

A Timed Model of *Circus* with the Reactive Design Miracle

Kun Wei, Jim Woodcock and Alan Burns

Computer Science, University of York

20 Jan 2009

Alphabetised relational calculus

- Circus is a combination of CSP, Z and the refinement calculus.
- The semantic model of UTP is based on the alphabetised relational calculus.
- A relation is a pair $(\alpha P, P)$
 - $\alpha P = in\alpha P \cup out\alpha P$
 - Conditional: $P \triangleleft b \triangleright Q$
 - Composition: $P;Q$
 - Assignment: $(x:=e)$
 - Non-determinism: $P \sqcap Q$
 - The complete lattice of relations: Abort(true) and Miracle(false)

A theory in UTP

A *theory* is a collection of relations, containing three essential parts:

- an alphabet: a set of variable names, e.g., *okay*, *tr*;
- a signature: it gives the rules for syntax;
- healthiness conditions: a collection of some fundamental laws which rule out those relations we are not interested in.

Reactive designs

- A design in UTP is a relation that can be split into a precondition-postcondition pair with an observational variable.

$$P \vdash Q \hat{=} okay \wedge P \Rightarrow okay' \wedge Q$$

- To describe timed reactive processes and its behaviour with time, we introduce more observational variables:
 - *wait* and *wait'*: it is used to distinguish an intermediate observation from the observation made on termination.
 - *t* and *t'*: the start point and end point of a time interval.
 - *tr* and *tr'*: traces
 - *ref* and *ref'*: refusal sets
 - *v*, *v'*: values of a process's local variables

Healthiness conditions for reactive processes

- R1 $P = P \wedge tr \leq tr'$: never change history
- R2 $P(tr, tr') = P(\langle \rangle, tr' - tr)$: ignore history
- R3 $P = \mathbb{I}_{rea} \triangleleft wait \triangleright P$: wait to start, preserving divergence
- R4 $P = P \wedge t \leq t'$: never undo time

Then a reactive miracle is defined as follows:

$$\top_R \hat{=} R(true \vdash false) = R(\neg okay)$$

So the new timed model is a complete lattice, rather than the complete partial orders of the standard model for CSP.

Primitive processes

- Chaos: the reactive abort

$$CHAOS \hat{=} R(\text{false} \vdash \text{true})$$

- Stop: a deadlocked process

$$STOP \hat{=} R(\text{true} \vdash tr' = tr \wedge wait')$$

- Skip: terminates immediately

$$SKIP \hat{=} R(\text{true} \vdash tr' = tr \wedge v' = v \wedge \neg wait' \wedge t' = t)$$

Prefixing a miracle

$$a \rightarrow SKIP \hat{=} R \left(\begin{array}{l} tr' = tr \wedge a \notin ref' \\ true \vdash \triangleleft wait' \triangleright \wedge v' = v \\ tr' = tr \hat{\wedge} \langle (t', a) \rangle \end{array} \right)$$

$$a \rightarrow \top_R = R (true \vdash (tr' = tr \wedge a \notin ref' \wedge wait' \wedge v' = v))$$

This is a miraculous process that is waiting for interaction with its environment, but never actually performing it even if the event a has been offered.

External choice with a miracle

$$(a \rightarrow \text{SKIP}) \sqcap \top_R$$

$$\hat{=} R(\text{true} \vdash \neg \text{wait}' \wedge \text{tr}' = \text{tr} \hat{\wedge} \langle (t', a) \rangle \wedge v' = v)$$

This is another very strange process that states, if its predecessor successfully terminates, it performs the event a and terminates immediately. Note that, since there is not the case that wait' is true, the process has no state that it is waiting for the interaction of its environment.

- Internal choice: $P \sqcap \top_R = P$
- Parallel composition: $P \parallel \top_R = \top_R$

Ordered simultaneity

- In timed CSP, a process , $a \rightarrow b \rightarrow SKIP$, may have a trace like $\langle \{t, a\}, \{t, b\} \rangle$.
- True concurrency is allowed in CSPP in which multiple events can occur simultaneously but without any order.
- The most important contribution of reactive miracles to Circus is we can define ordinary *urgent* events.
- In timed CSP, internal events are *urgent* because they cannot be blocked.
- In our model, an urgent event means it cannot be blocked under some circumstances.

Urgent events

$$((a \rightarrow \text{SKIP}) \square \top_R) \parallel_{\{a\}} \text{STOP} = \top_R$$

$$c \rightarrow (a \rightarrow (b \rightarrow \text{SKIP} \square \top_R)) \parallel_{\{a\}} \text{STOP}$$

$$= c \rightarrow \text{STOP}$$

$$c \rightarrow (a \rightarrow (b \rightarrow \text{SKIP} \square \top_R)) \parallel_{\{b\}} \text{STOP}$$

$$= c \rightarrow (a \rightarrow \top_R)$$

$$\dagger a \rightarrow \text{SKIP} \hat{=} (a \rightarrow \text{SKIP}) \square \top_R$$

Radio time announcement

- In a hour band,
 $Time(i) = hour.i; WAIT\ 1; Time((i + 1) mod\ 24)$
- In a second band,
 $Ticks(i) = beep \xrightarrow{1} beep \xrightarrow{1} beep \xrightarrow{1} beep \xrightarrow{1} beeeep \xrightarrow{1} announce.i \rightarrow SKIP$
- To maintain the consistency of mapping, we define
 $UTicks(i) = beep \xrightarrow{1} \ddagger beep \xrightarrow{1} \ddagger beep \xrightarrow{1} \ddagger beep \xrightarrow{1} \ddagger beeeep \xrightarrow{1} \ddagger announce.i \rightarrow SKIP$

Radio time announcement

- To maintain the coordination, we define *hour.i* is urgent:

$$\begin{aligned} & (Time(i) ||| UTicks(i)) \\ & \quad || \\ & \quad \{beeeep, hour.i\} \\ & beeeep \rightarrow \ddagger hour.i \rightarrow SKIP \end{aligned}$$

Deadlines

Strick deadline operator:

$$P \blacktriangleright d \hat{=} P \triangleright_d \top_R$$

- $a \rightarrow ((b \rightarrow \text{SKIP}) \blacktriangleright d)$
- $(a \rightarrow b \rightarrow \text{SKIP}) \blacktriangleright d$
-

$$\begin{array}{c} ((a \rightarrow \text{SKIP}) \blacktriangleright d \parallel\parallel (b \rightarrow \text{SKIP}) \blacktriangleright d) \\ \parallel \\ \{a, b\} \\ a \rightarrow b \rightarrow \text{SKIP} \end{array}$$

Conclusion and future work

- In this work we present a timed model for *Circus* involving the reactive miracle.
- The model seems to be a promising contribution to the timebands model.
- We need to further discuss the refinement of systems in detail.
- We are also about to explore a hybrid system that effectively uses discrete time to express concurrency and naturally uses continuous time to describe the variables that are changing continuously.