Circus-SCJ Time Action Model

Kun Wei and Jim Woodcock

Department of Computer Science
University of York, UK

October 11, 2011
Outline

- Introduction and motivation
- The syntax and semantics of *Circus*-SCJ Time
- Why is the reactive-design miracle introduced?
- How can we mechanise *Circus*-SCJ Time?
- Conclusion and future work
Introduction and Motivation

- **Circus-SCJ Time** is used in the abstract (top) model in the development and verification of SCJ programs (hiJaC project).
- **Circus-SCJ Time** is a new version of **Circus** and an extension to **Circus Time** with constructs that correspond to the components of SCJ programs.

What we have changed to the original **Circus Time**?

- Fixed a number of ‘bugs’
- Changed to reactive design semantics
- Introduced Miracle and some time operators
- Proved a number of laws involving Miracle
The syntax and semantics of *Circus*-SCJ Time

- UTP theories
- The syntax of *Circus*-SCJ Time
- The semantics of *Circus*-SCJ Time
UTP theories

- Theories of relations, designs and reactive processes
  - A relation is a predicate with an alphabet (undashed and dashed)
  - A design is a relation expressed as a pre and postcondition pair with $ok$
    $$P \vdash Q \equiv ok \land P \Rightarrow ok' \land Q$$
  - A reactive process can interact with its environment and satisfies $R1, R2$ and $R3$.
- Observational variables in Circus-SCJ Time
  - $ok, ok', \text{wait, wait'}$: boolean
  - $tr, tr': seq^+(seq \ Event)$
  - $ref, ref': seq^+(\mathbb{P} \ Event)$
  - $\text{state, state'}: N \rightarrow value$
The syntax of Circus-SCJ Time

\[ \text{Action} ::= \text{Skip} | \text{Stop} | \text{Miracle} | \text{Chaos} \]
\[ | \text{Communication} \rightarrow \text{Action} | \text{Action} ; \text{Action} \]
\[ | \text{Action} \triangleleft b \triangleright \text{Action} | N := e \]
\[ | b \& \text{Action} | \text{Action} \sqcap \text{Action} | \text{Action} \sqsupset \text{Action} \]
\[ | \text{Action} | [s_1 | \{ | \text{CS} | \} | s_2] | \text{Action} | \text{Action} \setminus \text{CS} \]
\[ | \text{Wait} d | \text{Wait} (d_1 .. d_2) \]
\[ | \text{Action} \triangleright \{d\} \text{Action} | \text{Action} \triangleright d | \text{Action} \triangleright d \]
\[ | \text{Communication}@t \rightarrow \text{Action} \]
\[ | \mu N \bullet \text{Action} \]

\[ \text{Communication} ::= \ N \ CParameter^* \]
\[ CParameter ::= ?N | !e | .e \]
Healthiness conditions in *Circus*-SCJ Time

\[ R_{1st}(X) \equiv X \land tr \preceq tr' \land front(ref) \leq ref' \land \#ref' = \#tr' \land \#ref = \#tr \]
\[ R_{2st}(X) \equiv \exists r \cdot X[⟨⟨⟩⟩, \text{diff}(tr', tr), ⟨∅⟩, ref' \rightarrow front(ref')/tr, tr', ref, ref'] \]
\[ R_{3st}(X) \equiv II_{st} \triangle wait \triangleright X \]
\[ R_{st} = R_{1st} \circ R_{2st} \circ R_{3st} \]
\[ CSP_{1st}(X) \equiv X \lor \left( \neg ok \land tr \preceq tr' \land front(ref) \leq ref' \land \#ref' = \#tr' \land \#ref = \#tr \right) \]
\[ CSP_{2st}(X) \equiv X ; \left( (ok \supset ok') \land tr' = tr \land front(ref') = front(ref) \land wait' = wait \land state' = state \right) \]
\[ CSP_{3st}(X) \equiv Skip ; X \]
\[ CSP_{4st}(X) \equiv X ; Skip \]
\[ CSP_{5st}(X) \equiv X \| Skip \]
Examples of reactive designs

- **Relational semantics in Circus Time**
  \[
  c.e \rightarrow Skip \equiv CSP_{1t}(ok' \land R_t(wait\_com(c) \lor terminating\_com(c.e)))
  \]

- **Reactive design semantics in Circus-SCJ Time**
  \[
  c.e \rightarrow Skip \equiv R_{st}(true \vdash wait\_com(c) \lor terminating\_com(c.e))
  \]

- **Reactive design miracle**
  \[
  Miracle \equiv R_{st}(true \vdash false) = R_{1st}(\neg ok) \lor (ok' \land wait \land II')
  \]

- **Chaos**
  \[
  Chaos \equiv R_{st}(true) = R_{1st}(\neg wait) \lor Miracle
  \]
New time operators in *Circus*-SCJ Time

- Timed event prefix
  \[ c.e@t \rightarrow Skip \equiv R_{st}(true \vdash wait\_com(c) \lor termination\_com\_time(c.e, t)) \]

- Deadlines
  \[ A \leftarrow d \equiv A \square (Wait\ d\ ;\ Miracle) \]
  \[ A \rightarrow d \equiv R_{st}(true \vdash A \land \#tr' - \#tr \leq d) \]
What *Miracle* brings to us?

\[ c.e \rightarrow \text{Miracle} = R_{st}(true \vdash \text{wait}' \land \lnot/\text{tr}' = \lnot/\text{tr} \land \text{possible}(\text{tr}, \text{tr}', c)) \]

This action interestingly states that, if the action starts stably, it will wait for interaction with its environment \((\text{wait}' = \text{true})\), but never actually performs any event \((\lnot/\text{tr}' = \lnot/\text{tr})\) even if the event \(c.e\) has been offered. This action violates an axiom of the standard CSP failures-divergences model,

\[ \text{F3. } (s, X) \in F \land \exists a \in Y \bullet s \lnot\langle a \rangle \notin \text{traces}_\bot(P) \Rightarrow (s, X \cup Y) \in F \]

saying if at a state an event is not in the refusal set then the process is willing to execute the event.

So, what's the behaviour of \(\text{Wait } d ; \text{ Miracle}\)?
What *Miracle* brings to us?

- \((c.e \rightarrow Skip) \square Miracle = R_{st}(true \vdash \neg wait' \land dif(t', t) = \langle c.e \rangle)\)

This actually states that this action performs the event \(c.e\) and terminates immediately. There is no state in which the action is waiting for the environment to offer \(c.e\). It simply occurs instantly since no \(t' = t\) can be found in the semantics. Obviously, it violates another important axiom of the standard CSP models where traces are prefix closed.

\[(c.e \rightarrow Skip) \square Miracle \square (a.e \rightarrow Skip) = R_{st}(true \vdash \neg wait' \land (diff(t', t) = \langle c.e \rangle \lor dif(t', t) = \langle a.e \rangle))\]

- External events become urgent.
- A number of algebraic laws involving *Miracle* have been proved to understand how it influences other operators.
Mechanisation of *Circus*-SCJ Time

- *Circus* and UTP in ProofPower-Z
  - Marcel Oliveira’s initial model
  - Frank Zeyda’s generic model
- Recent work that embeds UTP theories in Isabelle/HOL
- Can we model check *Circus/*Circus*-SCJ Time?
  - Yes, we may, because
  - John Derrick et al. have developed a tool (Z2SAL) which automatically translate Z specifications into SAL input language.
  - we have explored the translation of *Circus* Action model with Miracle to SAL, and the experiment result is very promising.
Conclusion and Future work

- We have developed a new version of Circus Time to describe timing behaviour of SCJ programs.
- We have improved consistency of the denotational semantics of Circus-SCJ Time by fixed a number of errors in the old version of Circus Time.
- The behaviour of Miracle with other operators has been fully explored, so as to generate a right operational semantics.
- Future work
  - Mechanisation of the semantics of Circus-SCJ Time in a theorem prover.
  - Collapsing Parallelism, e.g., if $P = a \rightarrow P$ and $Q = b \rightarrow Q$, then $P ||| Q = R$ where $R = a \rightarrow R \& b \rightarrow R$. 