Circus Time and SCJ

Department of Computer Science
University of York

15 November, 2011

Motivation

Programming real-time systems using a high-level language is notoriously difficult because time response depends on many facts in a low level such as complier, OS, hardware and so on.

One of solutions is to use (hard) deadlines, which state enforced timing requirements.

- Hayes et al introduced a deadline command to the safety-critical SPARK programming language.
- Failure to meet such a deadline results in infeasibility.

Apart from some usual time operators from Timed CSP, *Circus Time* provides a **deadline** operator.

Summary of Circus Time

- A discrete-time model
- Semantics is based on UTP
- An extension to *Circus* and original *Circus Time* (Sherif and He)
- A brand-new deadline operator and an infeasible process (Miracle)
- Reactive-design semantics to each process
- A number of algebraic laws

Observation in Circus Time

Observational variables:

- ok,ok': *boolean*
- wait,wait': *boolean*
- tr,tr':$seq^{+}(seq\ Event)$
- ref,ref':$seq^{+}(\mathbb{P}\ Event)$
- state, state':$N \pfun value$

For example,

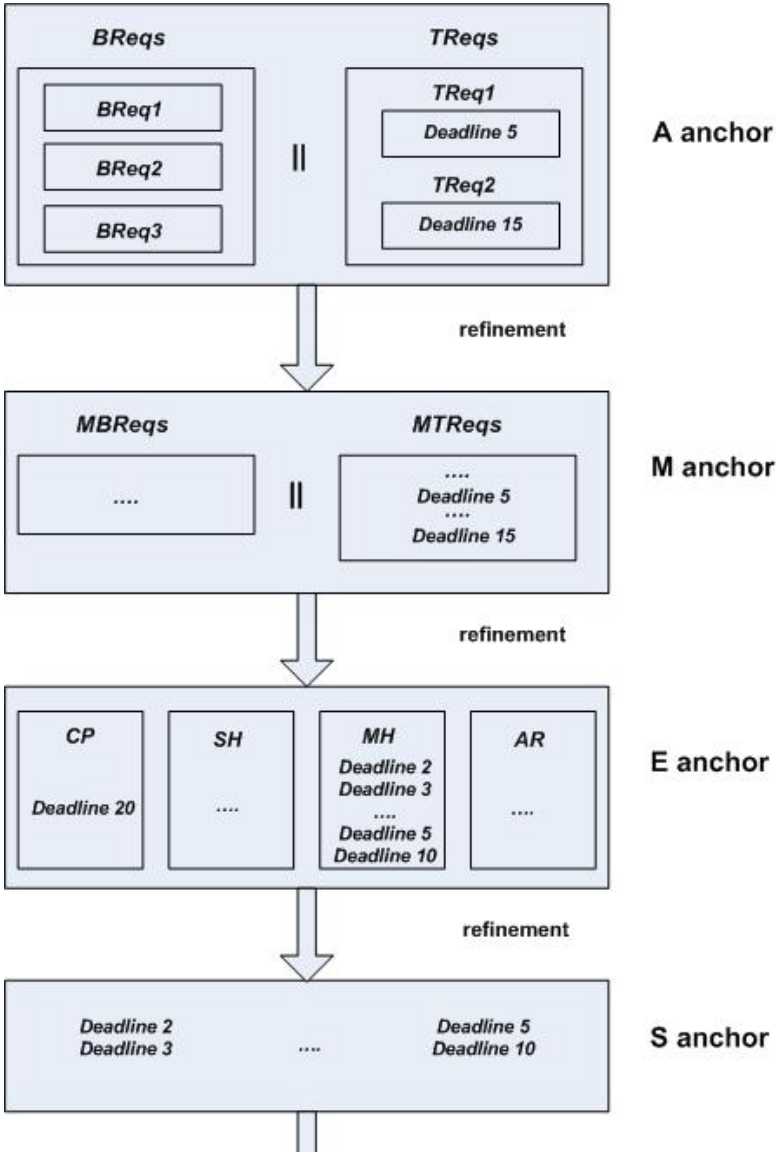
$$tr' = \langle \langle a \rangle, \langle b, c \rangle, \langle d \rangle, \langle e, f \rangle, ... \rangle$$
$$ref' = \langle r_1, r_2, r_3, r_4, ... \rangle$$

Time operators in Circus Time

- $Wait\ d$: wait for d time units
- $Wait\ d_1..d_2$: non-deterministic wait
- $P \triangleright \{d\}\ Q$: if no observable event in P happens within d, Q will take place
- $P \blacktriangleright d$: P **must** terminate within d
- $P \blacktriangleleft d$: observable events in P **must** happen within d
- $c.e@t \rightarrow P$: t records the amount of time which has elapsed between the start and the occurrence of $c.e$

$$(Wait\ 2\,;\,(a \rightarrow P)) \blacktriangleright 5 \text{ or } (a \rightarrow b \rightarrow Skip) \blacktriangleleft 5$$

Refinement Strategy



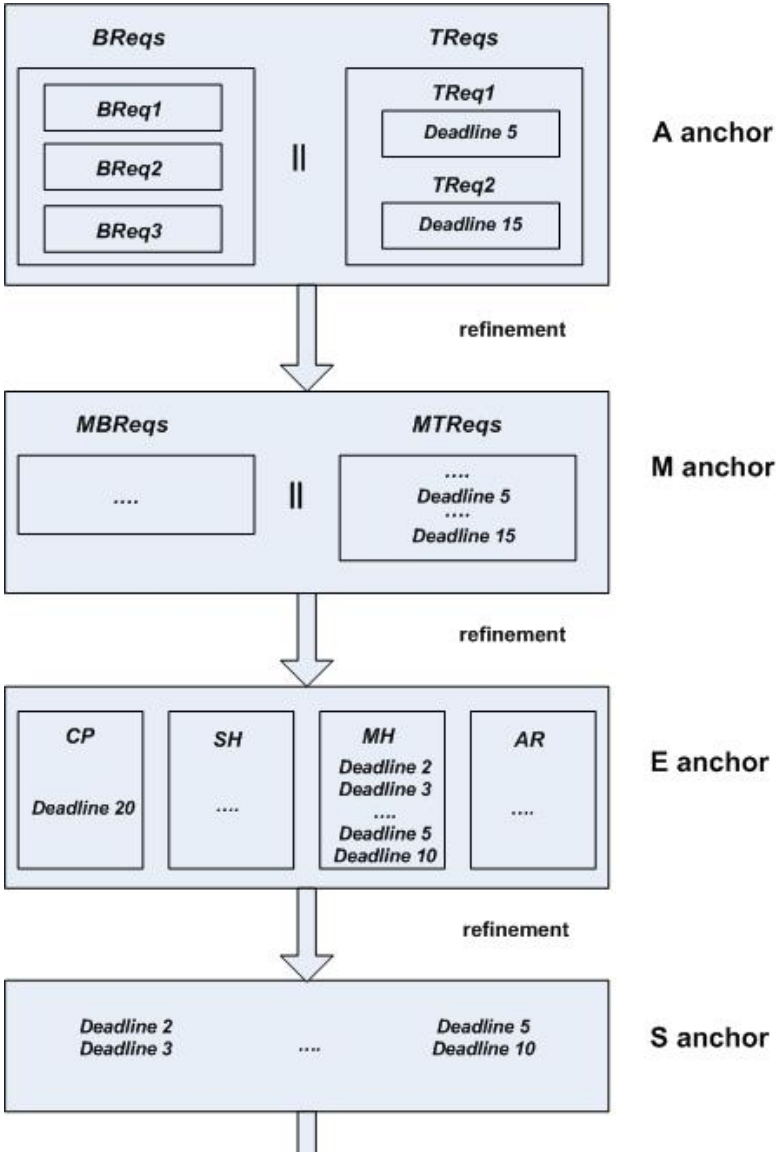
$$P \blacktriangleright d$$

$$\sqsubseteq$$

$$P_1 \blacktriangleright d_1 \,;\, P_2 \blacktriangleright d_2$$

$$\text{provided } P \sqsubseteq P_1 \,;\, P_2 \wedge d = d_1 + d_2$$

Refinement Strategy



$$P \blacktriangleright d$$

$$\sqsubseteq$$

$$P_1 \blacktriangleright d_1 \,;\, P_2 \blacktriangleright d_2$$

$$\textit{provided}\ P \sqsubseteq P_1 \,;\, P_2 \wedge d = d_1 + d_2$$

Conclusion and Future work

- We have developed a new version of *Circus Time* to describe timing behaviour of SCJ programs.
- New reactive-design semantics has been developed as well.
- 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 Time* in a theorem prover.
- Collapsing parallelism
- Refinement laws