**Published paper**

Burns A, Hayes IJ (2010)
Title: A timeband framework for modelling real-time systems
45 (1-2) 106-142
**http://dx.doi.org/10.1007/s11241-010-9094-5**

**Real-Time Systems manuscript No.**
(will be inserted by the editor)

# A Timeband Framework for Modelling Real-Time Systems

**Alan Burns · Ian J. Hayes**

March 25, 2010

**Abstract** Complex real-time systems must integrate physical processes with digital control, human operation and organisational structures. New scientific foundations are required for specifying, designing and implementing these systems. One key challenge is to cope with the wide range of time scales and dynamics inherent in such systems. To exploit the unique properties of time, with the aim of producing more dependable computer-based systems, it is desirable to explicitly identify distinct time bands in which the system is situated. Such a framework enables the temporal properties and associated dynamic behaviour of existing systems to be described and the requirements for new or modified systems to be specified. A system model based on a finite set of distinct time bands is motivated and developed in this paper.

## 1 Introduction

The construction of large socio-technical real-time systems, such as those envisaged in cyber-physical applications, imposes a number of significant challenges, both technical and organisational. Their complexity makes all stages of their development (requirements analysis, specification, design, implementation, deployment and maintenance/evolution) subject to failure and costly re-working. Even the production of an unambiguous behavioural description of an existing system is far from straightforward.

One characteristic of these computer-based systems is that they are required to function at many different time scales (from microseconds or less to days or more). Time is clearly a crucial notion in the specification (or behavioural description) of computer-based systems, but it is usually represented, in modelling schemes for example, as a single flat physical phenomenon. Such an abstraction fails to support the

A. Burns
Department of Computer Science, University of York, UK.

I.J. Hayes
The University of Queensland, School of Information Technology and Electrical Engineering, Australia.

structural properties of the system, forces different temporal notions on to the same flat description, and fails to support the separation of concerns that the different time scales of the system facilitate. Even with a single time scale, system architects seem to have great difficulty in specifying temporal properties in anything other than very concrete implementation-level terms. But just as the functional properties of a system can be modelled at different levels of abstraction or detail, so too should its temporal properties be representable in different, but provably consistent, time scales.

To make better use of 'time', with the aim of producing more dependable computer-based systems, we propose a framework that explicitly identifies a number of distinct *time bands* in which the system under study is situated. The framework enables the temporal properties of existing systems to be described and the requirements for new or modified systems to be specified.

In the following section we motivate the main notions and properties of the time-band framework. Then, in Section 3, an abstract model of timebands is presented, and in Section 4 is extended to describe state. Section 5 gives a brief summary of the model. The model is, in itself, not intended to be a complete semantic description. That is achieved by 'embedding' the model in a parent notation/logic. The focus of this paper is, however, the timeband framework. Section 6 introduces some notation to allow specification of properties in the timeband framework. A short example of the use of the framework in presented in Section 7. Related and future work is discussed in Section 8 and conclusions are covered in Section 9.

## 2 Motivation

A large real-time system exhibits dynamic behaviour on many different levels. The computational components have circuits that have nanosecond speeds, faster electronic subcomponents and slower functional units. Communication on a fast bus is at the microsecond level but may be tens of milliseconds on slow or wide-area media. Human time scales move from the 1ms neuron firing time to simple cognitive actions that range from 100ms to 10 seconds or more. Higher rational actions take minutes and even hours. Indeed it takes on the order of 1000 hours to become an expert at a skilled task, such as flying a plane [45] and the development of highly skilful behaviour may take many years. At the organisational and social level, time scales range from a few minutes, through days, months and even years. Perhaps for some environmentally sensitive systems, consequences of failure may endure for centuries. To move from nanoseconds to centuries requires a framework with considerable descriptive and analytical power.

The concept of timebands comes from a detailed study of existing computer-based systems[1] and their requirements (eg.[29]), ethnographical studies (eg. [7,2]), the work of Newell [39] in his attempts to describe human cognition, work on system structure such as that of Simon [47], reports on system failures (eg. Columbus [5]), studies from areas such as the psychology and sociology of time [19,22,44,34,3], formalisms such as the teleo-reactive programming model [40,41] and Statecharts

[1] As part of the Dependability Interdisciplinary Research Collaboration – DIRC, see web.

that require instantaneous state changes, and the few examples of modelling work that do attempt to consider more than one time scale within a system (eg. Corsetti *et al* [17, 13]).

As the concept of 'now' (present moment) seems to be fundamental to our reasoning about time, it follows that notions such as 'instantaneous', 'simultaneous' and 'immediate' are natural ones to use in specifying temporal properties. As Bergadaà [3] states in his work on a temporal framework:

> *The present time could be made of moments that enable the allocation of time to different activities. They could also be made of duration in which the activity will take the time necessary for its completion.*

From such observations and the literature noted above we distil the following properties that we identify as being of relevance to the modelling of complex real-time systems.

- The dynamics of a system (how quickly things change) are central to understanding its behaviour.
- Systems clearly operate at many different granularities (of time), ie. there are different abstract views of the dynamics of the system.
- It is useful to consider certain actions (events) as atomic and instantaneous (whilst allowing them to have internal state and behaviour at a more detailed level of description).
- It is useful to consider two or more events as occurring simultaneously (instantaneously), or the response to some event being immediate (whilst allowing them to be separated in time at a more detailed level of description).
- The order (but not necessarily the time) at which events occur is important; precedence can give rise to causality.
- The durations of certain actions are important, but the measuring of time must not be overly precise and must allow for tolerance (non-determinacy) in the temporal domain.
- Abstract clocks are useful for relating and coordinating activities, but real clocks are never perfectly reliable or accurate.
- At each level of temporal behaviour it is useful to have access to both continuous and discrete notions of time – controlling actions are typically described using discrete time, the controlled object due to its continuously changing nature often requires dense time for its behavioural description.
- Hierarchical control (cascade control) and hierarchical scheduling (planning) are often observed through the time levels of a system.
- At each level of temporal behaviour similar phenomena are observed (e.g., cyclic/ repetitive actions, deadline-driven actions, synchronous and asynchronous event handling, agreement, coordination, etc.)

Engineers, even of real-time systems, seem to have great difficulty over the use of precise values of time. Why choose an iteration rate of 20ms? – why not 19ms or 21ms? What does a deadline of 15ms actually mean? – would a delay of 10$\mu$s be significant? This difficulty with temporal quantities is not mirrored in the physical domain where tolerances on lengths, weight etc. are commonly expressed.

In the timebands framework, apparently more natural (and essentially atemporal) notions are available such as ‘immediate’, ‘instantaneous’, ‘simultaneous’, ’definitely’ and ‘possible’. And durations are first expressed in general terms - for example “this is a minute-level activity” (ie. it will last a few minutes, rather than hours or seconds). Orders of magnitude between rates of change give an initial decomposition of the system. Indeed the framework uses time itself to separate concerns in any architectural description or system specification.

The central notion in the framework is that of a time band that is defined by a *granularity* (eg. 1 minute) and a *precision* (eg. 5 seconds). Granularity defines the unit of time of the band; precision bounds the actual duration of an event that is deemed to be instantaneous in this band.

A system is assumed to consist not of a single time dimension but a finite set of *bands*. System activities are placed in some band B if they engage in significant events at the time scale represented by B, ie. they have dynamics that give rise to changes that are observable or meaningful in band B’s granularity. So, for example, at the 10 millisecond band, neural circuits are firing, significant computational functions are completing, and an amount of data communication will occur. At the 5 minute band, work shifts are changing, meetings are starting, etc. For any system there will be a highest and lowest band that gives a temporal system boundary – although there will always be the potential for larger and smaller bands. Note that at higher bands the physical system boundary may well be extended to include wider (and slower) entities such as legislative constraints or supply chain changes. To complete this short motivation section the important topics of sampling and rates of change are addressed.

*Sampling.* Our focus is on embedded real-time systems and hence we cannot avoid issues like sampling of inputs, and discretization of continuous quantities. For example, assume two proximity conditions are represented by boolean variables *top* and *bottom*, representing that a controlled gate is at the top or bottom, respectively, of its travel. It is an error for the two proximity sensors to give simultaneous positive inputs. By placing this requirement for error detection in the minute band (with precision of 5 seconds) the following constraints are derived

- If in any interval of duration five seconds, or more, *top* and *bottom* are permanently true then the error condition *must* be identified.
- If in any interval of duration five seconds, or less, *top* and *bottom* are true for part of the interval then the error condition *can* be identified. Note that the intervals over which *top* and *bottom*, respectively, hold don’t have to be the same, or even overlap.

This dual use of *must* and *can* cannot be eliminated. One may move the requirement between time bands to decrease the value of the precision parameter, but even in the lowest band in the system there is an inevitable non-determinacy because true perfectly simultaneous polling of the two sensors is not possible.

*Rate of change.* Even though an environmental entity is subject to continuous change, it does not mean that all such behaviour must be captured at the lowest possible band

– and that this band must have a dense notion of time whilst the others can be discrete. Rather, within any band, many (perhaps most) entities will be discrete, but some may be continuous. So if the purpose of an automatic 'plant watering' system in a greenhouse is to aid the growth of plants in some controlled environment, the rate of growth of the crop per week or day may be significant (but not per second or millisecond).

Consider, for example, the maximum rate of flow of water from a piston-style pump. At a higher time band, this may be stated as $r$ litres per time unit, but at a lower time band, there are two phases of the piston: one filling the cylinder, in which there is almost no flow of water out of the pump; and the other emptying the cylinder, during which the rate of flow of water out of the pump is about twice $r$.

The maximum rate of change of a state variable may be uniform between some pairs of time bands, but not between others. By uniform, we mean that the maximum rates of change are the same (although they will be expressed with respect to the granularity of their respective time bands). For example, with the piston pump the rate of flow is not uniform at the time band that distinguishes the filling and emptying phases of the cylinder, but between a pair of higher bands the rate of flow may be uniform. At a still higher pair of bands, we may be switching the pump on and off to control the rate of flow over a broader time base. Again the rate of flow won't be uniform, but between still higher bands, which don't distinguish the on and off phases, it may emerge to be uniform again.

The motivation for proposing this timeband framework is to simplify the specification of complex systems, improve the dependability of deployed systems and reduce the cost of designing (and redesigning) such systems. It allows dynamic properties to be partitioned but not isolated from each other.

## 3 Definition of the Timeband Model

From the above considerations, a timeband model has been developed[2] that is described in this and the next section (with some illustrative small examples). The aim of a timeband model is to be an essential part of any complete system description. It enables the temporal properties of existing systems to be described and the requirements for new or modified systems to be specified. The informal description of the framework is supported by a formal model expressed in the Z notation [48, 25].

The framework is developed in a number of stages that build up the full model. Some examples of how this model can be extended into a language for actual use in specifying systems is then given. The list of topics discussed in this section are: time bands, granularity and precision, events and classes of events, precedence, simultaneous and immediate, activities, mappings between bands, durations, and clocks. The next section covers state-related aspects of the model: a less determined view of state, change-of-state events, mapping states, accuracy and rates of change.

[2] Initial developments of the framework are described in technical reports [8, 7].

### 3.1 Time bands

For our formal model, we take the set of time bands ($\mathcal{B}$) as a primitive type. Both "$\sqsubseteq$" and "$\sqsubset$" are relations between bands, with "$\sqsubseteq$" forming a partial ordering[3] on time bands. The type of a relation between bands is given as $\mathcal{B} \leftrightarrow \mathcal{B}$. A relation is equivalent to a set of pairs, ie. $(\mathcal{B} \leftrightarrow \mathcal{B}) = \mathbb{P}(\mathcal{B} \times \mathcal{B})$, where $\mathbb{P}\, X$ stands for *power set* of $X$, (i.e., the set of all subsets of $X$).

$$
\begin{array}{|l}
\_ \sqsubseteq \_ : \mathit{partial\_order}[\mathcal{B}] \\
\_ \sqsubset \_ : \mathcal{B} \leftrightarrow \mathcal{B} \\
\hline
\forall\, b1, b2 : \mathcal{B} \bullet (b1 \sqsubset b2 \Leftrightarrow b1 \sqsubseteq b2 \land b1 \neq b2)
\end{array}
$$

For example, we may have that

$$\mathit{MinuteBand} \sqsubset \mathit{DayBand} \sqsubset \mathit{MonthBand}$$

From a focus on some band $B$, adjacent bands $A$ and $C$, where $C \sqsubset B \sqsubset A$, can be identified. Slower (higher or coarser) bands (e.g. $A$) can be taken to be unchanging (essentially constant) for issues of concern to $B$. At the other extreme, behaviours in the faster (lower or finer) bands (e.g. $C$) are assumed to be instantaneous in $B$. The actual differences in granularity between $A$, $B$ and $C$ are not precisely defined (and indeed may depend on the bands themselves) but will typically be in the range 1/10th to 1/100th. When bands map on to hierarchies (structural or control) then activities in band $A$ can be seen to constrain the dynamics of band $B$, whereas those in $C$ enable $B$ to proceed in a timely fashion. The ability to relate behaviour at different time bands is one of the main properties of the framework.

As an example, consider a university lecture course. Here there are immediately four bands to identify. The year band in which new courses and curriculum are planned, the weekly band in which lectures are scheduled, the minute band that allows each lecture to be structured, and the second band that can capture various interactions with the available technical support (eg. laptop response). Whilst giving a lecture, one can assume that the curriculum is stable (unchanging) and that the laptop reacts instantaneously to slide change requests. Systems that don't respect some form of time band structure can become extremely complex and difficult to comprehend, e.g., changing the course syllabus while lecturing is likely to lead to great confusion.

It is important to emphasise that the full behaviour of a system is not obtained by refining down to the lowest band or by projecting emergent behaviours up to the highest band. Rather it is the amalgamation of all band descriptions – all have behaviours that may be needed in any assertion about the system as a whole.

[3] Although in most systems the bands will be totally ordered, there are applications, perhaps in the domain of systems of systems, where this is not the case. For example a *week* band and a *fortnight* band are too similar to seen as hierarchically related. They would however both be strictly ordered with respect to a higher *year* band and a lower *minute* band. For a formal definition of partial order, see Appendix A.

### 3.2 Granularity and precision

For each band its *granularity*, representing the unit of time in that band, and *precision*, representing the measure of accuracy of events within that band. They must both be expressed relative to a lower band. For example, the granularity of the *MonthBand* with respect to the *DayBand* may have a granularity defined as follows:

$$\mathit{Granularity}(\mathit{MonthBand}, \mathit{DayBand}) = \{28, 29, 30, 31\}$$

and the granularity of the *DayBand* with respect to the *MinuteBand* is defined as follows:

$$\mathit{Granularity}(\mathit{DayBand}, \mathit{MinuteBand}) = \{1440\},$$

because there are $24 * 60 = 1440$ minutes in a day. Hence the granularity of the *MonthBand* with respect to the *DayBand* is

$$\mathit{Granularity}(\mathit{MonthBand}, \mathit{MinuteBand}) = \{28 * 1440, 29 * 1440, 30 * 1440, 31 * 1440\}.$$

Note that this set is not contiguous. For ease of presentation we assume that standard units such as minutes, milliseconds, etc, are well defined. However not all time scales will give rise to time bands.

For a band $b1$, its granularity will be defined with respect to all lower bands; hence the domain of the granularity function is all pairs of bands $(b1, b2)$, such that $b2$ is lower than $b1$. If $b1$ is related to a lower band $b2$, and $b2$ to $b3$, then the granularity of $b1$ with respect to $b3$ is the composition of the granularities of $b1$ with respect to $b2$ and $b2$ with respect to $b3$. The set $\mathbb{N}$ is the natural numbers and $\mathbb{N}_1$ is the non-zero natural numbers. *Granularity* is a partial function ($\pfun$) from pairs of time bands to a non-empty set ($\mathbb{P}_1$) of non-zero natural numbers.

$$\begin{array}{|l}
\mathit{Granularity} : (\mathcal{B} \times \mathcal{B}) \pfun \mathbb{P}_1\, \mathbb{N}_1 \\
\hline
\mathrm{dom}(\mathit{Granularity}) = \{b1, b2 : \mathcal{B} \mid b2 \sqsubset b1\} \\
\forall\, b1, b2, b3 : \mathcal{B};\ g : \mathbb{N}_1 \bullet b3 \sqsubset b2 \sqsubset b1 \Rightarrow \\
\quad g \in \mathit{Granularity}(b1, b3) \Leftrightarrow \\
\quad\quad (\exists\, g1, g2 : \mathbb{N}_1 \bullet g = g1 * g2 \wedge \\
\quad\quad\quad g1 \in \mathit{Granularity}(b1, b2) \wedge g2 \in \mathit{Granularity}(b2, b3))
\end{array}$$

Within a band, behaviour is defined using *events* (which are instantaneous), *activities* (that have duration) and *state* (both discrete and continuous). These are defined in later sections, but important here is the property that events are defined to be instantaneous. And two or more events may be defined to be simultaneous. A band's *precision* is a constraint on the duration of 'instantaneous' and 'simultaneous' when measured in a finer band. For example, if the precision of the hour band is defined to be one minute then two simultaneous events must occur within a minute of each other.

Because precision can only be expressed using the granularity of a finer band, it is defined on a pair of (upper and lower) time bands.

$$
\begin{array}{|l}
\mathit{Precision} : (\mathcal{B} \times \mathcal{B}) \pfun \mathbb{N}_1 \\
\hline
\mathrm{dom}(\mathit{Precision}) = \{b1, b2 : \mathcal{B} \mid b2 \sqsubset b1\} \\
\forall\, b1, b2, b3 : \mathcal{B} \bullet b3 \sqsubset b2 \sqsubset b1 \Rightarrow \\
\qquad \mathit{Precision}(b1, b3) \leq \mathit{Precision}(b1, b2) * \mathit{min}(\mathit{Granularity}(b2, b3)) \\
\end{array}
$$

The definition of precision enables the framework to be used effectively for requirements specification. A temporal requirement such as a deadline is band-specific; similarly the definition of a timing failure. For example, being one second late may be a crucial failure in a computing device, whereas on a human scale being one second late for a meeting is meaningless. The duration of an activity is also 'imprecise' (within the band). Stating that a job will take three months is assumed to mean plus or minus a couple of days. Of course the precision of a band can only be explored in a lower band.

Again with the lecturing example, assume the precision of the minute band is one second. The instantaneous 'slide change' event when mapped to a laptop activity in a lower band must have a duration of not more than one second.

A key aspect of the timeband framework is that certain entities are considered to be instantaneous, and that they are then mapped to actions that have duration in a more detailed description of the system. One means of achieving this property would be to give all such entities a distinct precision. However in constructing behaviours from collections of entities, composition is much more straightforward if the same notion of precision applies. Moreover, the property of being 'instantaneous' relates to the level of the temporal abstraction not to the event itself. Hence the timeband framework starts by defining the bands and then places entities into the bands. If the entity is instantaneous it is represented by an event; if it has duration then it is represented by an activity with a duration that is adequately expressed using the granularity of the chosen band. Hence 'adequately' means with sufficient (but not excessive) precision over the value of the defined duration.

As well as the system itself manifesting behaviour at many different time bands, the environment will exhibit dynamic behaviour at different granularities. The bands are therefore linked to the environment at the level determined by these dynamics. In many system abstractions it is useful to assume the environment is in some form of steady state. But this assumption is clearly false as the environment evolves, perhaps as a result of the deployment of the embedded system under development. By mapping the rate of this evolutionary change to an appropriate (relatively slow) time band one can gain the advantage of the steady-state abstraction whilst not ignoring slower dynamics.

### 3.3 Classes of events/activities

In describing the behaviour of a system we often want to refer to repetitive activities/events. We say they are of a particular *class*, e.g., the event class corresponding to a door opening, where the door will be opened many times during the life of the system. Each class of events/activities has a unique time band and name that we use to characterise the class.

$$\begin{array}{|l} \mathcal{C} \\ band : \mathcal{B} \\ name : String \end{array}$$

The above is a Z schema: it defines a record type $\mathcal{C}$ with two fields, *band* and *name*.

### 3.4 Events

By definition, all actions within a band have similar timing dynamics. Within a band, *events* are instantaneous, while *activities* may have a non-zero duration. Events are a natural way of expressing change within a system. By first defining behaviours to be instantaneous, an abstract definition of their cause and effect can be given. Also, seemingly impossible specifications can be given clear semantics. For example, the change-of-state event to turn off a water pump (as used in the case study in Section 7) is an event that ideally is instantaneous at some level of abstraction but clearly must take time at a more detailed level of description (in a finer band).

In a particular behaviour, there may be any (countable) number of instances of events of a particular class, including zero. The set of instances of an event class within a behaviour are totally ordered by precedence (see below), and hence we can also assign a unique index to an event instance. An event instance, "event" for short, is characterised by its class (time band and name), and a natural number index, $n$, indicating that it is occurrence $n$ of events of that class within a behaviour.

$$\begin{array}{|l} \mathcal{E} \\ class : \mathcal{C} \\ index : \mathbb{N} \end{array}$$

We use the notation $c \# i$ to stand for the event of class $c$ that has index number $i$. The idea of indexing event instances comes from RTL [30]. We define a shorthand for the time band of an event.

$$\begin{array}{|l} band : \mathcal{E} \to \mathcal{B} \\ \hline \forall\, e : \mathcal{E} \bullet band(e) = e.class.band \end{array}$$

For a band, $b$, $Events(b)$ defines the set of all events in that band.

$$\begin{array}{|l} Events : \mathcal{B} \to \mathbb{P}\, \mathcal{E} \\ \hline \forall\, b : \mathcal{B} \bullet Events(b) = \{e : \mathcal{E} \mid band(e) = b\} \end{array}$$

### 3.5 Precedence

Although time is of central importance, there are contexts in which pure *order* is a more natural way of describing behaviour [33, 1, 27] (X was before Y, e.g., "before the end of the shift", "after the plane took off", "before the flood", "after the thread

has completed", "before the gate has fired"). The framework must therefore represent both precedence relations and temporal frames of reference.

There is a strong link between temporal order (i.e., time-stamped events and activities) and precedence relations. However, in this framework, we do not impose an equivalence between time and precedence. Due to issues of precision, time cannot be used to infer precedence unless the time interval between two events is sufficiently large in the band of interest.

Where bands are (at least partially) ordered by granularity, then order and hence potential causality is preserved as one moves from the finer to the coarser bands. However, order and hence causality are not necessarily maintained as one moves down through the bands. Where order is important then proof must be obtained by examining the inter-band relationships.

A precedence relation ($\preceq$) defines a partial ordering[4] on the events. Only events in the same time band are related by the precedence relation. We use the operator $\prec$ for strict precedence. A behaviour of a system will consist of a nonempty set of events, *ev*, ordered by precedence. The notation $(\_ \preceq \_)$ stands for the precedence relation taken as a whole.

*BehaviourEvents*

$$
\begin{array}{l}
ev : \mathbb{P}_1\, \mathcal{E} \\
\_ \preceq \_ : partial\_order[\mathcal{E}] \\
\_ \prec \_ : \mathcal{E} \leftrightarrow \mathcal{E} \\
\hline
\forall e, f : \mathcal{E} \bullet \\
\quad (e \preceq f \Rightarrow e \in ev \land f \in ev \land band(e) = band(f)) \land \\
\quad (e \prec f \Leftrightarrow e \preceq f \land e \neq f) \\
\forall c : \mathcal{C};\ i, j : \mathbb{N} \bullet i < j \land c \# j \in ev \Rightarrow c \# i \in ev \land c \# i \prec c \# j
\end{array}
$$

The above Z schema declares a number of fields and constrains them to satisfy the predicate below the line. Note that we don't insist that all pairs of events are related one way or the other, but if both $e \preceq f$ and $f \preceq e$, because "$\preceq$" is a partial order we insist that $e = f$. For each class of events, event instances are sequentially numbered from zero. Hence, if there is an instance of an event with index *j*, then there must be event instances of the same class for all indices less than *j* and these instances must precede the *j*th instance.

### 3.6 Simultaneous and immediate

In the specification of a system, an event may cause a response *immediately* (*instantaneously*) – meaning that at this band the response is within the precision of the band. This use of untimed notions helps eliminate the problem of over specifying requirements that is known to lead to implementation difficulties [29]. For example consider the naturally specified requirement 'when the fridge door opens the light must come

[4] See Appendix A.

on immediately'; this apparently gives no scope for an implementation to incorporate the necessary delays of switches, circuitry and the light's own latency. Making the term 'immediate' band specific, enables a finer-granularity band to include the necessary delays, latencies and processing time that are needed to support the immediate behaviour at the higher band. This separation of concerns removes the need to add a precise deadline to the 'light-on' event. An explicit deadline (of say 8.5ms) is too concrete – rather the deadline is 'the definition of immediate in this band'. Obviously being immediate in the hour band is not the same as being immediate in the microsecond band.

Two events may have a precedence relationship (eg. slide X before slide Y) but occur at the same time (same hour).

It follows from these observations that, in this framework, there is a difference between two events being simultaneous and being 'at the same time'. The former is a much stronger statement. Here two simultaneous events (in band B say) must, when viewed from a finer band, be within the precision of band B. Whereas 'at the same time' only required the two events to occur within the granularity of band B. As the precision is typically 1/10th to 1/100th of the granularity, clearly events being simultaneous is a much tighter constraint.

Precedence gives rise to potential causality. If P is before Q then information could flow between them, indeed P may be the cause of Q. In the use of the framework for specification we will need to use the stronger notion of precedence to imply causality. For example, "when the fridge door opens the light must come on". Within the band of human experience this can be taken to be *immediate* (simultaneous but ordered). At a finer band a number of electro-mechanical activities will be needed to be described that will sense when the door is open and enable power to flow to the light. Importantly, no causality relationship can be inferred (without explicit precedence) for two events occurring at the same time within their particular band. In effect they are logically concurrent and may occur in sequence or overlapped in time when viewed from a lower band.

We introduce a separate relation ($\simeq$) to denote that two events are simultaneous. While $\simeq$ is reflexive and symmetric, it isn't transitive.[5] One event, $f$, immediately follows another, $e$, written $e \trianglelefteq f$, if $f$ both follows $e$ and is simultaneous with $e$. Behaviours are extended to include simultaneous events. This schema includes schema *BehaviourEvents*, and hence includes all the fields of that schema as well as its constraints.

*BehaviourSimultaneous*

$$
\begin{array}{l}
\textit{BehaviourEvents} \\
\_ \simeq \_ : \textit{symmetric\_rel}[\mathcal{E}] \\
\_ \trianglelefteq \_ : \mathcal{E} \leftrightarrow \mathcal{E} \\
\hline
(\forall e, f, g, h : \mathcal{E} \bullet \\
\quad (e \simeq f \Rightarrow e \in ev \land f \in ev \land band(e) = band(f)) \land \\
\quad (e \preceq f \preceq h \land e \preceq g \preceq h \land e \simeq h \Rightarrow f \simeq g) \land \\
\quad (e \trianglelefteq f \Leftrightarrow e \preceq f \land e \simeq f))
\end{array}
$$

[5] See Appendix A for the definition of *symmetric_rel*.

### 3.7 Activities

An activity has a class (time band and name) and an instance number.

$$\begin{array}{|l}\mathcal{A} \\ \hline class : \mathcal{C} \\ instance : \mathbb{N} \\ \hline\end{array}$$

We also use the notation $c \# i$ to refer to the activity with class $c$ and instance $i$. We define a shorthand for the time band of an activity.

$$\begin{array}{|l} band : \mathcal{A} \rightarrow \mathcal{B} \\ \hline \forall a : \mathcal{A} \bullet band(a) = a.class.band \end{array}$$

For a band, $b$, $Activities(b)$ defines the set of all activities in that band.

$$\begin{array}{|l} Activities : \mathcal{B} \rightarrow \mathbb{P}\,\mathcal{A} \\ \hline \forall b : \mathcal{B} \bullet Activities(b) = \{a : \mathcal{A} \mid band(a) = b\} \end{array}$$

An activity has associated with it a nonempty set of events (ie. $\mathbb{P}_1\,\mathcal{E}$), all of which are in the same time band. Every activity, $a$, has associated with it a start event, $\uparrow a$, and possibly an end event, $\downarrow a$. An activity may not have an end event if it never terminates, or if we are only considering a partial trace of behaviour. For an activity of class $c$, the start events of such activities are of class $\uparrow c$ and the end events are of class $\downarrow c$; note that we have overloaded the up and down arrow symbols to function on both classes and activities. The start event of an activity should precede all events in the activity, which should themselves precede the activity's end event. The instance number of an activity is the same as the index number of its start event. Note that if we allow two activities of the same class to overlap, the instance number of an activity and the index number of its end event need not correspond.

$$\begin{array}{|l} BehaviourActivities \\ \hline BehaviourSimultaneous \\ act : \mathbb{P}\,\mathcal{A} \\ events\_of : \mathcal{A} \nrightarrow \mathbb{P}_1\,\mathcal{E} \\ \uparrow : \mathcal{A} \nrightarrow \mathcal{E} \\ \downarrow : \mathcal{A} \nrightarrow \mathcal{E} \\ \hline events\_of \in act \rightarrow \mathbb{P}_1\,ev \\ \mathrm{dom}(\uparrow) = act \land \mathrm{dom}(\downarrow) \subseteq act \\ \forall a : act \bullet \\ \quad events\_of(a) \subseteq Events(band(a)) \land \\ \quad \uparrow a \in events\_of(a) \land (\uparrow a).class = \uparrow(a.class) \land \\ \quad (a \in \mathrm{dom}(\downarrow) \Rightarrow \downarrow a \in events\_of(a) \land (\downarrow a).class = \downarrow(a.class)) \land \\ \quad (\uparrow a).index = a.instance \land \\ \quad (\forall e : events\_of(a) \bullet \uparrow a \preceq e \land (a \in \mathrm{dom}(\downarrow) \Rightarrow e \preceq \downarrow a)) \land \\ \quad (\forall i : \mathbb{N} \bullet i < a.instance \Rightarrow a.class \# i \in act) \end{array}$$

For the lecturing example, viewed at the year time band there may be an activity that corresponds to a course. The events of this activity include a set of lectures, all of which are after the start of the course and before its end. The lecture events may be related by precedence. At this level the precedence may just correspond to the dependence of material in one lecture on that in another, and hence the ordering of the lectures need not be total.

Many activities will have a repetitive cyclic behaviour with either a fixed periodicity or slowly varying pace. Other activities will be event-triggered. Most will have temporal constraints (deadlines). Activities are performed by *agents* (organisational, human or technical). In some bands all agents will be artificial (physical, computational or electrical), at others all human, and at others both will be evident. In addition to agents, there will often be the need for *resources* to enable the agent to make progress.

In this framework definition we will not include agents and resources; rather we concentrate on behaviour (events, activities and state). The scheduling of agents and resources so that activities meet their timing requirements is a natural extension to this description and would make use of standard scheduling and planning techniques.

## 3.8 Mappings between bands

In the components of the framework so far considered, all behaviours have been confined to a single band. In doing so, some notions such as 'instantaneous', 'simultaneous', and 'immediate' have been defined but their semantic properties have not yet been fully defined. To do this, multiple-band behaviours need to be accommodated. This is achieved by mapping events in one band to activities in finer bands.

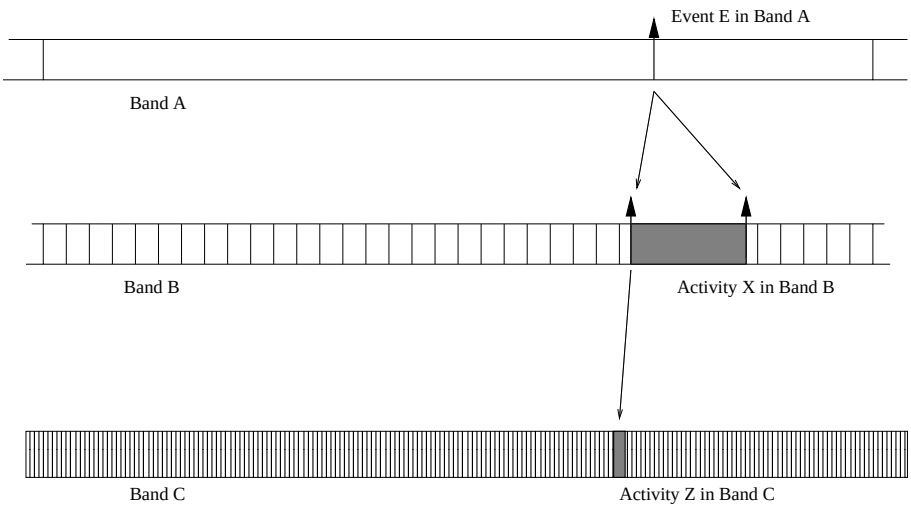


**Fig. 1** Time Band Example

Events that are instantaneous in band A may *map* to activities that have duration at some lower band *B* with a finer granularity. A key property of a band is the *precision* it defines for its time scale. This requires the activity associated with event E (in band A) to have a maximum duration of $\rho$ (the precision of band A - as measured in band B). An illustration of a three band system with the mapping of events to activities is shown in Figure 1. As noted earlier, the start and end of an activity are themselves represented as events.

The link between any two bands is expressed in terms of each band's granularity and precision. Usually the finer of the two bands can be used to express these two measures for the broader band. Where physical time units are used for both bands these relations are straightforward. For example a band with a granularity of an hour and a precision of two minutes is easily linked to a band with a granularity of ten seconds and precision of half a second. The granularity relation is a link from one time unit (1 hour) in the higher band to 360 units in the lower band. The precision of one minute means that a time reference at the higher band (e.g., 3 o'clock) will map down to the lower band to imply a time reference (interval) between 2.59 and 3.01. In general, two bands are said to be *ordered* if the precision of one band is larger then the granularity of the other.

If an event, $e$, maps to an activity, $a$, then that activity has a unique signature event, $sign(a)$, which corresponds to $e$ in the lower band. Behaviours are extended with activities. The mapping preserves the precedence relation between two higher band events $e1$ and $e2$ by requiring that the signature events of their corresponding activities $a1$ and $a1$ in the lower band are similarly related, ie. $sign(a1) \preceq sign(a2)$.

*BehaviourMapping*

$$
\begin{array}{l}
BehaviourActivities \\
\_ \rightsquigarrow \_ : \mathcal{E} \leftrightarrow \mathcal{A} \\
sign : \mathcal{A} \nrightarrow \mathcal{E} \\
\hline
\mathrm{dom}(sign) \subseteq act \\
\forall e : ev;\ a : act \bullet e \rightsquigarrow a \Rightarrow \\
\quad e \in ev \land a \in act \land (band(a) \sqsubset band(e) \land a \in \mathrm{dom}(sign)) \\
\forall e : ev;\ a1, a2 : act \bullet \\
\quad band(a1) = band(a2) \land e \rightsquigarrow a1 \land e \rightsquigarrow a2 \Rightarrow a1 = a2 \\
\forall a : act \bullet a \in \mathrm{dom}(sign) \Rightarrow sign(a) \in events\_of(a) \\
\forall e1, e2 : ev;\ a1, a2 : act \bullet \\
\quad e1 \rightsquigarrow a1 \land e2 \rightsquigarrow a2 \land band(a1) = band(a2) \Rightarrow \\
\qquad (e1 \preceq e2 \Rightarrow sign(a1) \preceq sign(a2))
\end{array}
$$

### 3.9 Durations

The function *duration* gives the time between any two events. To allow for lack of knowledge of the exact time between events and the granularity of the time base, the result of *duration* is a time interval, i.e., a contiguous nonempty set of times, each of

which is represented by a natural number.

$$Interval == \{I : \mathbb{P}_1 \mathbb{N} \mid (\forall t1, t2 : I;\ t : \mathbb{N} \bullet t1 < t < t2 \Rightarrow t \in I)\}$$

An activity that has not terminated (ie. is not in the domain of "$\downarrow$") cannot be given a duration. The duration of a terminating activity is determined from its start and end events.

*BehaviourDurations*

$$\begin{array}{l} BehaviourMapping \\ duration : (\mathcal{E} \times \mathcal{E}) \pfun Interval \\ act\_duration : \mathcal{A} \pfun Interval \\ \hline \mathrm{dom}(duration) = \{e, f : ev \mid e \preceq f\} \\ \forall e, f, g, h : ev \bullet e \preceq f \preceq g \preceq h \Rightarrow \\ \quad (\forall I_1, I_2 : Interval \bullet duration(e, h) = I_1 \land duration(f, g) = I_2 \Rightarrow \\ \qquad I_2 \subseteq I_1) \\ \mathrm{dom}(act\_duration) = \mathrm{dom}(\downarrow) \\ \forall a : \mathrm{dom}(act\_duration) \bullet act\_duration(a) = duration(\uparrow a, \downarrow a) \end{array}$$

Any event in the upper band is mapped to an activity in the lower band whose duration is within the precision of the upper band (with respect to the lower band).

Precision is not only important in defining the bounds on what it means for an event to be instantaneous (in a band), it is also used to constrain what is meant by two events to be simultaneous in some band. If $e$ and $f$ are simultaneous in band $b$ (with precision $\rho$ with respect to the lower band $c$) then the signature events of the mapped activities must occur within $\rho$ in band $c$. Similarly, two events can be defined to be 'not simultaneous' and may require some component of the system to test that this erroneous situation does not occur. Again, by placing such a requirement in the right band, the necessary tolerance on the implementation of the monitoring task is precisely defined.

*BehaviourPrecision*

$$\begin{array}{l} BehaviourDurations \\ \hline \forall e : ev;\ a : act \bullet \\ \quad e \rightsquigarrow a \Rightarrow max(act\_duration(a)) \leq Precision(band(e), band(a)) \\ \forall e1, e2 : ev;\ a1, a2 : act \bullet e1 \simeq e2 \land \\ \quad e1 \rightsquigarrow a1 \land e2 \rightsquigarrow a2 \land band(a1) = band(a2) \Rightarrow \\ \quad max(duration(sign(a1), sign(a2))) \leq Precision(band(e1), band(a1)) \end{array}$$

## 3.10 Clocks

For the time bands associated with computational activity, there is usually a strong notion of time and (adequately accurate) physical clocks that will aid scheduling and coordination. This is also increasingly the case with the bands of human experience

as external sources of time and temporal triggers abound [34]. So measures such as second, minute, hour, day, week, month, year, decade and century are now universal. But other time scales such as ‘generation’, ‘era’ and ‘age’ are also used in specific domains. In a different context the granularity of a band may relate to a physical property of the system, such as the rotation of the crank shaft for an engine control unit.

A frame of reference defines an abstract clock that counts *ticks* of the band’s granularity and can be used to give a time stamp to events and activities. A band may have more than one such abstract clock but they progress at the same rate. For example the day band will have a different abstract clock in each distinct geographical time zone.

We develop a consistent model of time by representing certain moments in the dynamics of a band as “clock tick” events, which are modeled just like any other event. When necessary, an event can be situated in absolute time (within the context of a defined band and clock) by stating a precedence relationship between the event and one or more clock ticks. So an event occurred between 2.00 and 3.00 (in the hour band) if the event occurred after the start of hour from 2.00 to 3.00 but before the end of that hour. Note this is different to saying the event occurred ‘at 2.00’. Here the implication is that it is simultaneous with the 2.00 event. So ‘I will arrive at 2.00’ is satisfied by arriving sufficiently close to the 2.00 event (within the precision of the hour band). However ‘I’ll will arrive by 3.00’ is quite different and allows the arrival event to occur anytime up to the 3.00 event.

A clock can be modeled as a sequence of clock-tick events of a given class, and hence a given time band. Successive clock-tick events are separated by one time unit in the granularity of the band. They are therefore never simultaneous.

## 4 State

In modelling state within the timeband framework there are a number of issues we need to take into account:

observations: within a particular time band, only a subset of the state variables (observations) will be relevant,

nondeterminism: at the time interval corresponding to an event within a band, there may be a set of possible values of a state variable,

change-of-state events: for discrete state observations, changes in value correspond to change-of-state events,

accuracy: for a continuous state observation, there will be a maximum change over a time interval corresponding to the precision of the band, and

rate-of-change: for continuous observations, there will be a maximum rate of change over an interval of size the granularity of the time band.

The observation variables of different time bands may be different. Typically, the representation becomes richer as one moves down to a lower time band with a finer granularity. We say that the state of the system is *projected* onto a band; in some bands not all possible observation variables will be accessible (as the time spent in that state is too short). To illustrate, consider an automatic door:

- at a high time band one can view the door as either *open* or *closed*, with "instantaneous" events to open or close it;
- at a lower time band the open and close events take time, and there are new activities *opening* and *closing*
- at a lower level still one may model how far open the door is by a percentage between 0% open (i.e., closed) and 100% open; this numeric measurement may either be discrete, with some granularity, or continuous; if it is discrete then, at a still lower time band, it may be discrete with a finer granularity.

This can be modeled by having different observation variables at different time bands.

### 4.1 States

The state space can be represented by a mapping from variable names, taken from the set $V$, to values, taken from the set $X$.

$$State[V, X] == V \pfun X$$

The above definition of *State* is generic in the sets of variables and values, for example, the instantiation $State[\{x, y\}, \{0, 1\}]$ represents states with variable names from the set $\{x, y\}$ and values from the set $\{0, 1\}$. A state, $\sigma \in V \pfun X$, maps each variable name in its domain to its value in that state. For simplicity we use the universal set $X$ for all values, rather than each variable having values of a particular type. The set of observation variables in a particular time band is fixed, and hence it is useful to refer to sets of states, all of which have the same variables (i.e., domain).

$$StateSet[V, X] == \{ss : \mathbb{P}\, State[V, X] \mid (\forall \sigma_1, \sigma_2 : ss \bullet \mathrm{dom}(\sigma_1) = \mathrm{dom}(\sigma_2))\}$$

For example, if $x$ and $y$ are variables and integers are values then $s$ is a state and $ss$ is a state set.

$$s = \{x \mapsto 0, y \mapsto 1\}$$
$$ss = \{\{x \mapsto 0, y \mapsto 0\}, \{x \mapsto 1, y \mapsto 0\}, \{x \mapsto 1, y \mapsto 1\}\}$$

The "sets of states" view is adequate for a single sequential process controlling all the variables in the state, but if there are concurrent processes or an externally evolving environment, observation of the state at a particular time precision may observe one variable at one instant and another at a slightly different instant. Hence, if we don't determine the order of observation of the variables, there is a set of values that we can observe for each variable at that time "instant". This leads to a less determined representation of the states, in which each variable is mapped to a set of possible values.

### 4.2 Values views of the state

Over the time interval corresponding to an event, $e$, within a particular time band, one can extract the set, $ss$, of actual states that occur in that interval. Unfortunately,

the set of states view doesn't reflect the reality of observing multiple variables, all of which are evolving over time. For example, if we have two variables $x$ and $y$ which are both initially zero, and if in quick succession $x$ changes to one and then $y$ changes to one, then there is no point at which the state has $x$ with value zero and $y$ with value one. The set of states for this transition is $ss$, above. However, a program sampling the two variables may first sample $x$ and get zero and then sample $y$ and get one, ie. obtain a state $\{x \mapsto 0, y \mapsto 1\}$, which is not in $ss$.

To address this issue we introduce a less determined representation of a set of states, which for each variable records the nonempty[6] set of values it has accumulated over all the states. This has less information than the equivalent set of states.

$$VState[V, X] == V \nrightarrow \mathbb{P}_1 X$$

Note that a sets of values view, or *values view* for short, is a form of state with values replaced by sets of values:

$$VState[V, X] = State[V, \mathbb{P}_1 X]$$

As with states, we define the sets of values views, all of which have the same variables.

$$VStateSet[V, X] == StateSet[V, \mathbb{P}_1 X]$$

### 4.3 Relating a set of states to a values view

The set of states $ss$ above corresponds to the values view $sv$.

$$sv = \{x \mapsto \{0, 1\}, y \mapsto \{0, 1\}\}$$

We define a function *values* to represent this relationship so that $values(ss) = sv$, where the notation $\{\sigma : ss \bullet \sigma(v)\}$ stands for the set of all values of $\sigma(v)$ for $\sigma$ in $ss$.[7]

$$\begin{array}{|l} values : StateSet[V, X] \rightarrow VState[V, X] \\ \hline \forall\, ss : StateSet[V, X] \bullet \\ \quad \textbf{let}\ vars == \{v : V \mid (\exists\, \sigma : ss \bullet v \in \mathrm{dom}(\sigma))\} \bullet \\ \quad\quad values(ss) = (\lambda\, v : vars \bullet \{\sigma : ss \bullet \sigma(v)\}) \\ \hline \end{array}$$

In the opposite direction, the set of states that may be *apparent* in a values view of the state can be extracted by considering all possible states such that each variable maps to an element of its set of possible values. For the values view $sv$, the corresponding set of apparent states is

$$\begin{aligned} apparent(sv) = \{&\{x \mapsto 0, y \mapsto 0\}, \{x \mapsto 1, y \mapsto 1\}, \\ &\{x \mapsto 0, y \mapsto 1\}, \{x \mapsto 1, y \mapsto 0\}\}. \end{aligned}$$

[6] Hence $\mathbb{P}_1$ rather than $\mathbb{P}$.

[7] This is more commonly written $\{\sigma(v) \mid \sigma \in ss\}$ but Z notation makes the fact that $\sigma$ is a bound variable explicit, and hence avoids the possible ambiguity in the commonly used syntax.

The function *apparent* is defined as follows.

$$
\begin{array}{|l}
\hline
[V,X] \\
\hline
apparent : VState[V,X] \rightarrow StateSet[V,X] \\
\hline
\forall\, sv : VState[V,X] \bullet \\
\quad apparent(sv) = \{\sigma : \mathrm{dom}(sv) \rightarrow X \mid (\forall\, v : \mathrm{dom}(sv) \bullet \sigma(v) \in sv(v))\} \\
\hline
\end{array}
$$

We have two properties that relate *apparent* and *values*.

**Theorem 1** *For all values views, sv,*

$$sv = values(apparent(sv)) \tag{1}$$

*and for all sets of states, ss,*

$$ss \subseteq apparent(values(ss)) \tag{2}$$

For example,

$$
\begin{aligned}
ss &= \{\{x \mapsto 0, y \mapsto 0\}, \{x \mapsto 1, y \mapsto 0\}, \{x \mapsto 1, y \mapsto 1\}\} \\
sv &= values(ss) \\
&= \{x \mapsto \{0,1\}, y \mapsto \{0,1\}\} \\
apparent(sv) &= \{\{x \mapsto 0, y \mapsto 0\}, \{x \mapsto 1, y \mapsto 1\}, \\
&\qquad \{x \mapsto 0, y \mapsto 1\}, \{x \mapsto 1, y \mapsto 0\}\} \\
&\supset ss \\
values(apparent(sv)) &= \{x \mapsto \{0,1\}, y \mapsto \{0,1\}\} \\
&= sv
\end{aligned}
$$

Hence a set of states has potentially finer information than the corresponding values view.

### 4.4 Behaviour with state

Each time band has associated with it a set of variables that are observable in that band. Within a behaviour, $ev\_val(e)$, returns the values view (over the observables of its band) that coincides with event $e$. For a discrete state variable, there is often a unique value, but if the event occurs close in time to a change of state then multiple values are possible to reflect our lack of knowledge of the actual value. For example, if an event is simultaneous with an hour band clock *clk* striking 12 then *ev_val* will return $\{clk \mapsto \{11, 12\}\}$. If the value of a continuous state variable is changing at the time of the event, then there is a range of values of the variable.

The set of values views over an interval between (but not including) two events, $e1$ and $e2$, in the same time band is given by $interval\_val(e1, e2)$. Behaviours are extended with state.

*BehaviourState*

$$
\begin{array}{l}
\textit{BehaviourPrecision} \\
\textit{observables} : \mathcal{B} \rightarrow \mathbb{P}\, V \\
\textit{ev\_val} : \mathcal{E} \nrightarrow \textit{VState}[V, X] \\
\textit{interval\_val} : \mathcal{E} \times \mathcal{E} \nrightarrow \textit{VStateSet}[V, X] \\
\hline
\mathrm{dom}(\textit{ev\_val}) = \textit{ev} \\
\mathrm{dom}(\textit{interval\_val}) = \{e1, e2 : \textit{ev} \mid \textit{band}(e1) = \textit{band}(e2)\} \\
\forall\, e : \textit{ev} \bullet \mathrm{dom}(\textit{ev\_val}(e)) = \textit{observables}(\textit{band}(e)) \\
\forall\, e1, e2 : \textit{ev} \bullet (e1, e2) \in \mathrm{dom}(\textit{interval\_val}) \Rightarrow \\
\quad (\forall\, \sigma : \textit{interval\_val}(e1, e2) \bullet \mathrm{dom}(\sigma) = \textit{observables}(\textit{band}(e1))) \\
\forall\, e1, e2, e3 : \textit{ev} \bullet e1 \prec e2 \prec e3 \Rightarrow \textit{ev\_val}(e2) \in \textit{interval\_val}(e1, e3) \\
\forall\, e1, e2 : \textit{ev} \bullet e1 \preceq e2 \Rightarrow \textit{interval\_val}(e2, e1) = \{\} \\
\forall\, e1, e2, e3, e4 : \textit{ev} \bullet e1 \preceq e2 \preceq e3 \preceq e4 \Rightarrow \\
\quad \textit{interval\_val}(e2, e3) \subseteq \textit{interval\_val}(e1, e4) \\
\forall\, e1, e2 : \textit{ev} \bullet e1 \prec e2 \Rightarrow \\
\quad (\forall\, e : \{e1, e2\} \bullet \\
\qquad (\exists\, \sigma : \textit{interval\_val}(e1, e2) \bullet \\
\qquad\quad (\forall\, v : \mathrm{dom}(\sigma) \bullet \textit{ev\_val}(e)(v) \cap \sigma(v) \neq \{\})))
\end{array}
$$

The values view associated with any event occurring within an interval must be in the values views of the interval. If two events $e2$ and $e3$ are surrounded by events $e1$ and $e4$, the values views of the interval between $e2$ and $e3$ must be contained in those of the interval between $e1$ and $e4$. For a nonempty interval the states corresponding to the end-point events "overlap" in values with some state within the interval.

The set of values views over an activity, $a$, is given by $\textit{act\_val}(a)$, which includes all the states between the start and end events of the activity, including at the start and end events.

*BehaviourStateActivities*

$$
\begin{array}{l}
\textit{BehaviourState} \\
\textit{act\_val} : \mathcal{A} \nrightarrow \textit{VStateSet}[V, X] \\
\hline
\mathrm{dom}(\textit{act\_val}) = \textit{act} \\
\forall\, a : \textit{act} \bullet \\
\quad (\forall\, \sigma : \textit{act\_val}(a) \bullet \mathrm{dom}(\sigma) = \textit{observables}(\textit{band}(a))) \wedge \\
\quad (\forall\, e1, e2 : \textit{events\_of}(a) \bullet \\
\qquad \textit{ev\_val}(e1) \in \textit{act\_val}(a) \wedge \\
\qquad \textit{interval\_val}(e1, e2) \subseteq \textit{act\_val}(a))
\end{array}
$$

## 4.5 Change of state events

For discrete state, changes in value can be modeled by events. We can represent a change of state event in which variable $v$ takes on the new value $x$, by the syntax $(v := x)$ for this class of event. A variable, $v$ is constant between state change events for $v$.

*BehaviourStateChange*

*BehaviourStateActivities*

$$\forall e_1 : ev;\ v : V;\ x : X \bullet e_1.class = (v := x) \Rightarrow$$
$$x \in ev\_val(e_1)(v) \wedge$$
$$(\forall e_2 : ev \bullet e_1 \prec e_2 \wedge$$
$$\neg\ (\exists e : ev;\ y : X \bullet e_1 \prec e \prec e_2 \wedge e.class = (v := y)) \Rightarrow$$
$$(\forall \sigma : interval\_val(e_1, e_2) \bullet \sigma(v) = \{x\}))$$

## 4.6 Mapping states

If an event $e$ is mapped to an activity $a$ in a lower band, then the values view at the event in the higher band corresponds to the union of the state values for the activity in the lower band.

*BehaviourStateMapping*

*BehaviourStateChange*

$$\forall e : ev;\ a : act \bullet e \rightsquigarrow a \Rightarrow$$
$$(\forall v : observables(band(e)) \cap observables(band(a)) \bullet$$
$$ev\_val(e)(v) = \{x : X \mid (\exists sv : act\_val(a) \bullet x \in sv(v))\})$$

## 4.7 Accuracy and rates of change

As discussed above, within a single band a numeric-valued variable may have an accuracy and a maximum rate of change. Its accuracy is the maximum amount it can change over a period of size the precision of the band, and its maximum rate of change is the maximum amount it can change over a period of size the granularity of the band.

Within a particular time band, the rate of change of a state variable can be viewed as the change in its value over a time unit within the band. We'll illustrate this by discussing the maximum rate of change of a state variable, $v$.

- Within a given time band the maximum rate of change of $v$ may be $r$, i.e., $v$ can change by at most $r$ over one time unit in that band.
- For a lower time band to be consistent with the upper band, the sum of the changes over a sequence of time units within the lower time band, with length corresponding to the granularity of the upper band with respect to the lower band, must be at most $r$.

*BehaviourRates*

$$
\begin{array}{l}
\mathit{BehaviourStateMapping} \\
\mathit{accuracy} : \mathcal{B} \rightarrow (V \nrightarrow \mathbb{R}) \\
\mathit{rate} : \mathcal{B} \rightarrow (V \nrightarrow \mathbb{R}) \\
\hline
\forall\, b : \mathcal{B} \bullet \textbf{let}\, \mathit{vars} == \mathrm{dom}(\mathit{accuracy}(b)) \bullet \\
\quad \mathit{vars} = \mathrm{dom}(\mathit{rate}(b)) \land \mathit{vars} \subseteq \mathit{observables}(b) \land \\
\quad (\forall\, e : \mathit{ev};\ v : \mathit{vars} \bullet \mathit{band}(e) = b \Rightarrow \\
\qquad (\forall\, x, y : \mathit{ev\_val}(e)(v) \bullet \mathit{abs}(x - y) \leq \mathit{accuracy}(b)(v))) \land \\
\quad (\forall\, e1, e2 : \mathit{ev};\ v : \mathit{vars} \bullet e1 \prec e2 \land \mathit{band}(e1) = b \land \mathit{band}(e2) = b \Rightarrow \\
\qquad (\forall\, \sigma_1, \sigma_2 : \mathit{interval\_val}(e1, e2) \bullet (\forall\, x : \sigma_1(v);\ y : \sigma_2(v) \bullet \\
\qquad\quad \mathit{abs}(x - y) \leq \mathit{rate}(b)(v) * \mathit{max}(\mathit{duration}(e1, e2))))) \\
\forall\, e : \mathit{ev};\ a : \mathit{act} \bullet e \rightsquigarrow a \Rightarrow \\
\quad (\forall\, v : \mathrm{dom}(\mathit{accuracy}(\mathit{band}(e))) \cap \mathrm{dom}(\mathit{rate}(\mathit{band}(a))) \bullet \\
\qquad \mathit{rate}(\mathit{band}(a))(v) * \mathit{max}(\mathit{act\_duration}(a)) \leq \mathit{accuracy}(\mathit{band}(e))(v))
\end{array}
$$

Satisfying the consistency condition has a special case if we consider the state variable to be uniform between two levels, or *uniform* for short. If the granularity of the upper time band with respect to the lower time band is $n$, then for a uniform state variable the maximum rate of change in the lower time band will be at most $r/n$. At a still lower time band with granularity $m$ with respect to the above lower band (and hence granularity $m * n$ with respect to the upper time band) the maximum rate of change in this still lower time band will be at most $r/(m * n)$. With a uniform state variable, as the size of the time unit of the band approaches zero the rate of change approaches the derivative of the state variable with respect to time.

## 5 Summary

Rather than have a single notion of time, the proposed framework allows a number of distinct time bands to be used in the specification or description of a system. System behaviours are always relative to (defined within) a band.

The above discussion has defined the timeband framework and introduced a number of key notions that are central to the framework. Here we summarize these ideas:

- *band* – a subset of system behaviours (discrete and continuous) with similar temporal properties;
- *system* – a partially ordered set of bands;
- *separation* – the property of being able to assume that activities in lower (quicker) bands are instantaneous and the state of higher (slower) bands is unchanging;
- *granularity* – the unit of time defined by a band;
- *precision* – the constraint on instantaneous behaviour within a band;
- *event* – an instantaneous action within a band;
- *activity* – an action with duration within a band;
- *duration* – a time interval between events;
- *clock* – an abstract band-specific clock that produces ticks (events) at the granularity of the band;

- *precedence* – one event happening after another event;
- *simultaneous* – two events occurring at the same instant;
- *immediate* – a precedence relation between two simultaneous events;
- *mapping* – a link between an event in one band to an activity in a lower band;
- *state* – the observations available within a time band;
- *set of values view* – the observations over the period of an event;
- *change-of-state events* – for discrete observations;
- *accuracy* – maximum "instantaneous" change in a continuous variable;
- *rate-of-change* – maximum rate of change of a continuous variable over a unit of time.

## 6 Towards a Language for Timebands

Having presented a model for the timeband framework it is then necessary to define a language that can be used to specify system requirements and behaviour. Such a language is derived from the abstract model. In this paper we do not attempt to provide a single, or even a complete, timeband language. Rather we illustrate features that such a language could usefully contain. These are used in a short example of the use of time bands in the Section 7.

### 6.1 Predicates

We represent a predicate over a state space, $\Sigma$, via the subset of states in $\Sigma$ that satisfy the predicate.

**Definition 1 (Predicate)** For a state space $\Sigma$, a predicate is represented by a set of states.

$$Pred[\Sigma] == \mathbb{P}\,\Sigma$$

We use the conventional notations, "$\wedge$", "$\vee$", and "$\neg$ " instead of intersection, union, and complement of sets (with respect to the state space $\Sigma$), respectively, when dealing with predicates. As usual, the unary operators have higher precedence than the binary operators. Point-wise implication, denoted $p \Rightarrow q$, is defined as $\neg\, p \vee q$, and point-wise equivalence is denoted by $p \Leftrightarrow q$. Universal implication, denoted $p \Rrightarrow q$, is defined as $(\forall \sigma \bullet \sigma \in p \Rightarrow \sigma \in q)$, or more succinctly as $p \subseteq q$. Universal equivalence is denoted $p \equiv q$.

### 6.2 Predicates on sets of states

Given a state predicate, $p$, there are two obvious ways to promote it to a set of states (as in modal logics [28]). If $p$ holds for *all states* in the set, we write $\boxplus p$, and if $p$ holds for *some state* in the set, we write $\boxdot p$.

**Definition 2 (All states and some states)**

$$
\begin{array}{|l}
\hline
[\Sigma] \\
\circledast : Pred[\Sigma] \to Pred[\mathbb{P}_1\ \Sigma] \\
\boxdot : Pred[\Sigma] \to Pred[\mathbb{P}_1\ \Sigma] \\
\hline
\forall p : Pred[\Sigma];\ ss : \mathbb{P}_1\ \Sigma \bullet \\
\quad (ss \in (\circledast p) \Leftrightarrow (\forall \sigma : ss \bullet \sigma \in p)) \land \\
\quad (ss \in (\boxdot p) \Leftrightarrow (\exists \sigma : ss \bullet \sigma \in p)) \\
\hline
\end{array}
$$

We promote the boolean operators to predicates on sets of states in the obvious way (because they are defined as predicates, but over sets of states rather than states). We have the following properties of "all states" and "some state" when combined with logical operators.

$$\neg\ \circledast p \equiv \boxdot(\neg\ p) \tag{3}$$
$$\circledast p \Rrightarrow \boxdot p \tag{4}$$
$$\circledast p \land \circledast q \equiv \circledast(p \land q) \tag{5}$$
$$\boxdot p \lor \boxdot q \equiv \boxdot(p \lor q) \tag{6}$$
$$\circledast p \lor \circledast q \Rrightarrow \circledast(p \lor q) \tag{7}$$
$$\boxdot(p \land q) \Rrightarrow \boxdot p \land \boxdot q \tag{8}$$

Note that property (4) is valid because the sets of states must be non-empty.

**Theorem 2** *Given state predicates p and q,*

$$(\boxdot p \Rightarrow \circledast q) \Rrightarrow (\circledast(p \Rightarrow q)) \tag{9}$$

*Proof.*

$\boxdot p \Rightarrow \circledast q$
$\equiv$ by the definition of implication
$\neg\ \boxdot p \lor \circledast q$
$\equiv$ by (3)
$\circledast \neg\ p \lor \circledast q$
$\Rrightarrow$ by (7)
$\circledast(\neg\ p \lor q)$
$\equiv$ by the definition of implication
$\circledast(p \Rightarrow q)$

□

### 6.3 Predicates on values views

We refer to a predicate on a values view of the state as a *values predicate*.

**Definition 3 (Values predicate)**

$$VPred[V,X] == Pred[VState[V,X]]$$

We promote a predicate, $p$, on a single state, to a predicate on a values view in two ways. If $p$ holds for all apparent states (see Section 4.3) derivable from the values view, $sv$, we say $p$ *definitely* holds for $sv$, written $sv \in \circledast p$, and if $p$ holds for at least one apparent state derivable from $sv$, we say $p$ *possibly* holds for $sv$, written $sv \in \odot p$.

**Definition 4 (Definitely and possibly)**

$$\begin{array}{l} [V,X] \\ \hline \circledast : Pred[State[V,X]] \to VPred[V,X] \\ \odot : Pred[State[V,X]] \to VPred[V,X] \\ \hline \forall p : Pred[State[V,X]];\ sv : VState[V,X] \bullet \\ \quad (sv \in (\circledast p) \Leftrightarrow (\forall \sigma : apparent(sv) \bullet \sigma \in p)) \wedge \\ \quad (sv \in (\odot p) \Leftrightarrow (\exists \sigma : apparent(sv) \bullet \sigma \in p)) \end{array}$$

We promote the boolean operators to values predicates in the obvious way because values predicates are predicates, but over values views rather than states.

In the example considered in Section 7, if the methane in a coal mine shaft is ever over a critical level, then to avoid causing an explosion, the pump extracting water from the mine must be off. We can formalise this property by the following values predicate.

$$\odot(methane \geq Critical) \Rightarrow \circledast(pump = Off)$$

If the methane is possibly critical at some instant (i.e., the values view includes an apparent state in which the methane is critical), then the pump is definitely off (i.e., it is off for all apparent states).

From Definitions 2 and 4, the "definitely" and "possibly" operators are related to "all states" and "some state" as follows for all values views, $sv$.

$$sv \in \circledast p \Leftrightarrow apparent(sv) \in \boxplus p \tag{10}$$

$$sv \in \odot p \Leftrightarrow apparent(sv) \in \boxdot p \tag{11}$$

Hence, we have the following properties directly derivable from the properties of predicates on sets of states (3)–(8).

$$\neg\ \circledast p \equiv \odot(\neg\, p) \tag{12}$$

$$\circledast p \Rrightarrow \odot p \tag{13}$$

$$\circledast p \wedge \circledast q \equiv \circledast(p \wedge q) \tag{14}$$

$$\odot p \vee \odot q \equiv \odot(p \vee q) \tag{15}$$

$$\circledast p \vee \circledast q \Rrightarrow \circledast(p \vee q) \tag{16}$$

$$\odot(p \wedge q) \Rrightarrow \odot p \wedge \odot q \tag{17}$$

There are two interesting properties of definitely ($\circledast$) and possibly ($\odot$) that don't hold for "all states" ($\boxtimes$) and "some states" ($\boxdot$).

**Theorem 3** *If the free variables occurring in the predicates p and q are disjoint, then*

$$\odot p \wedge \odot q \equiv \odot(p \wedge q) \tag{18}$$
$$\circledast(p \vee q) \equiv \circledast p \vee \circledast q \tag{19}$$

If the set of free variables occurring in the predicate $p$ is $w$, and $\sigma_1$ and $\sigma_2$ are two states that agree on all the variables in $w$, (i.e., $w \lhd \sigma_1 = w \lhd \sigma_2$, where $w \lhd \sigma$ is the state $\sigma$ restricted to just those variables in the set $w$), then $\sigma_1 \in p \Leftrightarrow \sigma_2 \in p$.

*Proof.* We focus on property (18) because (19) can be derived from it using $\circledast p = \neg\ \odot \neg\ p$. The reverse implication is property (17) above. In the forward direction, if we assume $sv \in \odot p$ and $sv \in \odot q$, then $(\exists \sigma : \mathit{apparent}(sv) \bullet \sigma \in p)$ and $(\exists \sigma : \mathit{apparent}(sv) \bullet \sigma \in q)$. Hence let $\sigma_1 \in p \cap \mathit{apparent}(sv)$, and $\sigma_2 \in q \cap \mathit{apparent}(sv)$. If $w$ is the set of free variables occurring in $p$, we let $\sigma = (w \lhd \sigma_1) \cup (w \lhd\!\!\!\!- \sigma_2)$, where $w \lhd \sigma$ is the state $\sigma$ restricted to just the variables in $w$ and $w \lhd\!\!\!\!- \sigma$ is $\sigma$ restricted to the variables not in $w$. Because $p$ depends only on variables in $w$ and $w \lhd \sigma = w \lhd \sigma_1$, it follows that $\sigma \in p$. Similarly, because $q$ only depends on variables not in $w$, $\sigma \in q$. Finally, because both $\sigma_1$ and $\sigma_2$ are in $\mathit{apparent}(sv)$, $\sigma \in \mathit{apparent}(sv)$. Hence $(\exists \sigma : \mathit{apparent}(sv) \bullet \sigma \in p \wedge q)$, i.e., $sv \in \odot(p \wedge q)$. □

Note that (18) holds for $\odot$ but not $\boxdot$. For example, if

$$ss = \{\{x \mapsto 0, y \mapsto 0\}, \{x \mapsto 1, y \mapsto 0\}, \{x \mapsto 1, y \mapsto 1\}\}$$

we have $ss \in \boxdot(x = 0) \wedge \boxdot(y = 1)$ but not $ss \in \boxdot(x = 0 \wedge y = 1)$.

Using Theorem 3 we can show the following theorem.

**Theorem 4** *Provided the free variables of p and q are disjoint,*

$$\circledast(p \Rightarrow q) \equiv (\odot p \Rightarrow \circledast q)$$

*Proof.*

$\circledast(p \Rightarrow q)$
$\equiv \circledast(\neg\ p \vee q)$
$\equiv$ Theorem 3; free variables in the two disjuncts are disjoint
$\circledast(\neg\ p) \vee \circledast q$
$\equiv \neg\ \odot p \vee \circledast q$
$\equiv \odot p \Rightarrow \circledast q$

□

### 6.4 On the relationship between reality and observation

For each event there is an interval over which the event occurs and a set of actual states, $ss$, over that interval. The corresponding values view is $values(ss)$. From property (2), i.e., $ss \subseteq apparent(values(ss))$, if we want to show a property, $p$, holds for all actual states in $ss$, it is sufficient to show the stronger property that $p$ holds for all states in $apparent(values(ss))$, i.e., $\circledast\, p$ holds for $values(ss)$. Similarly, if we know a property $p$ holds for some actual state in $ss$, we can deduce $\odot\, p$ for $values(ss)$. These relationships are captured by the following theorem.

**Theorem 5** *For any set of actual states, ss,*

$$values(ss) \in \circledast\, p \Rightarrow ss \in \boxtimes p \tag{20}$$

$$ss \in \boxdot p \Rightarrow values(ss) \in \odot\, p \tag{21}$$

Consider the simple case in which we are only dealing with one free variable, $x$, in a predicate, e.g., the predicate is of the form $\circledast(x \in S)$ or $\odot(x \in S)$, where $S$ is constant over the observation interval, then $\circledast(x \in S)$ in the values view is equivalent to $\boxtimes(x \in S)$ in reality, and $\odot(x \in S)$ in the values view is equivalent to $\boxdot(x \in S)$ in reality. Special cases of these predicates are comparisons of a variable with an expression, $C$, that is constant over the observation interval, e.g., $x = C$ or $x < C$.

If one samples a variable, $x$, in the environment, and gets the value $C$, one can deduce $\boxdot(x = C)$, which is equivalent to $\odot(x = C)$. Similarly, by sampling $y$ we may deduce $\boxdot(y = D)$, which is equivalent to $\odot(y = D)$. By Theorem 3 these samples allow one to deduce $\odot(x = C \wedge y = D)$ but not the stronger condition $\boxdot(x = C \wedge y = D)$. This formalises the property that sampling two boolean variables, *top* and *bottom*, introduced in Section 2. Getting two sample values, e.g., *true* and *true*, does not allow one to deduce that the two variables simultaneously have those values (i.e., we can't deduce $\boxdot(top \wedge bottom)$), but we can deduce the weaker property $\odot(top \wedge bottom)$. That is

$$ss \in \boxdot(top \wedge bottom) \Rightarrow values(ss) \in \odot(top \wedge bottom)$$

but not the other way around, in general. Note that $\odot(top \wedge bottom) \equiv \odot\, top \wedge \odot\, bottom$, but we only have $\boxdot(top \wedge bottom) \Rrightarrow \boxdot\, top \wedge \boxdot\, bottom$, in general.

### 6.5 Application to state model

In this section we develop some notation for using values predicates with the values view model. Given a behaviour and an event $e$, $ev\_val(e)$ gives the values view corresponding to event $e$. For a values predicate, $p$, we introduce the notation $p \mathbin{@} e$ to state that $p$ holds for the values view corresponding to $e$, i.e.,

$$p \mathbin{@} e \Leftrightarrow ev\_val(e) \in p$$

For events $e1$ and $e2$, $interval\_val(e1, e2)$ gives the set of values views occurring (strictly) between the two events. We introduce the notation $p$ during $(e1, e2)$ to

stand for $p$ holding for all values views between $e1$ and $e2$, and $p$ within $(e1, e2)$ to state that $p$ holds for some values view between $e1$ and $e2$, i.e.,

$$p \text{ during } (e1, e2) \Leftrightarrow \mathit{interval\_val}(e1, e2) \in \circledast p$$
$$p \text{ within } (e1, e2) \Leftrightarrow \mathit{interval\_val}(e1, e2) \in \boxdot p$$

We also overload these operators so that we can use an activity instead of a pair of events, e.g., $p$ during $a$, with the understanding that the pair of events are the start and end of the activity, e.g., $p$ during $(\uparrow a, \downarrow a)$.

The properties of behaviours on state allow one to deduce properties expressed in terms of these relations. For example, if a property definitely holds at an end point event of a nonempty closed interval then, because the state of the end point "overlaps" with those in the interval, we get the following property for all pairs of events $e1$ and $e2$ where $e1$ precedes $e2$,

$$(\circledast p) \mathbin{@} e1 \Rightarrow (\odot p) \text{ within } (e1, e2) \tag{22}$$

For example, if $\circledast(m \geq C) \mathbin{@} e1$ holds, i.e., for all values of $m$ in the values view corresponding to $e1$ we have $m \geq C$ holding, then because the end of event $e1$ corresponds to the start of the interval between $e1$ and $e2$, we have that $m \geq C$ at the very start of the interval, but note that we don't have any guarantee that $m$ stays above $C$ for any given period—not even the precision of the band—during the interval, and hence we can only deduce $\odot(m \geq C)$ holds within the interval.

We introduce two further shorthand forms to state that a values predicate holds just before and after an event, respectively.

$$p \text{ before } e \Leftrightarrow (\exists e' : \mathit{ev} \bullet e' \preceq e \wedge p \text{ during } (e', e))$$
$$p \text{ after } e \Leftrightarrow (\exists e' : \mathit{ev} \bullet e \preceq e' \wedge p \text{ during } (e, e'))$$

We assume all these relations have higher precedence than logical operators, but lower precedence than the other operators.

From the properties for predicates on sets of states we can deduce properties for during and within, e.g., the following property.

$$\begin{aligned}(p \text{ within } (e1, e2)) \Rrightarrow (q \text{ during } (e1, e2)) \Rrightarrow \\ (p \Rightarrow q) \text{ during } (e1, e2)\end{aligned} \tag{23}$$

This holds because

$$\begin{aligned}
&(p \text{ within } (e1, e2)) \Rightarrow (q \text{ during } (e1, e2)) \\
\equiv\ &\text{letting } \mathit{ssv} = \mathit{interval_val}(e1, e2) \\
&\mathit{ssv} \in \boxdot p \Rightarrow \mathit{ssv} \in \circledast q \\
\equiv\ &\\
&\mathit{ssv} \in (\boxdot p \Rightarrow \circledast q) \\
\Rrightarrow\ &\text{by Theorem 2} \\
&\mathit{ssv} \in \circledast(p \Rightarrow q) \\
\equiv\ &\\
&(p \Rightarrow q) \text{ during } (e1, e2)
\end{aligned}$$

## 7 Example: Mine pump

The mine pump case study has been used by a number of formal frameworks to investigate and illustrate different specification approaches (see [9,35,32] for a number of examples). In its briefest form, the case study involves two subsystems: a methane monitoring subsystem that sounds an alarm if the sensed level of methane is above a threshold, and a pump control subsystem that pumps water from the mine sump if the water level reaches a *High_water* level sensor (the pump then operates until a *Low_water* level sensor is reached). The two subsystems are coupled by the safety requirement not to operate the pump if the methane is high (due to risk of gas explosion). There is also a performance requirement that limits the number of days lost due to flooding to be two or less. In the following partial treatment we concentrate on the methane control subsystem.

We specify the mine pump in terms of rely and guarantee conditions [31], which are similar to preconditions and postconditions, except that rely and guarantee conditions are specified over the interval during which the system is running, rather than in terms of the before and after states for pre-/post-conditions. A *guarantee* condition is a condition that the system should maintain over its operating interval, provided the *rely* conditions hold for that interval.

*Mine pump guarantee.* A pump is used to extract water from a mine shaft. However, if there is a critical level of methane in the mine shaft, an explosion could result if the mine pump is operated. Hence, one requirement is that at all times while the system is running, it should guarantee that the pump is off at any time the methane level is critical.[8] This is represented by the following predicate which must hold for all states of the system while it is operating.

$$\boxtimes(\mathit{methane} \geq \mathit{Critical} \Rightarrow \mathit{pump} = \mathit{Off}) \tag{24}$$

For a guarantee about a system's behaviour, by (20), if one can show the system guarantees $\circledast\, p$ holds for the sampled view of the state, then $\boxtimes\, p$ holds for the corresponding real states. Hence, (24) holds provided the following values predicate holds while the system is operating,

$$\circledast(\mathit{methane} \geq \mathit{Critical} \Rightarrow \mathit{pump} = \mathit{Off}) \text{ during } \mathit{Operation} \tag{25}$$

where *Operation* stands for the activity representing the period the system is operating. The "during" operator is interpreted with respect to a particular time band (in particular, its precision $\rho$). Interestingly, from the point of view of this guarantee, all we require is that there is some time band in which this is satisfied (and this can be determined by the implementation). If we added a requirement ensuring that the pump is on whenever the methane level is safe (i.e., it is below critical by some bound), then this would constrain the choice of implementation time band by limiting the size of its precision.

From Theorem 4 we have that (25) is equivalent to the following.

$$(\odot(\mathit{methane} \geq \mathit{Critical}) \Rightarrow \circledast(\mathit{pump} = \mathit{Off})) \text{ during } \mathit{Operation} \tag{26}$$

[8] In a more detailed analysis, one may want to distinguish between the pump being turned off and it actually having come to a stop.

*Mine pump rely.* In order to implement the requirement, one may rely on a number of properties of the environment. We'll assume that the implementation will be operating within a particular time band, but the properties we rely on can be adapted to a range of possible choices of time bands. Assume the accuracy of the methane level within the band is *Acc_meth*. This represents the maximum amount the methane level can vary over a period of size the precision of the time band. Hence for any trace of the system and any value $Z$

$$(\odot(methane = Z) \Rightarrow \circledast(methane \in Z \pm Acc\_meth)) \text{ during } Operation \tag{27}$$

where $x \pm acc$ is the set of values $\{v \mid x - acc \leq v \leq x + acc\}$. Note that this trace predicate is implicitly using the precision, $\rho$, of the time band to split up the interval into subintervals of size no more than $\rho$, and the predicate has to hold on each of these. From (27) one can deduce

$$(\odot(methane \geq Z) \Rightarrow \circledast(methane \geq Z - Acc\_meth)) \text{ during } Operation \tag{28}$$

Note that this accuracy is only concerned with the timing precision of the band. We should separately consider the accuracy of the methane sampling sensor itself, but for the purposes of this example we assume there is no sampling error.

Assume the maximum rate of change of methane level in the band is *Rate_meth*. This represents the maximum change in the methane over a period of size the granularity of the time band. For an interval of duration $n$, the level of methane can change by at most $n$ times *Rate_meth* over that interval. Hence, for some value, $Z$, if the methane is ever at least $Z + n * Rate\_meth$ within the interval, then it must have been at least $Z$ at the start of the interval. For all pairs of events $e_1$ and $e_2$ within a behaviour such that $max(duration(e_1, e_2)) \leq n$, and for any value $Z$,

$$\begin{array}{l}(\odot(methane \geq Z + n * Rate\_meth) \text{ within } (e_1, e_2)) \Rightarrow \\ \quad (\odot(methane \geq Z) \mathbin{@} e_1)\end{array} \tag{29}$$

*Mine pump implementation.* The implementation samples the methane level at regular intervals. The $i$th sampling event is denoted by $s \# i$. It is assumed that the pump is off from the start of operation until the first sample.

$$\circledast(pump = Off) \text{ during } (\uparrow Operation, s \# 0) \tag{30}$$

We require that the maximum time between samples is $n$ time units in the granularity of the implementation band, i.e., $max(duration(s \# i, s \# i + 1)) \leq n$. A sample event, $s \# i$, corresponds to an activity at a lower time band, which takes place within the precision of the upper time band; the activity contains events to set up analog-to-digital converters to read the methane and water levels, wait until the conversions are complete and read the levels, and turn the pump off if the methane is above a threshold value. In addition, if the methane is below the threshold it turns the pump on or off depending upon the current water level. Here we focus on the properties we need of the sampling events, rather than giving an actual implementation. It is a separate (more straightforward) problem to show an implementation of the sampling events has these properties.

If the methane is definitely above the threshold for sample $i$, the *pump* must be off from immediately after the sample until the start of the next sample:

$$(\circledast(\mathit{methane} \geq \mathit{Threshold}) \mathbin{@} s \# i) \Rightarrow (\circledast(\mathit{pump} = \mathit{Off}) \mathbin{\mathsf{during}} (s \# i, s \# i + 1)) \tag{31}$$

If the pump was already off before sample $i$ then at sample $i$, if the methane is definitely at least the threshold, the pump remains off for the sample.

$$(\circledast(\mathit{pump} = \mathit{Off}) \mathbin{\mathsf{before}} s \# i) \wedge (\circledast(\mathit{methane} \geq \mathit{Threshold}) \mathbin{@} s \# i) \Rightarrow (\circledast(\mathit{pump} = \mathit{Off}) \mathbin{@} s \# i) \tag{32}$$

To allow for the maximum rate of change of the methane level (*Rate_meth*) between the sampling events (which are at most $n$ time units apart) and the inaccuracies of the sampling events (*Acc_meth*) at each end of the interval, we require the following constraint between *Critical* and *Threshold*.

$$\mathit{Critical} \geq \mathit{Threshold} + n * \mathit{Rate\_meth} + 2 * \mathit{Acc\_meth} \tag{33}$$

**Theorem 6** *An implementation that satisfies properties (30), (31), (32), and (33) fulfils the guarantee (26) provided the rely conditions (27) and (29) hold.*

*Proof.* We assume the rely conditions and the conditions specified for the implementation and show that the guarantee condition (26) holds over the operating interval, by showing it holds before the first sample, at every sampling event, and for every interval between one sample and the next. That the guarantee holds before the first sample event follows directly from (30). For the rest we have to show that for all natural numbers $i$

$$(\odot(\mathit{methane} \geq \mathit{Critical}) \Rightarrow \circledast(\mathit{pump} = \mathit{Off})) \mathbin{@} s \# i \tag{34}$$

$$(\odot(\mathit{methane} \geq \mathit{Critical}) \Rightarrow \circledast(\mathit{pump} = \mathit{Off})) \mathbin{\mathsf{during}} (s \# i, s \# i + 1) \tag{35}$$

First we show

$$\odot(\mathit{methane} \geq \mathit{Critical} - \mathit{Acc\_meth}) \mathbin{\mathsf{within}} (s \# i, s \# i + 1) \Rrightarrow \circledast(\mathit{pump} = \mathit{Off}) \mathbin{\mathsf{during}} (s \# i, s \# i + 1) \tag{36}$$

as follows

$\odot(\mathit{methane} \geq \mathit{Critical} - \mathit{Acc\_meth})$ within $(s \# i, s \# i + 1)$
$\Rrightarrow$ from the constraint on the threshold (33)
$\odot(\mathit{methane} \geq \mathit{Threshold} + n * \mathit{Rate\_meth} + \mathit{Acc\_meth})$ within $(s \# i, s \# i + 1)$
$\Rrightarrow$ from the maximum rate-of-change of methane (29)
$\odot(\mathit{methane} \geq \mathit{Threshold} + \mathit{Acc\_meth}) \mathbin{@} s \# i$
$\Rrightarrow$ from methane accuracy (28)
$\circledast(\mathit{methane} \geq \mathit{Threshold}) \mathbin{@} s \# i$
$\Rrightarrow$ as the pump is turned off if methane is high (31)
$\circledast(\mathit{pump} = \mathit{Off})$ during $(s \# i, s \# i + 1)$

To show (34) we first consider the case for sampling events other than sample zero.

$$
\begin{array}{ll}
 & \odot(\mathit{methane} \geq \mathit{Critical}) \mathbin{@} s \# i + 1 \\
\Rrightarrow & \text{from methane accuracy (28)} \\
 & \circledast(\mathit{methane} \geq \mathit{Critical} - \mathit{Acc\_meth}) \mathbin{@} s \# i + 1 \\
\Rrightarrow & \text{by (22)} \\
 & \odot(\mathit{methane} \geq \mathit{Critical} - \mathit{Acc\_meth}) \ \mathsf{within}\ (s \# i, s \# i + 1) \\
\Rrightarrow & \text{from (36)} \\
 & \circledast(\mathit{pump} = \mathit{Off}) \ \mathsf{during}\ (s \# i, s \# i + 1) \\
\Rrightarrow & \text{by the definition of } \mathsf{before} \\
 & \circledast(\mathit{pump} = \mathit{Off}) \ \mathsf{before}\ s \# i + 1 \\
\Rrightarrow & \text{by (32) and } \mathit{Critical} - \mathit{Acc\_meth} \geq \mathit{Threshold} \\
 & \circledast(\mathit{pump} = \mathit{Off}) \mathbin{@} s \# i + 1
\end{array}
$$

For sample zero we have $\circledast(\mathit{pump} = \mathit{Off})\ \mathsf{before}\ s \# 0$ from (30), and hence it is sufficient to apply the equivalent of the last step of the above proof.

To show (35), we first note that, using (23), it is implied by

$$
\odot(\mathit{methane} \geq \mathit{Critical}) \ \mathsf{within}\ (s \# i, s \# i + 1) \Rrightarrow \\
\quad \circledast(\mathit{pump} = \mathit{Off}) \ \mathsf{during}\ (s \# i, s \# i + 1)
$$

which we show as follows.

$$
\begin{array}{ll}
 & \odot(\mathit{methane} \geq \mathit{Critical}) \ \mathsf{within}\ (s \# i, s \# i + 1) \\
\Rrightarrow & \text{from methane accuracy (28)} \\
 & \circledast(\mathit{methane} \geq \mathit{Critical} - \mathit{Acc\_meth}) \ \mathsf{within}\ (s \# i, s \# i + 1) \\
\Rrightarrow & \text{as } \circledast p \Rightarrow \odot p \\
 & \odot(\mathit{methane} \geq \mathit{Critical} - \mathit{Acc\_meth}) \ \mathsf{within}\ (s \# i, s \# i + 1) \\
\Rrightarrow & \text{using (36)} \\
 & \circledast(\mathit{pump} = \mathit{Off}) \ \mathsf{during}\ (s \# i, s \# i + 1)
\end{array}
$$

□

The above reasoning has been generic with respect to the choice of time band, in particular, the precision and granularity of the time band and the choice of the sampling interval $n$. From (33) we have

$$\mathit{Critical} - \mathit{Threshold} \geq n * \mathit{Rate\_meth} + 2 * \mathit{Acc\_meth}.$$

The larger the gap between *Critical* and *Threshold* the less time the pump will be active when it is safe to be active. The predominant factor is the choice of the sampling interval, here represented by $n$ times the granularity of the band. If we add a further requirement that the pump should be active whenever the methane level is below a level of *Safe*, then this will put an upper bound on $n * \mathit{Rate\_meth} + 2 * \mathit{Acc\_meth}$.

This mine pump example does not need time bands in order to specify its behaviour. But the brief outline here does have a much simpler form than other descriptions. This clarity comes from being able to initially specify all behaviours as instantaneous events and all actions as immediate. This very real-time system did not need time in its initial specification, but it does allow time to play its full role in more detailed levels of description.

## 8 Related and Future Work

The concept of time granularity has been previously discussed in the literature [26, 15] and many projects have focused on time granularity within different areas of computer science, such as temporal databases, data mining, formal specification, etc. Generally, the basic idea of time granularity is to partition a universal time domain into differently-grained granules, and that a granularity is a set of indexed granules, any one of which is a set of time instants.

So far, most of this work have focused on embedding time granularity in temporal logic languages. For example, early exploration [18,36] consists of translation mechanisms that map a formula associated with different time constraints to the finest granularity. They later [17,13,14] revise the simple approach by extending the basic logic language with contextual and projection operators, so that the enhanced semantics can express more general and complete properties. Additional work [16,20] aimed at reasoning about time granularity is also proposed.

Interval temporal logic (ITL) was originally designed for reasoning about hardware circuits [24,37]. When modelling hardware, it is natural to look at a circuit's behaviour at different granularities of time. For example, the units of time might correspond to regularly spaced clock ticks or to nanoseconds. ITL has a temporal projection operator to denote the process of mapping from one level of time to another: $w_1$ *proj* $w_2$, where $w_1$ and $w_2$ both denote ITL formulas. This operator has been implemented in the Tempura programming language [38].

Bettini introduces a glossary of *time granularity concepts* [4]. Their work is not committed to a particular model of time, which could be discrete (such as the natural numbers), dense (such as the rationals), or continuous (such as the reals). The time domain is discretised into countable granules of time, and an interpretation function relates the index of each granule to an interval in the time domain.

Broy [6] takes a highly abstract view of real-time interactive systems, where the system is described by a set of timed events that represent possible observations. This set is represented by a function $time : E \rightarrow TIME$, that maps each event to the time of its occurrence. Broy considers time transformers to change the timing of systems. Suppose that $trans : TIME \rightarrow TIME$, then it can be used to transform the system using function composition: $time' = trans \circ time$. As a result of a time transformation, the new timing may be coarser. Two events $e_1$ and $e_2$, with the timing property $time(e1) < time(e2)$ may become simultaneous events under $time'$: we may get $time'(e1) = time'(e2)$. Broy goes on to introduce a pair of complementary functions $COA(n)$ and $FINE(n)$, which make a system's timing coarser or finer by a factor of $n$. These functions permit the scaling of a timed system by any rational amount.

Furia et al [23] is their extensive survey of how time is represented in models of computing, note that in most systems "*the dynamics of the components range over widely different time scales and time granularities (in particular, continuous and discrete components) are integrated.*" Nevertheless they note that few languages approach the granularity problem in a formal way.

All of this work, however, focuses only on incorporating different time scales, rather than the more expressive idea of bands as defined in this paper. Although they

address granularity they do not consider precision and the other properties summarized in Section 5.

The representation, within the timeband model, of repetitive events and activities (e.g. $E \# i$) is taken from the notion of occurrences in RTL [30]. The property that a set of events can be ordered but occur at the same time instance has some resonance with the notion of super-dense chop [11] in Duration Calculus [12].

Current and future work with the timeband framework is focused on the application of the ideas to industrial case studies – including system of systems. These studies are addressing the issues of extracting the right bands for a particular application, and the assignment of events and activities to these bands. Initial work [50] has indicated the value of first assigning events, as they are more abstract, and then considering activities as a refinement of these events.

Work has also involved[9] modelling timebands with a hierarchy of descriptions. The top-level description uses CircusTime [46], a compact time-based notation that is an extension of the Circus notation [51]. Reasoning at this level is done using an interactive theorem prover [42,43]. CircusTime is a powerful language, with support for imperative programming with concurrency operators similar to those found in CSP. It is also possible to write abstract specifications to specify all or part of a process, and a notion of refinement connects abstract specifications with their implementations [10].

Concrete descriptions in CircusTime can be translated into `CSP_M`, the language for the FDR theorem prover. `CSP_M` has no built-in notion of time, so CircusTime descriptions can be refined into an untimed description that uses a number of timers. The untimed description can then be analysed using the FDR refinement model-checker and the ProB animator. This translation uses the framework in [46] and the translation worked out in [21]. The `CSP_M` descriptions are then implemented in Java using JCSP [49], which is a pure Java class library providing a base range of CSP primitives plus a rich set of extensions. It also includes a package providing CSP process wrappers giving a channel interface to all Java AWT widgets and graphics operations.

## 9 Conclusion

In this paper we have argued that complex real-time systems exhibit behaviour at many different time levels and that a useful aid in describing and specifying such behaviour is to use time bands. Viewing a system as a collection of event and activities within a finite set of bands is an effective means of separating concerns and identifying inconsistencies between different 'layers' of the system. Time bands are not mapped on to a single notion of physical time. Within a system, there will always be a relationship between bands, but the bands need not be tightly synchronised. There is always some level of imprecision between any two adjacent bands.

The use of the timeband framework is intended to help develop a comprehensive foundation to the study and development of future systems. Of course an adequately expressive model of time is just one element of such a foundation, but it is perhaps the most important to define if dependable systems are to be engineered.

[9] Within the Indeed project – http://www.indeedproject.ac.uk/.

**Acknowledgements** The authors would like to thank the useful contributions of all the people who have discussed time bands with us – in particular Brijesh Dongol, Colin Fidge, Leandro Soares Indrusiak, Cliff Jones, Michael Jackson, Peter Robinson, Kun Wei, and Jim Woodcock. This work is supported, in part, by the EPSRC(UK) through its funding of the INDEED project and the TrAmS platform, and is also supported, in part, by the Australian Research Council (ARC) Discovery Grant DP0987452, *Combining Time Bands and Teleo-Reactive Programs for Advanced Dependable Real-Time Systems*.

## References

1. J. Allen. Towards a general theory of actions and time. *Artificial Intelligence*, 23:123–154, 1984.
2. G. Baxter, A. Burns, and K. Tan. Evaluating timebands as a tool for structuring the design of socio-technical systems. In P. Bust, editor, *Contemporary Ergonomics 2007*, pages 55–60. Taylor & Francis, 2007.
3. M. Bergadaà. Temporal frameworks and individual cultural activities: Four typical profiles. *Time and Society*, 16(2/3):387–408, 2007.
4. C. Bettini, C.E. Dyreson, W. S. Evans, R. T. Snodgrass, and X. S. Wang. A glossary of time granularity concepts. In *Temporal Databases, Dagstuhl*, pages 406–413, 1997.
5. S. Blount, M.J. Walker, and S. Leroy. Coping with temporal uncertainty: when rigid ambitious deadlines don't make sense. In *Organization at the Limit*, pages 122–34. Springer, 2007.
6. M. Broy. *Relating Time and Causality in Interactive Distributed Systems*, pages 75–130. IOS Press, 2008.
7. A. Burns and G.D. Baxter. Time bands in systems structure. In *Structure for Dependability*, pages 74–90. Springer, 2006.
8. A. Burns, I.J. Hayes, G.Baxter, and C.J. Fidge. Modelling temporal behaviour in complex socio-technical systems. Computer Science Technical Report YCS 390, University of York, 2005.
9. A. Burns and A. M. Lister. A framework for building dependable systems. *Computer Journal*, 34(2):173–181, 1991.
10. A. Cavalcanti, A. Sampaio, and J. Woodcock. A refinement strategy for circus. *Formal Asp. Comput.*, 15(2–3):146–181, 2003.
11. Z. Chaochen and M.R. Hansen. Chopping a point. In *BCS-FACS 7th Refinement Workshop*. Electronic Workshops in Computing, Springer-Verlag, 1996.
12. Z. Chaochen, C.A.R. Hoare, and A.P. Ravn. A Calculus of Duration. *Information Processing Letters*, pages 40:269–276, 1991.
13. E. Ciapessoni, E. Corsetti, A. Montanari, and P. San Pietro. Embedding time granularity in a logical specification language for synchronous real-time systems. *Science of Computer Programming*, 20:141–171, 1993.
14. E. Ciapessoni, E. Corsetti, A. Montanari, and P. San Pietro. Embedding time granularity in a logical specification language for synchronous real-time systems. In *6IWSSD: Selected Papers of the Sixth International Workshop on Software Specification and Design*, pages 141–171, Amsterdam, 1993. Elsevier Science Publishers B. V.
15. J. Clifford and A. Rao. A simple, general structure for temporal domains. In *Temporal Aspects in information Systems*, pages 23–30. AFCET, 1987.
16. C. Combi, M. Franceschet, and A. Peron. Representing and reasoning about temporal granularities. *J. Log. and Comput.*, 14(1):51–77, 2004.
17. E. Corsetti, A. Montanari, and E. Ratto. Dealing with different time granularities in formal specifications of real-time systems. *Journal of Real-Time Systems*, 3(2):191–215, 1991.
18. E. Corsetti, A. Montanari, and E. Ratto. Time granularity in logical specifications. In *Proceedings of the 6th Italian Conference on Logic Programmming, Pisa, Italy*, 1991.
19. P. Fraisse. *The psychology of time*. New York: Harper and Row, 1963.
20. M. Franceschet and A. Montanari. Temporalized logics and automata for time granularity. *Theory Pract. Log. Program.*, 4(5-6):621–658, 2004.
21. A. Freitas and A. Cavalcanti. Automatic translation from circus to java. In J. Misra, T. Nipkow, and E. Sekerinski, editors, *FM*, volume 4085 of *Lecture Notes in Computer Science*, pages 115–130. Springer, 2006.
22. W. Friedman. *About time: Inventing the fourth dimension*. Cambridge, MA: MIT Press, 1990.
23. C.A. Furia, D. Mandrioli, A. Morzenti, and M. Rossi. Modeling time in computing: A taxonomy and a comparative survey. *ACM Computing Surveys*, 42(6):1–59, 2010.

24. J.Y. Halpern, Z. Manna, and B. C. Moszkowski. A hardware semantics based on temporal intervals. In J. Díaz, editor, *ICALP*, volume 154 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 1983.
25. I.J. Hayes, editor. *Specification Case Studies*. Prentice-Hall, 1987.
26. J. Hobbs. Granularity. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence, Los Angeles, California*, pages 432–435, 1985.
27. E. Hollnagel. *Human Reliability Analysis: Context and Control*. Academic Press, 1993.
28. G. E. Hughes and M. J. Cresswell. *An Introduction to Modal Logic*. University Paperbacks. Routledge, 1968.
29. S.G. Hutchesson and N. Hayes. Technology transfer and certification issues in safety critical real-time systems. In *Digest of the IEE Colloquium on Real-Time Systems*, volume 98/306, April 1998.
30. F. Jahanian and A.K. Mok. Safety analysis of timing properties in real-time systems. *Transactions on Software Engineering*, SE-12(9), 1986.
31. C.B. Jones, I.J. Hayes, and M.A. Jackson. Deriving specifications for systems that are connected to the physical world. In Cliff B. Jones, Zhiming Liu, and Jim Woodcock, editors, *Formal Methods and Hybrid Real-Time Systems*, volume 4700 of *Lecture Notes in Computer Science*, pages 364–390. Springer Verlag, 2007.
32. M. Joseph, editor. *Real-Time Systems: Specification, Verification and Analysis*. Prentice-Hall, 1996.
33. L. Lamport. Time, clocks, and the ordering of events in a distributed system. *CACM*, 21(7):558–565, 1978.
34. R. Levine. *A geography of time*. New York: Guilford Press, 1997.
35. B. Mahony and I.J. Hayes. A case_study in timed refinement: A mine pump. *IEEE Transactions on Software Engineering*, SE-18(9):817–826, 1992.
36. A. Montanari, E. Ratto, E. Corsetti, and A. Morzenti. Embedding time granularity in logical specifications of real-time systems. In *Proceedings of the Third Euromicro Workshop on Real-Time Systems, Paris, France*, 1991.
37. B. Moszkowski. *Reasoning about Digital Circuits*. PhD thesis, Department of Computer Science, Stanford University. (Available as technical report STANCS83970.), 1983.
38. B. Moszkowski. *Executing Temporal Logic Programs*. Cambridge University Press, 1986.
39. A. Newell. *Unified theories of cognition*. Harvard University Press, Cambridge, MA, 1990.
40. N.J. Nilsson. Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1:139–158, 1994.
41. N.J. Nilsson. Teleo-reactive programs and the triple-tower architecture. *Electronic Transactions on Artificial Intelligence*, 5:99–110, 2001.
42. M. Oliveira, A. Cavalcanti, and J. Woodcock. A utp semantics for *ircus*. *Formal Asp. Comput.*, 21(1-2):3–32, 2009.
43. M.V.M. Oliveira, A. Cavalcanti, and J. Woodcock. Unifying Theories in ProofPowerZ. *Formal Aspects of Computing, online first*, 2007. DOI 10.1007/s00165-007-0044-5.
44. J.E. Roeckelein. *The concept of time in psychology: A resource book and annotated bibliography*. CT: Greenwood Press, 2000.
45. W. Schneider. Training high-performance skills: Fallacies and guidelines. *Human Factors*, 27(3):285–300, 1985.
46. A. Sherif and J. He. Towards a time model for circus. In C. George and H. Miao, editors, *ICFEM*, volume 2495 of *Lecture Notes in Computer Science*, pages 613–624. Springer, 2002.
47. H.A. Simon. *The Science of the Artificial - 3rd Edition*. MIT Press, 1996.
48. J. M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall International, second edition, 1992.
49. P. H. Welch. Process oriented design for java: Concurrency for all. In H. R. Arabnia, editor, *PDPTA*. CSREA Press, 2000.
50. R. White. *Capturing the temporal properties of complex systems: an evaluation of the timebands approach*. PhD thesis, University of York, Computer Science, York, UK, 2010.
51. J. Woodcock and A. Cavalcanti. The semantics of circus. In D. Bert, J. P. Bowen, M. C. Henson, and K. Robinson, editors, *ZB*, volume 2272 of *Lecture Notes in Computer Science*, pages 184–203. Springer, 2002.

## A Notation

A partial order "$\preceq$" on some set $X$ is reflexive, transitive, and anti-symmetric.

$[X]$

$$partial\_order : \mathbb{P}(X \leftrightarrow X)$$

$$\begin{aligned} partial\_order = \{ \_ \preceq \_ : X \leftrightarrow X \mid \forall x1, x2, x3 : X \bullet \\ x1 \preceq x1 \land \\ (x1 \preceq x2 \land x2 \preceq x3 \Rightarrow x1 \preceq x3) \land \\ (x1 \preceq x2 \land x2 \preceq x1 \Rightarrow x1 = x2)\} \end{aligned}$$

$[X]$

$$symmetric\_rel : \mathbb{P}(X \leftrightarrow X)$$

$$\begin{aligned} symmetric\_rel = \{ \_ \simeq \_ : X \leftrightarrow X \mid (\forall e, f : X \bullet \\ e \simeq e \land (e \simeq f \Leftrightarrow f \simeq e))\} \end{aligned}$$