

# *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

- 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  $ok$

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

- A reactive process can interact with its environment and satisfies  $R1, R2$  and  $R3$ .
- Observational variables in *Circus-SCJ Time*
  - $ok, ok', wait, wait'$ : *boolean*
  - $tr, tr'$ :  $seq^+(seq\ Event)$
  - $ref, ref'$ :  $seq^+(\mathbb{P}\ Event)$
  - $state, state'$ :  $N \rightarrow value$

# The syntax of *Circus-SCJ* Time

$Action ::= Skip \mid Stop \mid Miracle \mid Chaos$   
 $\mid Communication \rightarrow Action \mid Action ; Action$   
 $\mid Action \triangleleft b \triangleright Action \mid N := e$   
 $\mid b \& Action \mid Action \square Action \mid Action \sqcap Action$   
 $\mid Action \llbracket s_1 \mid \{ \} CS \rrbracket \mid s_2 \rrbracket Action \mid Action \setminus CS$   
 $\mid Wait\ d \mid Wait\ (d_1..d_2)$   
 $\mid Action \triangleright \{d\} Action \mid Action \blacktriangleright d \mid Action \blacktriangleleft d$   
 $\mid Communication@t \rightarrow Action$   
 $\mid \mu N \bullet Action$

$Communication ::= N\ CParameter^*$

$CParameter ::= ?N \mid !e \mid .e$

# Healthiness conditions in *Circus*-SCJ Time

$$R1_{st}(X) \hat{=} X \wedge tr \preceq tr' \wedge front(ref) \leq ref' \wedge \#ref' = \#tr' \wedge \#ref = \#tr$$

$$R2_{st}(X) \hat{=} \exists r \bullet X[\langle\langle\rangle\rangle, diff(tr', tr), \langle\emptyset\rangle, ref' - front(ref)/tr, tr', ref, ref']$$

$$R3_{st}(X) \hat{=} \mathbb{I}_{st} \triangleleft wait \triangleright X$$

$$R_{st} = R1_{st} \circ R2_{st} \circ R3_{st}$$

$$CSP1_{st}(X) \hat{=} X \vee \left( \begin{array}{l} \neg ok \wedge tr \preceq tr' \wedge front(ref) \leq ref' \\ \wedge \#ref' = \#tr' \wedge \#ref = \#tr \end{array} \right)$$

$$CSP2_{st}(X) \hat{=} X; \left( \begin{array}{l} (ok \Rightarrow ok') \wedge tr' = tr \wedge front(ref') = front(ref) \\ \wedge wait' = wait \wedge state' = state \end{array} \right)$$

$$CSP3_{st}(X) \hat{=} Skip; X$$

$$CSP4_{st}(X) \hat{=} X; Skip$$

$$CSP5_{st}(X) \hat{=} X ||| Skip$$

# Examples of reactive designs

- Relational semantics in *Circus* Time

$$c.e \rightarrow \text{Skip} \hat{=} \text{CSP1}_t(\text{ok}' \wedge R_t(\text{wait\_com}(c) \vee \text{terminating\_com}(c.e)))$$

- Reactive design semantics in *Circus-SCJ* Time

$$c.e \rightarrow \text{Skip} \hat{=} R_{st}(\text{true} \vdash \text{wait\_com}(c) \vee \text{terminating\_com}(c.e))$$

- Reactive design miracle

$$\text{Miracle} \hat{=} R_{st}(\text{true} \vdash \text{false}) = R1_{st}(\neg \text{ok}) \vee (\text{ok}' \wedge \text{wait} \wedge \text{II}')$$

- *Chaos*  $\text{Chaos} \hat{=} R_{st}(\text{true}) = R1_{st}(\neg \text{wait}) \vee \text{Miracle}$

# New time operators in *Circus-SCJ* Time

- Timed event prefix

$c.e@t \rightarrow Skip \hat{=} R_{st}(true \vdash wait\_com(c) \vee termination\_com\_time(c.e, t))$

- Deadlines

$A \blacktriangleleft d \hat{=} A \square (Wait\ d; Miracle)$

$A \blacktriangleright d \hat{=} R_{st}(true \vdash A \wedge \#tr' - \#tr \leq d)$

# What *Miracle* brings to us?

- $c.e \rightarrow \text{Miracle} = R_{st}(\text{true} \vdash \text{wait}' \wedge \wedge / \text{tr}' = \wedge / \text{tr} \wedge \text{possible}(\text{tr}, \text{tr}', c))$

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 ( $\wedge / \text{tr}' = \wedge / \text{tr}$ ) even if the event  $c.e$  has been offered. This action violates an axiom of the standard CSP failures-divergences model,

$$\mathbf{F3.} \quad (s, X) \in F \wedge \exists a \in Y \bullet s \hat{\ } \langle a \rangle \notin \text{traces}_{\perp}(P) \Rightarrow (s, X \cup Y) \in F$$

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 *Wait d*; *Miracle*?

# What *Miracle* brings to us?

- $(c.e \rightarrow \text{Skip}) \sqcap \text{Miracle} = R_{st}(\text{true} \vdash \neg \text{wait}' \wedge \text{diff}(tr', tr) = \langle c.e \rangle)$

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  $tr' = tr$  can be found in the semantics. Obviously, it violates another important axiom of the standard CSP models where traces are prefix closed.

$$(c.e \rightarrow \text{Skip}) \sqcap \text{Miracle} \sqcap (a.e \rightarrow \text{Skip}) \\ = R_{st}(\text{true} \vdash \neg \text{wait}' \wedge (\text{diff}(tr', tr) = \langle c.e \rangle \vee \text{diff}(tr', tr) = \langle a.e \rangle))$$

- 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  $P \parallel Q = R$  where  $R = a \rightarrow R \square b \rightarrow R$ .