e, i, t) IMPLIES psi(e, t)

 RTLax5 :  AXIOM
  FORALL (a :  Action, i :  Occ, t1 :  Time):
   th(dwn(a), i, t1) IMPLIES
    EXISTS(t2 :  Time):  th(up(a), i, t2) AND t2 <= t1

 RTLax6 :  AXIOM
  FORALL (a :  Action, i :  Occ, t1 :  Time):
   th(up(a), i + 1, t1) IMPLIES
    EXISTS(t2 :  Time):  th(dwn(a), i, t2) AND t2 <= t1

END rtl
```

9 Conclusions and Related Work

This paper has represented RTL in classical many-sorted logic with real and integer arithmetic. In doing so, a finite occurrence density axiom has been

added which allows RTL time to be continuous, and RTL actions have been modified to allow them to be instantaneous. It has been argued that functional behaviour should be handled in RTL using predicates over timed variables, and mappings between these predicates and events have been defined. RTL has been shown to be a natural formalism for expressing a wide range of typical requirements on real-time systems, and a shallow embedding of RTL has been given in the PVS logic.

One of the main logics discussed in the literature for specifying and modeling real-time systems is the Duration Calculus (DC). The basic duration calculus, [ZHR91], is a modal interval logic with a chop operator, extended with an integration operator that sums the time that a state predicate has been true over an interval. The DC has continuous time, although state predicates are constrained to have finite variability. Many variants of the DC has been proposed in the literature, including weakly monotonic time and probabilistic versions, [PaH98] and [Liu96]. The DC has been given a deeper embedding into PVS than the RTL embedding presented above, [SkS94].

It was illustrated in Section 4 how a "duration" schema could be used to handle predicates over timed variables in a similar style to the way durations are handled in the DC. However, without further syntactic changes, RTL would still refer to explicit times, something which the designers of the DC tried to avoid. The author remains to be convinced of the advantages of hiding references to time. The ability in RTL to refer to the number of occurrences of an event makes it relatively easy to specify properties such as, "the pump may be turned no more than ten times in any interval of x time units." It is not immediately obvious how to capture this in the DC, although it is a typical type of requirement for a systems such as mine pumps and gas burners, which the DC has been developed to handle.

A variant of RTL has been proposed in the literature, *Extended RTL* or *ERTL*, [LeH95] and [Hal96]. ERTL adds a new predicate to RTL, the *holding* predicate, Φ. Φ asserts that a particular *untimed* predicate holds for a particular occasion at a particular time. That is, $\Phi(P, i, t)$ asserts that (untimed) predicate P holds for the i^{th} time at time t. ERTL stratifies the language into untimed and timed predicates to avoid making Φ a secord-order predicate.

It is noted that RTL (as defined here) is already a second-order logic, thus presumably undermining one motivation for ERTL. It is also noted that in ERTL it is not possible to define the value of a variable at one time as depending upon its value at an earlier time - for example, a monotonically increasing value over time cannot be defined because the predicate asserted to hold and which constraints the value, being untimed, cannot refer to earlier times. This contrasts with multiple time predicates over timed variables discussed in Section 4.

RTL would benefit from better tool support, for example, by producing a deeper embedding of RTL into PVS, similar to the DC proof assistant described in [SkS94]. The author's main interest in RTL, however, is in using it to explicate the semantics of various informal concepts and notations used in the specification and design of real-time systems, [Pay96], [Pay99], [PAH00], and [Sim00]. RTL has proved to be a simple yet expressive logic for such work.

10 Acknowledgements

Matra BAe Dynamics (UK) Ltd. funded this research. My ideas about RTL have benefited from conversations with Drs. J.M. Armstrong and J.S. Fitzgerald, and Mr. N. Henderson.

References

[AbL91] Abadi, M. and Lamport, L.: 'An Old Fashioned Recipe for Real-Time', *Proc. REX Workshop - Real-Time: Theory in Practice*, LNCS 600, Springer, 1991, pp. 1-27.

[ArB96] Armstrong, J.M. and Barroca, L.: 'Specification and Verification of Reactive System Behaviour: The Railroad Crossing Example', *J. Real-Time Systems*, 10 (2), March 1996, pp. 143-178.

[Arm98] Armstrong, J.M.: 'Industrial Integration of Graphical Formal Specifications', *J. Systems Software*, 40, 1998, pp. 211-225.

[Arm99] Armstrong, J.M.: 'The Proofcharts-PVS Manual - Part II: The Proofcharts-PVS Theorem Library', BAE SYSTEMS DCSC Technical Report, DCSC/TR/1999/5, University of Newcastle, 1999.

[Fid92] Fidge, C.J.: 'Specification and Verification of Real-Time Behaviour Using Z and RTL', *Proc.* 2^{nd} *Int. Symp. on Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 571, Springer, 1992.

[FoW96] Fowler, S. and Wellings, A.J.: 'Formal Analysis of a Real-Time Kernel Specification', *Proc.* 4^{th} *Int. Symp. on Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 1135, Springer, 1996.

[FoW97] Fowler, S. and Wellings, A.J.: 'Formal Development of a Real-Time Kernel', *Proc.* 18^{th} *IEEE Real-Time Systems Symp.*, San Francisco, December 1997.

[Göd31] Gödel, K.: 'Über formal unentscheidbare Sätze der Principia Mathematica and verwandter Systeme I', In Monatshefte für Mathematik and Physik, Vol. 38, pp. 173-198, 1931. Reprinted in Translation in 'On Formally Undecidable Propositions of Principia Mathematica and Related Systems', Dover Publications, 1992.

[HaL96] Hall, J.G. and Lemos, R. de.: 'ERTL: An Extension to RTL for the Specification, Analysis and Verification of Hybrid Systems', *Proc. of the IEEE EuroMirco'96 Conf.*, 1996.

[Haj95] Hajnicz, E.: 'An Analysis of Structure of Time in the First Order Predicate Calculus', In L. Bolc and A. Szalas (Editors): 'Time and Logic: A Computational Approach', UCL Press (London), 1995, pp. 279-322.

[Hav97] Haveman, J.: 'Transaction Decomposition: Refinement of Timing Constraints', *Proc. of the South Pacific Conf. on Formal Methods*, 1997.

[JaM86] Jahanian, F., and Mok, A.K.: 'Safety Analysis of Timing Properties in Real-Time Systems', *IEEE Trans. on Soft. Eng.*, 12(9), 1986, pp. 890-904.

[JaM87] Jahanian, F., and Mok, A.K.: 'A Graph-Theoretic Approach for Timing Analysis and its Implementation', *IEEE Trans. on Comp.*, 36(18), 1987, pp. 961-975.

[JaM94] Jahanian, F., and Mok, A.K.: 'Modechart: A Specification Language for Real-Time Systems', *IEEE Trans. on Soft. Eng.*, 20(12), 1994, pp. 933-947.

[JLM88] Jahanian, F., Lee, R. and Mok, A.K.: 'Semantics of Modechart in Real-Time Logic', *IEEE Proc. of the* 21^{st} *Annual Hawaiian Int. Conf. on System Science*, 1988.

[JMS88] Jahanian, F., Mok, A.K., and Stuart, D.A.: 'Formal Specification of Real-time Systems', Technical Report TR-88-25, Department of Computer Science, University of Texas at Austin, June 1988.

[Kee52] Kleene, S.C.: 'Introduction to Metamathematics', North-Holland Publishing Company, 1952.

[LaH94] Lakhneche, Y. and Hooman, J.: 'Reasoning about Durations in Metric Temporal Logic', *Proc.* 3^{rd} *Int. Symp. on Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 863, Springer-Verlag, 1994, pp. 488-510.

[Lam86] Lamport, L.: 'On Interprocess Communication - Part 1: Basic Formalism', *Distributed Comp.* 1, 1986, pp. 77-85.

[LeH95] Lemos, R. de. and Hall, J.G. : 'Extended RTL in the Specification and Verification of an Industrial Press', *Proc. of DIMAC'95 Conf.*, 1995.

[Liu96] Liu, Z.: 'Specification and Verification in the Duration Calculus', Chapter 7 of M. Joseph (Editor): 'Real-Time Systems: Specification, Verification, and Analysis', Prentice-Hall International Series in Computer Science, 1996.

[MaP93] Manna, Z. and Pnueli, A.: 'Models of Reactivity', *Acta Informatica*, 30(7), 1993, pp. 609-678.

[Man93] Manzano, M.: 'Introduction to Many-Sorted Logic', in 'Many-Sorted Logic and its Applications', Edited by K. Meinke and J.V. Tucker, Wiley Professional Computing, 1993

[Man96] Manzano, M.: 'Extensions of First-Order Logic', Tracts in Theoretical Computer Science, Cambridge University Press, 1996

[Mil92] Millet, O.: 'Multicycles and RTL Logic Satisfiability', *Proc.* 2^{nd} *Int. Symp. on Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 571, Springer, 1992.

[MSJ96] Mok, A.K., Stuart, D.A., and Jahanian, F.: 'Specification and Analysis of Real-Time Systems: Modechart Language and Toolset', Chapter in Heitmeyer, C. and Mandrioli, D. (Editors): 'Formal Methods for Real-Time Computing', Trends in Software 5, Wiley, 1996.

[OSR99a] Owre, S., Shanker, N., Rushby, J.M., and Stringer-Calvert, D.W.J.: 'PVS Language: Version 2.3', Computer Science Laboratory, SRI International, September 1999.

[OSR99b] Owre, S., Shanker, N., Rushby, J.M., and Stringer-Calvert, D.W.J.: 'PVS System Guide: Version 2.3', Computer Science Laboratory, SRI International, September 1999.

[PaH98] Pandya, P.K. and Hung, D.V.: 'Duration Calculus of Weakly Monotonic Time', *Proc.* 5^{th} *Int. Symp. on Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 1486, Springer, 1998.

[Pay96] Paynter, S.E.: 'Real-Time Mode-Machines', *Proc.* 4^{th} *Int. Symp. on Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 1135, Springer, 1996, pp. 90-109.

[Pay99] Paynter, S.E.: 'Real-Time Transactions Revisited', Unclassified MBD(UK) Technical Report, DR16972, 1999.

[PAH00] Paynter, S.E., Armstrong, J.M., and Haveman, J.: 'ADL: An Activity Description Language for Real-Time Networks', To Appear, *Formal Aspects of Comp.*, 2000.

[Rat95] Rathjen, M.: 'Recent Advances in Ordinal Analysis: $\prod^1_2 - CA$ and Related Systems', *Bulletin of Symbolic Logic*, 1, 1995, pp. 468-485.

[SkS94] Skakkebaek, J.U. and Shankar, N.: 'Towards a Duration Calculus Proof Assistant in PVS', *Proc.* 3^{rd} *Int. Symp. on Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 863, Springer-Verlag, 1994.

[Sim00] Simpson, H.R.: 'Protocols for Process Interaction', *IEE Proc. on Soft. Eng.*, To Appear.

[ZHR91] Zhou, C. Hoare, C.A.R., and Ravn, A.P.: 'A Calculus of Durations', *Inform. Proc. Letters*, 40, 1991, pp. 269-276.