

PARS: A Process Algebra with Resources and Schedulers

MohammadReza Mousavi, Michel Reniers
Twan Basten, Michel Chaudron

Eindhoven University of Technology

FORMATS'03,
Marseille, France

Overview

1. Motivation
2. A Process Algebra With Resources
3. Specifying Schedulers
4. Applying Schedulers to Processes
5. Conclusion and Future Works

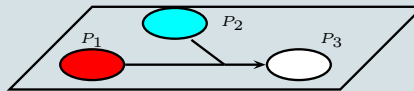
I. Motivation

1. **Scheduling theory:** a well-studied branch real-time systems research (dating back to 60's)
2. **Process algebra:** a formal approach to system design (since 70's)
3. No **formal link** between the two worlds until recently (e.g., RTSL of [Fredette and Cleaveland 93] and ACSR of [Gregoire and Lee 97])
4. Formal link between the two worlds is **not yet perfect**, e.g.,:
 - (a) Scheduling should be coded as priorities in ACSR
 - (b) There is no structure in scheduler specification in [Buchholtz et. al. 02]

I. Motivation

Our approach defines a language for:

1. specification of system behavior (with resource consumptions)



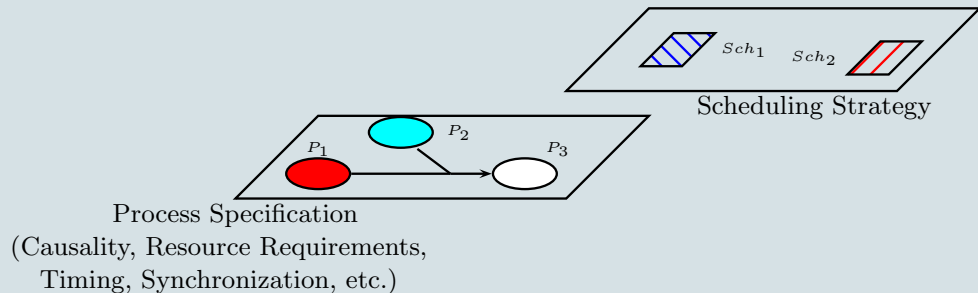
Process Specification

(Causality, Resource Requirements,
Timing, Synchronization, etc.)

I. Motivation

Our approach defines a language for:

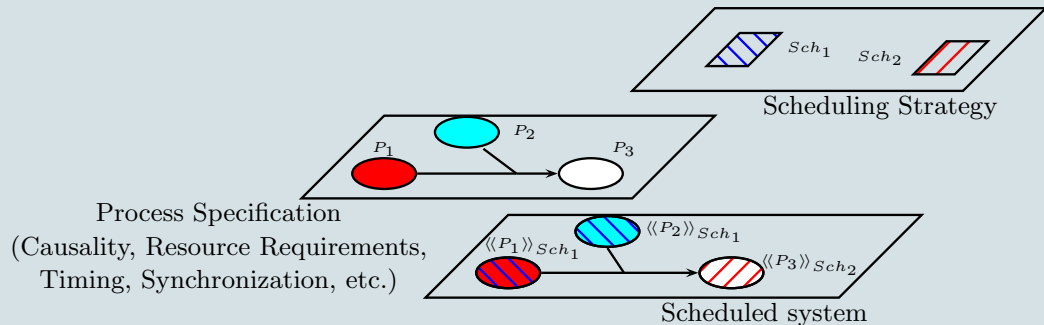
1. specification of system behavior (with resource consumptions)
2. specification of schedulers (resource provisions)



I. Motivation

Our approach defines a language for:

1. specification of system behavior (with resource consumptions)
2. specification of schedulers (resource provisions)
3. composing processes with schedulers



I. Motivation

Benefits of using PARS:

1. **independent** and **process algebraic** specification of processes and schedulers

$$\langle\langle P \rangle\rangle_{Sch}$$

2. applying schedulers to processes at **different levels**

$$\langle\langle\langle P_1 \rangle\rangle_{Sch_1} \parallel \langle\langle P_2 \rangle\rangle_{Sch_2} \rangle\rangle_{Sch_3}$$

3. embedding the **schedulability analysis** in **formal verification** process

Overview

✓ Motivation

2. A Process Algebra With Resources
3. Specifying Schedulers
4. Applying Schedulers to Processes
5. Conclusion and Future Works

2. A Process Algebra with Resources

1. durational atomic actions with dense time domain

$$\begin{array}{l} \text{decode}(2) \xrightarrow{0.5} \text{decode}(1.5) \dots \\ \text{decode}(2) \xrightarrow{1} \text{decode}(1) \dots \\ \text{decode}(2) \xrightarrow{2} \text{decode}(0) \xrightarrow{\text{decode}} \epsilon(0)\checkmark \end{array}$$

2. A Process Algebra with Resources

1. durational atomic actions with dense time domain
2. attaching resource consumption to basic actions

$$\begin{array}{r}
 (decode, \{Mem \mapsto 100\})(2) \quad \begin{array}{l} \{Mem \mapsto 100\}, 0.5 \\ \longrightarrow \quad \dots \\ \{Mem \mapsto 100\}, 1 \\ \longrightarrow \quad \dots \\ \{Mem \mapsto 100\}, 2 \\ \longrightarrow \quad \dots \end{array}
 \end{array}$$

2. A Process Algebra with Resources

1. durational atomic actions with dense time domain
2. attaching resource consumption to basic actions
3. relative commitment deadline operator

$$\sigma_1((\text{decode}, \{Mem \mapsto 100\})(2))$$

$$\begin{array}{l} \{Mem \mapsto 100\}, 0.5 \\ \rightarrow \end{array} \sigma_{0.5}((\text{decode}, \{Mem \mapsto 100\})(1.5))$$

$$\begin{array}{l} \{Mem \mapsto 100\}, 0.5 \\ \rightarrow \end{array} \sigma_0((\text{decode}, \{Mem \mapsto 100\})(1))$$

deadlock

2. A Process Algebra with Resources

1. durational atomic actions with dense time domain
2. attaching resource consumption to basic actions
3. relative commitment deadline operator
4. grouping processes using identifiers

$$d : (\text{decode}, \{Mem \mapsto 100\})(2) \quad [(D, \{Mem \mapsto 100\}), 1] \dots$$

$$D = (d, Dl, WCET)$$

2. A Process Algebra with Resources

1. durational atomic actions with dense time domain
2. attaching resource consumption to basic actions
3. relative commitment deadline operator
4. grouping processes using identifiers
5. asynchronous and synchronous time parallelism

$$\begin{array}{l}
 d : (\text{decode}, \{Mem \mapsto 100\})(2) \\
 \quad \quad \quad \parallel \\
 e : (\text{encode}, \{Mem \mapsto 200\})(2)
 \end{array}
 \begin{array}{l}
 \begin{array}{l}
 \xrightarrow{[(D, \{Mem \mapsto 100\}), 1]} \\
 \dots \\
 \xrightarrow{[(D, \{Mem \mapsto 100\}), (E, \{Mem \mapsto 200\}), 1]} \\
 \dots \\
 \xrightarrow{[(E, \{Mem \mapsto 200\}), 1]} \\
 \dots
 \end{array}
 \end{array}$$

2. A Process Algebra with Resources

An example:

$$P = \mu X.((a, \{CPU \mapsto 1, Mem \mapsto 100\})(t) \parallel \epsilon(t') ; X)$$

a periodic process with duration t and period t'

2. A Process Algebra with Resources

An example:

$$P = \mu X.((a, \{CPU \mapsto 1, Mem \mapsto 100\})(t) \parallel \epsilon(t') ; X)$$

$$S = \mu X.((b, \{CPU \mapsto 1\})(t) \parallel \int_{t \in \mathbb{R}^{\geq}} \epsilon(t) ; X)$$

a sporadic process with duration t

2. A Process Algebra with Resources

An example:

$$P = \mu X.((a, \{CPU \mapsto 1, Mem \mapsto 100\})(t) \parallel \epsilon(t') ; X)$$

$$S = \mu X.((b, \{CPU \mapsto 1\})(t) \parallel \int_{t \in \mathbb{R}^{\geq}} \epsilon(t) ; X)$$

$$SysProc = System : (S) \parallel User : (P)$$

a system consisting of
periodic user processes and sporadic system processes

Overview

- ✓ Motivation
- ✓ A Process Algebra With Resources
- 3. Specifying Schedulers
- 4. Applying Schedulers to Processes
- 5. Conclusion and Future Works

3. Specifying Schedulers

1. durational atomic actions providing resources

$$\{CPU \mapsto -1\}(2)$$

3. Specifying Schedulers

1. durational atomic actions providing resources
2. decorated with predicates specifying eligible processes

$$(\underline{Id}.id = d, \{CPU \mapsto -1\})(2)$$

3. Specifying Schedulers

1. durational atomic actions providing resources
2. decorated with predicates specifying eligible processes
3. precedence operator \triangleright

$$\begin{aligned} & (\underline{Id}.id = System, \{CPU \mapsto -1\})(2) \\ \triangleright & (\underline{Id}.id = User, \{CPU \mapsto -1\})(2) \end{aligned}$$

3. Specifying Schedulers

1. durational atomic actions providing resources
2. decorated with predicates specifying eligible processes
3. precedence operator \triangleright
4. (time) quantified precedence (choice) operator

$$\bigvee_{t \in \mathbb{R}^{\geq}} (\underline{Id}.Dl = t, \{CPU \mapsto -1\})(t)$$

3. Specifying Schedulers

An example:

$$SysProc = P_0 \parallel P_1 \parallel \dots \parallel P_n$$

$$P_i = \mu X.(2i + 1) : ((a_i, \rho_i)(t)) \parallel (2i) : (\epsilon(t')) ; X$$

periods are tagged with $2i$ and processes with $2i + 1$

3. Specifying Schedulers

An example:

$$SysProc = P_0 \parallel P_1 \parallel \dots \parallel P_n$$

$$P_i = \mu X. (2i + 1) : ((a_i, \rho_i)(t)) \parallel (2i) : (\epsilon(t')) ; X$$

$$RMSch(i, t) = (\underline{Id}.id = 2i + 1 \wedge Id_0.id = 2i \wedge Id_0.WCET = t, \{CPU \mapsto 1\})([0, \infty))$$

$$RMSch = \bigvee_{t \in \mathbb{R}^{\geq 0}} (RMSch(0, t) + \dots + RMSch(n, t))$$

a *Rate Monotonic Scheduler* giving priority to processes with higher rates (shorter periods)

3. Specifying Schedulers

Semantics: defining a symbolic transition system with

1. all possible resource provisions
2. positive predicates specifying eligible processes
3. negative predicates specifying processes that should not appear in the context (disallowing lower priority processes from taking over higher priority ones)

Overview

- ✓ Motivation
- ✓ A Process Algebra With Resources
- ✓ Specifying Schedulers
- 4. Applying Schedulers to Processes
- 5. Conclusion and Future Works

4. Applying Schedulers to Processes

$$\begin{array}{ccc}
 P & \xrightarrow{[(Ids, \rho)], t} & P' \\
 Sch & \xrightarrow{[(pred, npred, \bar{\rho})], t} & Sch'
 \end{array}$$

$$\langle\langle P \rangle\rangle_{Sch} \xrightarrow{[Ids, \rho \oplus \bar{\rho}], t} \langle\langle P' \rangle\rangle_{Sch'}$$

if:

1. there exists an $id \in Ids$ that satisfies $pred$
2. P cannot make a transition that satisfies $npred$

4. Applying Schedulers to Processes

$$\begin{aligned} Proc = 1 & : (\sigma_1(a, \{CPU \mapsto 1, Mem \mapsto 100\}(1))) \parallel \\ & 2 : (\sigma_2(b, \{CPU \mapsto 1, Mem \mapsto 100\}(2))) \end{aligned}$$

$$EDF_1 = \mu X. (\bigvee_{t \in \mathbb{R}^{\geq 0}} (\underline{Id}.Dl = t) \{CPU \mapsto -2, Mem \mapsto -200\}(2)) ; X$$

$$Sys_1 = \langle\langle Proc \rangle\rangle_{EDF_1}$$

an *Earliest Deadline First* scheduling
results in deadlock for the process number 2

4. Applying Schedulers to Processes

$$Proc = 1 : (\sigma_1(a, \{CPU \mapsto 1, Mem \mapsto 100\}(1))) \parallel \\ 2 : (\sigma_2(b, \{CPU \mapsto 1, Mem \mapsto 100\}(2)))$$

$$EDF_2 = \mu X. (\bigvee_{x_t \in \mathbb{R}^{\geq 0}} ((\underline{Id}.Dl = x_t) \{CPU \mapsto -1, Mem \mapsto -100\}(2)) \parallel \parallel \\ \bigvee_{x_t \in \mathbb{R}^{\geq 0}} ((\underline{Id}.Dl = x_t) \{CPU \mapsto -1, Mem \mapsto -100\}(2)))$$

$$Sys_2 = \langle\langle Proc \rangle\rangle_{EDF_2}$$

another *Earliest Deadline First* scheduling
results in successful termination of both processes

Overview

- ✓ Motivation
- ✓ A Process Algebra With Resources
- ✓ Specifying Schedulers
- ✓ Applying Schedulers to Processes
- 5. Conclusion and Future Works

5. Conclusion and Future Works

What we have done:

Defining a process algebraic language for specification of scheduled systems

with the following features:

1. resource requiring processes
2. resource providing schedulers
3. asynchronous parallel composition (leaving the ordering of processes to schedulers)

5. Conclusion and Future Works

Ongoing research:

1. axiomatizing PARS
2. translating PARS to a model checker input format
3. implementing a realistic case study (an application prototype from avionics domain)

For more information, see:

<http://www.win.tue.nl/~mousavi/sacc/>