Concurrency Theory

Lecture 22: Timed Modelling & Conclusions

Joost-Pieter Katoen Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH AACHEN UNIVERSITY

{katoen,noll}@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/ct13/

Winter Semester 2013/14

Wanted: Software Engineering HiWis

- What we offer: work in
  - ESA project HASDEL
    - Hardware-Software Dependability for Launchers
    - successor of COMPASS project (compass.informatik.rwth-aachen.de)
    - goal: enhance COMPASS for rocket design validation
  - EU project D-MILS
    - Dependability and Security of Distributed Information and Communication Infrastructures
    - design and implementation of high-level specification language
- What we expect: prospective candidates
  - like formal methods (model checking, program/model transformations)
  - program efficiently (Python)
  - work 9–19 hrs/week
- Contact: Thomas Noll (noll@cs.rwth-aachen.de)

Outline

So far: "Qualitative" Modelling

- Algebraic language (CCS) for syntactic description of concurrent systems
- Meaning given by LTSs that define dynamic behaviour of process terms
- Structural operational semantics for mapping CCS processes to LTSs
- Notions of behavioural equivalence (trace equivalence, bisimilarity) for comparing process behaviours
- Modal logics (HML) to specify desired system properties
- Petri Nets as model of true concurrency with partial-order semantics

So far: “Qualitative” Modelling

- Algebraic language (CCS) for syntactic description of concurrent systems
- Meaning given by LTSs that define dynamic behaviour of process terms
- Structural operational semantics for mapping CCS processes to LTSs
- Notions of behavioural equivalence (trace equivalence, bisimilarity) for comparing process behaviours
- Modal logics (HML) to specify desired system properties
- Petri Nets as model of true concurrency with partial-order semantics

$\Rightarrow$ very abstract (if any) notion of time:
logical order of computation steps

Real-Time Reactive Systems

Example 22.1 (Real-time reactive systems)

- brake systems and airbags in cars
- plant controls
- mobile phones
- ...

Real-Time Reactive Systems

Example 22.1 (Real-time reactive systems)

- brake systems and airbags in cars
- plant controls
- mobile phones
- ...

Real-time requirements

The correct behaviour of a real-time system does not only depend on the logical order in which events are performed but also on their timing.

Real-Time Reactive Systems

Example 22.1 (Real-time reactive systems)

- brake systems and airbags in cars
- plant controls
- mobile phones
- ...

Real-time requirements

The correct behaviour of a real-time system does not only depend on the logical order in which events are performed but also on their timing.

Example 22.2 (Untimed vs. timed)

- Untimed: "if the car crashes, eventually the airbag will be inflated"
- Timed: "if the car crashes, the airbag must be inflated within 50 milliseconds"

Theory of Real-Time Systems

Extensive research work on formal methods for real-time systems:

- **Modelling**
  - extensions of CCS: Timed CCS (TCCS; Yi 1990), Temporal Process Algebra (Hennessy/Regan 1995), Temporal CCS (Moller/Tofts 1990)
  - extensions of other untimed process algebras (ACP, CSP)
  - timed automata (Alur/Dill 1990)

Theory of Real-Time Systems

Extensive research work on formal methods for real-time systems:

- **Modelling**
  - extensions of CCS: Timed CCS (TCCS; Yi 1990), Temporal Process Algebra (Hennessy/Regan 1995), Temporal CCS (Moller/Tofts 1990)
  - extensions of other untimed process algebras (ACP, CSP)
  - timed automata (Alur/Dill 1990)
- **Requirement specification**
  - HML with time (Laroussinie et al. 1990)
  - extensions of LTL: Timed Propositional Temporal Logic (TPTL; Alur/Henzinger 1994), Metric Temporal Logic (MTL; Koymans 1990)
  - extension of CTL: Timed Computation Tree Logic (TCTL; Alur et al. 1993)

Theory of Real-Time Systems

Extensive research work on formal methods for real-time systems:

- **Modelling**
  - extensions of CCS: Timed CCS (TCCS; Yi 1990), Temporal Process Algebra (Hennessy/Regan 1995), Temporal CCS (Moller/Tofts 1990)
  - extensions of other untimed process algebras (ACP, CSP)
  - timed automata (Alur/Dill 1990)
- **Requirement specification**
  - HML with time (Laroussinie et al. 1990)
  - extensions of LTL: Timed Propositional Temporal Logic (TPTL; Alur/Henzinger 1994), Metric Temporal Logic (MTL; Koymans 1990)
  - extension of CTL: Timed Computation Tree Logic (TCTL; Alur et al. 1993)
- **Analysis**
  - timed behavioural equivalences (timed trace equivalence, timed bisimilarity)
  - abstraction of timed automata via regions and zones

Theory of Real-Time Systems

Extensive research work on formal methods for real-time systems:

- **Modelling**
  - extensions of CCS: Timed CCS (TCCS; Yi 1990), Temporal Process Algebra (Hennessy/Regan 1995), Temporal CCS (Moller/Tofts 1990)
  - extensions of other untimed process algebras (ACP, CSP)
  - timed automata (Alur/Dill 1990)
- **Requirement specification**
  - HML with time (Laroussinie et al. 1990)
  - extensions of LTL: Timed Propositional Temporal Logic (TPTL; Alur/Henzinger 1994), Metric Temporal Logic (MTL; Koymans 1990)
  - extension of CTL: Timed Computation Tree Logic (TCTL; Alur et al. 1993)
- **Analysis**
  - timed behavioural equivalences (timed trace equivalence, timed bisimilarity)
  - abstraction of timed automata via regions and zones
- Here: syntax and semantics of Timed CCS

Outline

Example 22.3 (Light switch)

1. If the switch is off, and is pressed once, then the light will turn on.
2. If the switch is pressed again “soon” after the light was turned on, the light becomes brighter. Otherwise, the light is turned off by the next button press.
3. The light is also turned off by a button press when it is bright.

Example 22.3 (Light switch)

1. If the switch is off, and is pressed once, then the light will turn on.
   - in CCS: $\mathit{Off} = \mathit{press}.\mathit{Light}$
2. If the switch is pressed again "soon" after the light was turned on, the light becomes brighter. Otherwise, the light is turned off by the next button press.
3. The light is also turned off by a button press when it is bright.
   - in CCS: $\mathit{Bright} = \mathit{press}.\mathit{Off}$

Example 22.3 (Light switch)

1. If the switch is off, and is pressed once, then the light will turn on.
   - in CCS: $\mathit{Off} = \mathit{press}.\mathit{Light}$
2. If the switch is pressed again "soon" after the light was turned on, the light becomes brighter. Otherwise, the light is turned off by the next button press.
   - in CCS: $\mathit{Light} = \mathit{press}.\mathit{Bright} + \tau.\mathit{press}.\mathit{Off}$
   - but: does not properly capture the "soon" requirement
   - rather: system may internally choose to switch off light after next button press (after "timeout" action τ)
3. The light is also turned off by a button press when it is bright.
   - in CCS: $\mathit{Bright} = \mathit{press}.\mathit{Off}$

Timed Modelling

Modelling with time delays

$$Light = press.Bright + \varepsilon(1.5).\tau.press.Off$$

- passage of time viewed as "action" performed by system
- specified by new prefixing operator $\varepsilon(d).P$ where $d \in \mathbb{R}_{\geq 0}$ gives amount of time that needs to elapse before $P \in Prc$ is enabled
- thus: "soon" interpreted as "within 1.5 time units"
- use of τ is crucial here: must be performed when enabled (details later)

Timed Labelled Transition Systems I

Definition 22.4 (Timed labelled transition system)

A timed labelled transition system (TLTS) is a triple $(S, Lab, \longrightarrow)$ consisting of

- a set S of states
- a set $Lab = Act \cup \mathbb{R}_{\geq 0}$ of labels
  - actions $a \in Act$
  - time delays $d \in \mathbb{R}_{\geq 0}$
- a transition relation $\longrightarrow \subseteq S \times Lab \times S$ (written $s \xrightarrow{\lambda} s'$)

Timed Labelled Transition Systems I

Definition 22.4 (Timed labelled transition system)

A timed labelled transition system (TLTS) is a triple $(S, Lab, \longrightarrow)$ consisting of

- a set S of states
- a set $Lab = Act \cup \mathbb{R}_{\geq 0}$ of labels
  - actions $a \in Act$
  - time delays $d \in \mathbb{R}_{\geq 0}$
- a transition relation $\longrightarrow \subseteq S \times Lab \times S$ (written $s \xrightarrow{\lambda} s'$)

Additional requirements:

- time additivity: if $s \xrightarrow{d} s'$ and $0 \leq d' \leq d$, then $s \xrightarrow{d'} s'' \xrightarrow{d-d'} s'$ for some $s'' \in S$
- self-reachability without delay: $s \xrightarrow{0} s$ for each $s \in S$
- time determinism: if $s \xrightarrow{d} s'$ and $s \xrightarrow{d} s''$, then $s' = s''$

Timed Labelled Transition Systems II

Example 22.5 (Timed labelled transition system)

$$(S, Lab, \longrightarrow)$$

where

- $S = \mathbb{R}_{\geq 0}$
- $Lab = \{a\} \cup \mathbb{R}_{\geq 0}$
- $\xrightarrow{a} = \{(5, 0)\}$
- for all $d \in \mathbb{R}_{\geq 0}$: $\xrightarrow{d} = \{(s, s + d) \mid s \in \mathbb{R}_{\geq 0}\}$

(diagram on the board)

Outline

Definition 22.6 (Syntax of TCCS (cf. Definition 2.1))

- Let A be a set of (action) names.

Definition 22.6 (Syntax of TCCS (cf. Definition 2.1))

- Let A be a set of (action) names.
- $\overline{A} := \{\overline{a} \mid a \in A\}$ denotes the set of co-names.

Syntax of Timed CCS I

Definition 22.6 (Syntax of TCCS (cf. Definition 2.1))

- Let A be a set of (action) names.
- $\overline{A} := \{\overline{a} \mid a \in A\}$ denotes the set of co-names.
- $Act := A \cup \overline{A} \cup \{\tau\}$ is the set of actions where τ denotes the silent (or: unobservable) action.

Syntax of Timed CCS I

Definition 22.6 (Syntax of TCCS (cf. Definition 2.1))

- Let A be a set of (action) names.
- $\overline{A} := \{\overline{a} \mid a \in A\}$ denotes the set of co-names.
- $Act := A \cup \overline{A} \cup \{\tau\}$ is the set of actions where τ denotes the silent (or: unobservable) action.
- Let Pid be a set of process identifiers.

Syntax of Timed CCS I

Definition 22.6 (Syntax of TCCS (cf. Definition 2.1))

- Let A be a set of (action) names.
- $\overline{A} := \{\overline{a} \mid a \in A\}$ denotes the set of co-names.
- $Act := A \cup \overline{A} \cup \{\tau\}$ is the set of actions where τ denotes the silent (or: unobservable) action.
- Let Pid be a set of process identifiers.
- The set Prc of process expressions is defined by the following syntax:

$$
\begin{array}{rcll}
P & ::= & \text{nil} & \text{(inaction)} \\
& | & \alpha.P & \text{(prefixing)} \\
& | & P_1 + P_2 & \text{(choice)} \\
& | & P_1 \parallel P_2 & \text{(parallel composition)} \\
& | & P \setminus L & \text{(restriction)} \\
& | & P[f] & \text{(relabelling)} \\
& | & C & \text{(process call)} \\
& | & \varepsilon(d).P & \text{(time delay)}
\end{array}
$$

where $\alpha \in Act$, $d \in \mathbb{R}_{\geq 0}$, $L \subseteq A$, $C \in Pid$, and $f : Act \to Act$ such that $f(\tau) = \tau$ and $f(\overline{a}) = \overline{f(a)}$ for each $a \in A$.

Syntax of Timed CCS II

Definition 22.6 (continued)

- A (recursive) process definition is an equation system of the form
$$(C_i = P_i \mid 1 \leq i \leq k)$$
where $k \geq 1$, $C_i \in \mathit{Pid}$ (pairwise distinct), and $P_i \in \mathit{Prc}$ (with process identifiers from $\{C_1, \ldots, C_k\}$).
- An occurrence of a process identifier $C \in \mathit{Pid}$ in an expression $P \in \mathit{Prc}$ is guarded if it occurs within a subexpression of P of the form $\lambda.Q$ where $\lambda \in \mathit{Act}$ or $\lambda = \varepsilon(d)$ for some $d > 0$
- A process expression/definition is guarded if all occurrences of process identifiers are guarded

Syntax of Timed CCS II

Definition 22.6 (continued)

- A (recursive) process definition is an equation system of the form

$$(C_i = P_i \mid 1 \leq i \leq k)$$

  where $k \geq 1$, $C_i \in Pid$ (pairwise distinct), and $P_i \in Prc$ (with process identifiers from $\{C_1, \ldots, C_k\}$).
- An occurrence of a process identifier $C \in Pid$ in an expression $P \in Prc$ is guarded if it occurs within a subexpression of P of the form $\lambda.Q$ where $\lambda \in Act$ or $\lambda = \varepsilon(d)$ for some $d > 0$
- A process expression/definition is guarded if all occurrences of process identifiers are guarded

Conventions:

- Processes P and $\varepsilon(0).P$ will not be distinguished
- All process definitions have to be guarded

Syntax of Timed CCS III

Example 22.7

1. $(a.C_1 + (C_2 \parallel b.C_3) + C_1) \parallel (\varepsilon(4.2).(C_4 \parallel \mathsf{nil}) + \varepsilon(1.2).C_3)$
   - first occurrence of C_1 is guarded, second unguarded
   - occurrence of C_2 is unguarded
   - both occurrences of C_3 are guarded
   - occurrence of C_4 is guarded
   - overall expression is unguarded

Syntax of Timed CCS III

Example 22.7

1. $(a.C_1 + (C_2 \parallel b.C_3) + C_1) \parallel (\varepsilon(4.2).(C_4 \parallel \mathsf{nil}) + \varepsilon(1.2).C_3)$
   - first occurrence of C_1 is guarded, second unguarded
   - occurrence of C_2 is unguarded
   - both occurrences of C_3 are guarded
   - occurrence of C_4 is guarded
   - overall expression is unguarded
2. $$\begin{aligned} \mathit{Off} &= \mathit{press}.\mathit{Light} \\ \mathit{Bright} &= \mathit{press}.\mathit{Off} \\ \mathit{Light} &= \mathit{press}.\mathit{Bright} + \varepsilon(1.5).\tau.\mathit{press}.\mathit{Off} \end{aligned}$$

   is guarded

Outline

Definition 22.8 (Semantics of TCCS – action transitions; cf. Def. 2.4)

A process definition $(C_i = P_i \mid 1 \leq i \leq k)$ determines the TLTS $(Prc, Lab, \longrightarrow)$ whose transitions can be inferred from the following rules ($P, P', Q, Q' \in Prc$, $\alpha \in Act$, $\lambda \in A \cup \overline{A}$, $a \in A$):

$$\text{(Del)}\ \frac{P \xrightarrow{\alpha} P'}{\varepsilon(0).P \xrightarrow{\alpha} P'} \qquad \text{(Act)}\ \frac{}{\alpha.P \xrightarrow{\alpha} P}$$

$$\text{(Sum}_1\text{)}\ \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \text{(Sum}_2\text{)}\ \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

$$\text{(Par}_1\text{)}\ \frac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q} \qquad \text{(Par}_2\text{)}\ \frac{Q \xrightarrow{\alpha} Q'}{P \parallel Q \xrightarrow{\alpha} P \parallel Q'}$$

$$\text{(Com)}\ \frac{P \xrightarrow{\lambda} P' \quad Q \xrightarrow{\overline{\lambda}} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} \qquad \text{(Res)}\ \frac{P \xrightarrow{\alpha} P' \quad (\alpha, \overline{\alpha} \notin L)}{P \setminus L \xrightarrow{\alpha} P' \setminus L}$$

$$\text{(Rel)}\ \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]} \qquad \text{(Call)}\ \frac{P \xrightarrow{\alpha} P' \quad (C = P)}{C \xrightarrow{\alpha} P'}$$

Definition 22.8 (Semantics of TCCS – timed transitions)

Additionally for $d, d' \in \mathbb{R}_{\geq 0}$:

$$\text{(tAdd)}\ \frac{P \xrightarrow{d'} P'}{\varepsilon(d).P \xrightarrow{d+d'} P'} \qquad \text{(tSub)}\ \frac{(d' \leq d)}{\varepsilon(d).P \xrightarrow{d'} \varepsilon(d-d').P'}$$

$$\text{(tAct)}\ \frac{(\alpha \neq \tau)}{\alpha.P \xrightarrow{d} \alpha.P} \qquad \text{(tTau)}\ \frac{}{\tau.P \xrightarrow{0} \tau.P}$$

$$\text{(tSum)}\ \frac{P \xrightarrow{d} P' \quad Q \xrightarrow{d} Q'}{P + Q \xrightarrow{d} P' + Q'} \qquad \text{(tRes)}\ \frac{P \xrightarrow{d} P'}{P \setminus L \xrightarrow{d} P' \setminus L}$$

$$\text{(tRel)}\ \frac{P \xrightarrow{d} P'}{P[f] \xrightarrow{d} P'[f]} \qquad \text{(tCall)}\ \frac{P \xrightarrow{d} P' \quad (C = P)}{C \xrightarrow{d} P'}$$

Definition 22.8 (Semantics of TCCS – timed transitions)

Additionally for $d, d' \in \mathbb{R}_{\geq 0}$:

$$\text{(tAdd)}\ \frac{P \xrightarrow{d'} P'}{\varepsilon(d).P \xrightarrow{d+d'} P'} \qquad \text{(tSub)}\ \frac{(d' \leq d)}{\varepsilon(d).P \xrightarrow{d'} \varepsilon(d-d').P'}$$

$$\text{(tAct)}\ \frac{(\alpha \neq \tau)}{\alpha.P \xrightarrow{d} \alpha.P} \qquad \text{(tTau)}\ \frac{}{\tau.P \xrightarrow{0} \tau.P}$$

$$\text{(tSum)}\ \frac{P \xrightarrow{d} P' \quad Q \xrightarrow{d} Q'}{P + Q \xrightarrow{d} P' + Q'} \qquad \text{(tRes)}\ \frac{P \xrightarrow{d} P'}{P \setminus L \xrightarrow{d} P' \setminus L}$$

$$\text{(tRel)}\ \frac{P \xrightarrow{d} P'}{P[f] \xrightarrow{d} P'[f]} \qquad \text{(tCall)}\ \frac{P \xrightarrow{d} P' \quad (C = P)}{C \xrightarrow{d} P'}$$

Remarks:

- parallel composition considered later
- delay transitions do *not* resolve non-deterministic choices (according to time-determinism property of Definition 22.4)

Example 22.9

$$\begin{aligned} \mathit{Off} &= \mathit{press}.\mathit{Light} \\ \mathit{Bright} &= \mathit{press}.\mathit{Off} \\ \mathit{Light} &= \mathit{press}.\mathit{Bright} + \varepsilon(1.5).\tau.\mathit{press}.\mathit{Off} \end{aligned}$$

Example 22.9

$$\begin{aligned} Off &= press.Light \\ Bright &= press.Off \\ Light &= press.Bright + \varepsilon(1.5).\tau.press.Off \end{aligned}$$

1. $Light \xrightarrow{press} Bright$

Example 22.9

$$
\begin{aligned}
\mathit{Off} &= \mathit{press}.\mathit{Light}\\
\mathit{Bright} &= \mathit{press}.\mathit{Off}\\
\mathit{Light} &= \mathit{press}.\mathit{Bright} + \varepsilon(1.5).\tau.\mathit{press}.\mathit{Off}
\end{aligned}
$$

1. $\mathit{Light} \xrightarrow{\mathit{press}} \mathit{Bright}$
2. for all $0 \leq d \leq 1.5$: $\mathit{Light} \xrightarrow{d} \mathit{press}.\mathit{Bright} + \varepsilon(1.5 - d).\tau.\mathit{press}.\mathit{Off}$

Semantics of Timed CCS III

Example 22.9

$$\begin{aligned} \mathit{Off} &= \mathit{press}.\mathit{Light} \\ \mathit{Bright} &= \mathit{press}.\mathit{Off} \\ \mathit{Light} &= \mathit{press}.\mathit{Bright} + \varepsilon(1.5).\tau.\mathit{press}.\mathit{Off} \end{aligned}$$

1. $\mathit{Light} \xrightarrow{\mathit{press}} \mathit{Bright}$
2. for all $0 \leq d \leq 1.5$: $\mathit{Light} \xrightarrow{d} \mathit{press}.\mathit{Bright} + \varepsilon(1.5 - d).\tau.\mathit{press}.\mathit{Off}$
3. especially for $d = 1.5$: $\mathit{Light} \xrightarrow{1.5} \mathit{press}.\mathit{Bright} + \varepsilon(0).\tau.\mathit{press}.\mathit{Off}$
   $\xrightarrow{\tau} \mathit{press}.\mathit{Off}$
   $\xrightarrow{d'} \mathit{press}.\mathit{Off}$
   $\xrightarrow{\mathit{press}} \mathit{Off}$

   (for all $d' \in \mathbb{R}_{\geq 0}$)

Example 22.9

$$\begin{aligned} Off &= press.Light \\ Bright &= press.Off \\ Light &= press.Bright + \varepsilon(1.5).\tau.press.Off \end{aligned}$$

1. $Light \xrightarrow{press} Bright$
2. for all $0 \leq d \leq 1.5$: $Light \xrightarrow{d} press.Bright + \varepsilon(1.5 - d).\tau.press.Off$
3. especially for $d = 1.5$: $Light \xrightarrow{1.5} press.Bright + \varepsilon(0).\tau.press.Off \xrightarrow{\tau} press.Off \xrightarrow{d'} press.Off \xrightarrow{press} Off$

   (for all $d' \in \mathbb{R}_{\geq 0}$)
4. moreover: $press.Bright + \varepsilon(0).\tau.press.Off \not\xrightarrow{d}$ (for any $d > 0$)
   $\implies$ first alternative only enabled up to time point 1.5

Properties of the Semantics

Lemma 22.10 (cf. Definition 22.4)

1. *time additivity:* if $P \xrightarrow{d} P'$ and $0 \leq d' \leq d$, then $P \xrightarrow{d'} P'' \xrightarrow{d-d'} P'$ for some $P'' \in Prc$
2. *self-reachability without delay:* $P \xrightarrow{0} P$ for each $P \in Prc$
3. *time determinism:* if $P \xrightarrow{d} P'$ and $P \xrightarrow{d} P''$, then $P' = P''$
4. *persistency of action transitions:* for all $P, Q \in Prc$, $\alpha \in Act$ and $d \in \mathbb{R}_{\geq 0}$, if $P \xrightarrow{\alpha}$ and $P \xrightarrow{d} Q$, then $Q \xrightarrow{\alpha}$

(1)–(3) implies that the semantics of a TCCS process is indeed a TLTS.

Properties of the Semantics

Lemma 22.10 (cf. Definition 22.4)

1. *time additivity:* if $P \xrightarrow{d} P'$ and $0 \leq d' \leq d$, then $P \xrightarrow{d'} P'' \xrightarrow{d-d'} P'$ for some $P'' \in Prc$
2. *self-reachability without delay:* $P \xrightarrow{0} P$ for each $P \in Prc$
3. *time determinism:* if $P \xrightarrow{d} P'$ and $P \xrightarrow{d} P''$, then $P' = P''$
4. *persistency of action transitions:* for all $P, Q \in Prc$, $\alpha \in Act$ and $d \in \mathbb{R}_{\geq 0}$, if $P \xrightarrow{\alpha}$ and $P \xrightarrow{d} Q$, then $Q \xrightarrow{\alpha}$

(1)–(3) implies that the semantics of a TCCS process is indeed a TLTS.

Proof.

1. by Rules (tAdd), (tSub) and (tAct)
2. by Rules (tSub), (tAct) and (tTau) (note that every P is guarded)
3. by induction on derivation tree
4. by induction on derivation tree

Outline

The Light Switch Example Revisited

Example 22.11

$$\begin{aligned}
Off &= press.Light \\
Bright &= press.Off \\
Light &= press.Bright + \varepsilon(1.5).\tau.press.Off \\
FastUser &= \overline{press}.\varepsilon(0.3).\overline{press}.FastUser
\end{aligned}$$

The Light Switch Example Revisited

Example 22.11

$$\begin{aligned} \mathit{Off} &= \mathit{press}.\mathit{Light} \\ \mathit{Bright} &= \mathit{press}.\mathit{Off} \\ \mathit{Light} &= \mathit{press}.\mathit{Bright} + \varepsilon(1.5).\tau.\mathit{press}.\mathit{Off} \\ \mathit{FastUser} &= \overline{\mathit{press}}.\varepsilon(0.3).\overline{\mathit{press}}.\mathit{FastUser} \end{aligned}$$

- Expect immediate synchronisation between $\mathit{FastUser}$ and Off:
  $(\mathit{FastUser} \parallel \mathit{Off}) \setminus \mathit{press} \xrightarrow{\tau} (\varepsilon(0.3).\overline{\mathit{press}}.\mathit{FastUser} \parallel \mathit{Light}) \setminus \mathit{press}$

The Light Switch Example Revisited

Example 22.11

$$\begin{aligned} \mathit{Off} &= \mathit{press}.\mathit{Light} \\ \mathit{Bright} &= \mathit{press}.\mathit{Off} \\ \mathit{Light} &= \mathit{press}.\mathit{Bright} + \varepsilon(1.5).\tau.\mathit{press}.\mathit{Off} \\ \mathit{FastUser} &= \overline{\mathit{press}}.\varepsilon(0.3).\overline{\mathit{press}}.\mathit{FastUser} \end{aligned}$$

- Expect immediate synchronisation between $\mathit{FastUser}$ and Off:
  $(\mathit{FastUser} \parallel \mathit{Off}) \setminus \mathit{press} \xrightarrow{\tau} (\varepsilon(0.3).\overline{\mathit{press}}.\mathit{FastUser} \parallel \mathit{Light}) \setminus \mathit{press}$
- Now $\overline{\mathit{press}}$-transition only enabled after 0.3 time units, which is also a possible delay for Light: $\mathit{Light} \xrightarrow{0.3} \mathit{press}.\mathit{Bright} + \varepsilon(1.2).\tau.\mathit{press}.\mathit{Off}$

The Light Switch Example Revisited

Example 22.11

$$\begin{aligned} \mathit{Off} &= \mathit{press}.\mathit{Light} \\ \mathit{Bright} &= \mathit{press}.\mathit{Off} \\ \mathit{Light} &= \mathit{press}.\mathit{Bright} + \varepsilon(1.5).\tau.\mathit{press}.\mathit{Off} \\ \mathit{FastUser} &= \overline{\mathit{press}}.\varepsilon(0.3).\overline{\mathit{press}}.\mathit{FastUser} \end{aligned}$$

- Expect immediate synchronisation between $\mathit{FastUser}$ and Off:
  $(\mathit{FastUser} \parallel \mathit{Off}) \setminus \mathit{press} \xrightarrow{\tau} (\varepsilon(0.3).\overline{\mathit{press}}.\mathit{FastUser} \parallel \mathit{Light}) \setminus \mathit{press}$
- Now $\overline{\mathit{press}}$-transition only enabled after 0.3 time units, which is also a possible delay for Light: $\mathit{Light} \xrightarrow{0.3} \mathit{press}.\mathit{Bright} + \varepsilon(1.2).\tau.\mathit{press}.\mathit{Off}$
- Therefore expected that whole system can delay:
  $((\varepsilon(0.3).\overline{\mathit{press}}.\mathit{FastUser}) \parallel \mathit{Light}) \setminus \mathit{press}$
  $\xrightarrow{0.3} ((\overline{\mathit{press}}.\mathit{FastUser}) \parallel (\mathit{press}.\mathit{Bright} + \varepsilon(1.2).\tau.\mathit{press}.\mathit{Off})) \setminus \mathit{press}$

The Light Switch Example Revisited

Example 22.11

$$\begin{aligned} \mathit{Off} &= \mathit{press}.\mathit{Light} \\ \mathit{Bright} &= \mathit{press}.\mathit{Off} \\ \mathit{Light} &= \mathit{press}.\mathit{Bright} + \varepsilon(1.5).\tau.\mathit{press}.\mathit{Off} \\ \mathit{FastUser} &= \overline{\mathit{press}}.\varepsilon(0.3).\overline{\mathit{press}}.\mathit{FastUser} \end{aligned}$$

- Expect immediate synchronisation between $\mathit{FastUser}$ and Off:
  $(\mathit{FastUser} \parallel \mathit{Off}) \setminus \mathit{press} \xrightarrow{\tau} (\varepsilon(0.3).\overline{\mathit{press}}.\mathit{FastUser} \parallel \mathit{Light}) \setminus \mathit{press}$
- Now $\overline{\mathit{press}}$-transition only enabled after 0.3 time units, which is also a possible delay for Light: $\mathit{Light} \xrightarrow{0.3} \mathit{press}.\mathit{Bright} + \varepsilon(1.2).\tau.\mathit{press}.\mathit{Off}$
- Therefore expected that whole system can delay:
  $((\varepsilon(0.3).\overline{\mathit{press}}.\mathit{FastUser}) \parallel \mathit{Light}) \setminus \mathit{press}$
  $\xrightarrow{0.3} ((\overline{\mathit{press}}.\mathit{FastUser}) \parallel (\mathit{press}.\mathit{Bright} + \varepsilon(1.2).\tau.\mathit{press}.\mathit{Off})) \setminus \mathit{press}$
- Now another synchronisation should be possible:
  $((\overline{\mathit{press}}.\mathit{FastUser}) \parallel (\mathit{press}.\mathit{Bright} + \varepsilon(1.2).\tau.\mathit{press}.\mathit{Off})) \setminus \mathit{press} \qquad (*)$
  $\xrightarrow{\tau} (\mathit{FastUser} \parallel \mathit{Bright}) \setminus \mathit{press}$

The Light Switch Example Revisited

Example 22.11

$$\begin{aligned} Off &= press.Light \\ Bright &= press.Off \\ Light &= press.Bright + \varepsilon(1.5).\tau.press.Off \\ FastUser &= \overline{press}.\varepsilon(0.3).\overline{press}.FastUser \end{aligned}$$

- Expect immediate synchronisation between $FastUser$ and Off:
  $(FastUser \parallel Off) \setminus press \xrightarrow{\tau} (\varepsilon(0.3).\overline{press}.FastUser \parallel Light) \setminus press$
- Now $\overline{press}$-transition only enabled after 0.3 time units, which is also a possible delay for $Light$: $Light \xrightarrow{0.3} press.Bright + \varepsilon(1.2).\tau.press.Off$
- Therefore expected that whole system can delay:
  $((\varepsilon(0.3).\overline{press}.FastUser) \parallel Light) \setminus press$
  $\xrightarrow{0.3} ((\overline{press}.FastUser) \parallel (press.Bright + \varepsilon(1.2).\tau.press.Off)) \setminus press$
- Now another synchronisation should be possible:
  $((\overline{press}.FastUser) \parallel (press.Bright + \varepsilon(1.2).\tau.press.Off)) \setminus press \qquad (*)$
  $\xrightarrow{\tau} (FastUser \parallel Bright) \setminus press$
- But: both parallel components of $(*)$ can delay for 1.2 time units, giving rise to
  $(*) \xrightarrow{1.2}\xrightarrow{\tau}\xrightarrow{\tau} (FastUser \parallel Off) \setminus press$

The Light Switch Example Revisited

Example 22.11

$$\begin{aligned} \mathit{Off} &= \mathit{press}.\mathit{Light} \\ \mathit{Bright} &= \mathit{press}.\mathit{Off} \\ \mathit{Light} &= \mathit{press}.\mathit{Bright} + \varepsilon(1.5).\tau.\mathit{press}.\mathit{Off} \\ \mathit{FastUser} &= \overline{\mathit{press}}.\varepsilon(0.3).\overline{\mathit{press}}.\mathit{FastUser} \end{aligned}$$

- Expect immediate synchronisation between $\mathit{FastUser}$ and Off:
  $(\mathit{FastUser} \parallel \mathit{Off}) \setminus \mathit{press} \xrightarrow{\tau} (\varepsilon(0.3).\overline{\mathit{press}}.\mathit{FastUser} \parallel \mathit{Light}) \setminus \mathit{press}$
- Now $\overline{\mathit{press}}$-transition only enabled after 0.3 time units, which is also a possible delay for Light: $\mathit{Light} \xrightarrow{0.3} \mathit{press}.\mathit{Bright} + \varepsilon(1.2).\tau.\mathit{press}.\mathit{Off}$
- Therefore expected that whole system can delay:
  $((\varepsilon(0.3).\overline{\mathit{press}}.\mathit{FastUser}) \parallel \mathit{Light}) \setminus \mathit{press}$
  $\xrightarrow{0.3} ((\overline{\mathit{press}}.\mathit{FastUser}) \parallel (\mathit{press}.\mathit{Bright} + \varepsilon(1.2).\tau.\mathit{press}.\mathit{Off})) \setminus \mathit{press}$
- Now another synchronisation should be possible:
  $((\overline{\mathit{press}}.\mathit{FastUser}) \parallel (\mathit{press}.\mathit{Bright} + \varepsilon(1.2).\tau.\mathit{press}.\mathit{Off})) \setminus \mathit{press} \qquad (*)$
  $\xrightarrow{\tau} (\mathit{FastUser} \parallel \mathit{Bright}) \setminus \mathit{press}$
- But: both parallel components of $(*)$ can delay for 1.2 time units, giving rise to
  $(*) \xrightarrow{1.2}\xrightarrow{\tau}\xrightarrow{\tau} (\mathit{FastUser} \parallel \mathit{Off}) \setminus \mathit{press}$
- How to enforce that intended synchronisation occurs immediately?

The Maximal-Progress Assumption

Maximal-progress assumption

If a process is ready to perform an action that is entirely under its control, then it will immediately do so without further delay.

The Maximal-Progress Assumption

Maximal-progress assumption

If a process is ready to perform an action that is entirely under its control, then it will immediately do so without further delay.

In the setting of timed CCS, the only action that is entirely under the control of a process is the τ-action. Therefore:

Maximal-progress assumption for Timed CCS

For each TCCS process $P \in Prc$, if $P \xrightarrow{\tau}$ then $P \not\xrightarrow{d}$ for any $d > 0$.

Operational Semantics with Maximal Progress

Definition 22.12 (Semantics of TCCS – timed parallel transitions)

Additionally for $P, P', Q, Q' \in Prc$ and $d \in \mathbb{R}_{\geq 0}$:

$$\text{(tPar)}\ \frac{P \xrightarrow{d} P' \quad Q \xrightarrow{d} Q' \quad NoSync(P, Q, d)}{P \parallel Q \xrightarrow{d} P' \parallel Q'}$$

where predicate $NoSync(P, Q, d)$ expresses that no synchronisation between P and Q becomes enabled by delaying less than d time units:

For each $0 \leq d' < d$ and $P', Q' \in Prc$,
if $P \xrightarrow{d'} P'$ and $Q \xrightarrow{d'} Q'$ then $P' \parallel Q' \stackrel{\tau}{\not\longrightarrow}$.

Operational Semantics with Maximal Progress

Definition 22.12 (Semantics of TCCS – timed parallel transitions)

Additionally for $P, P', Q, Q' \in Prc$ and $d \in \mathbb{R}_{\geq 0}$:

$$\text{(tPar)}\ \frac{P \xrightarrow{d} P' \quad Q \xrightarrow{d} Q' \quad NoSync(P, Q, d)}{P \parallel Q \xrightarrow{d} P' \parallel Q'}$$

where predicate $NoSync(P, Q, d)$ expresses that no synchronisation between P and Q becomes enabled by delaying less than d time units:

For each $0 \leq d' < d$ and $P', Q' \in Prc$,
if $P \xrightarrow{d'} P'$ and $Q \xrightarrow{d'} Q'$ then $P' \parallel Q' \not\xrightarrow{\tau}$.

Example 22.13

1. $(\varepsilon(0.3).\overline{press}.FastUser \parallel Light) \setminus press \not\xrightarrow{d}$ for any $d > 0.3$

Operational Semantics with Maximal Progress

Definition 22.12 (Semantics of TCCS – timed parallel transitions)

Additionally for $P, P', Q, Q' \in Prc$ and $d \in \mathbb{R}_{\geq 0}$:

$$(\text{tPar})\frac{P \xrightarrow{d} P' \quad Q \xrightarrow{d} Q' \quad NoSync(P, Q, d)}{P \parallel Q \xrightarrow{d} P' \parallel Q'}$$

where predicate $NoSync(P, Q, d)$ expresses that no synchronisation between P and Q becomes enabled by delaying less than d time units:

For each $0 \leq d' < d$ and $P', Q' \in Prc$,
if $P \xrightarrow{d'} P'$ and $Q \xrightarrow{d'} Q'$ then $P' \parallel Q' \stackrel{\tau}{\not\longrightarrow}$.

Example 22.13

1. $(\varepsilon(0.3).\overline{press}.FastUser \parallel Light) \setminus press \stackrel{d}{\not\longrightarrow}$ for any $d > 0.3$
2. $((\overline{press}.FastUser) \parallel (press.Bright + \varepsilon(1.2).\tau.press.Off)) \setminus press \stackrel{d}{\not\longrightarrow}$ for any $d > 0$

Modelling a Slow User

Example 22.14 (cf. Example 22.11)

$$\begin{aligned} \mathit{Off} &= \mathit{press}.\mathit{Light} \\ \mathit{Bright} &= \mathit{press}.\mathit{Off} \\ \mathit{Light} &= \mathit{press}.\mathit{Bright} + \varepsilon(1.5).\tau.\mathit{press}.\mathit{Off} \\ \mathit{SlowUser} &= \overline{\mathit{press}}.\varepsilon(1.7).\overline{\mathit{press}}.\mathit{SlowUser} \end{aligned}$$

Modelling a Slow User

Example 22.14 (cf. Example 22.11)

$$\begin{aligned} \mathit{Off} &= \mathit{press}.\mathit{Light} \\ \mathit{Bright} &= \mathit{press}.\mathit{Off} \\ \mathit{Light} &= \mathit{press}.\mathit{Bright} + \varepsilon(1.5).\tau.\mathit{press}.\mathit{Off} \\ \mathit{SlowUser} &= \overline{\mathit{press}}.\varepsilon(1.7).\overline{\mathit{press}}.\mathit{SlowUser} \end{aligned}$$

- As before:
  $(\mathit{SlowUser} \parallel \mathit{Off}) \setminus \mathit{press} \xrightarrow{\tau} (\varepsilon(1.7).\overline{\mathit{press}}.\mathit{SlowUser} \parallel \mathit{Light}) \setminus \mathit{press}$

Modelling a Slow User

Example 22.14 (cf. Example 22.11)

$$\begin{aligned} \mathit{Off} &= \mathit{press}.\mathit{Light} \\ \mathit{Bright} &= \mathit{press}.\mathit{Off} \\ \mathit{Light} &= \mathit{press}.\mathit{Bright} + \varepsilon(1.5).\tau.\mathit{press}.\mathit{Off} \\ \mathit{SlowUser} &= \overline{\mathit{press}}.\varepsilon(1.7).\overline{\mathit{press}}.\mathit{SlowUser} \end{aligned}$$

- As before:
  $(\mathit{SlowUser} \parallel \mathit{Off}) \setminus \mathit{press} \xrightarrow{\tau} (\varepsilon(1.7).\overline{\mathit{press}}.\mathit{SlowUser} \parallel \mathit{Light}) \setminus \mathit{press}$
- Now $\overline{\mathit{press}}$-transition only enabled after 1.7 time units, but Light can only delay for at most 1.5 units:
  $((\varepsilon(1.7).\overline{\mathit{press}}.\mathit{SlowUser}) \parallel \mathit{Light}) \setminus \mathit{press} \xrightarrow{1.5}$
  $((\varepsilon(0.2).\overline{\mathit{press}}.\mathit{SlowUser}) \parallel (\mathit{press}.\mathit{Bright} + \varepsilon(0).\tau.\mathit{press}.\mathit{Off})) \setminus \mathit{press} \quad (*)$

Modelling a Slow User

Example 22.14 (cf. Example 22.11)

$$
\begin{aligned}
\mathit{Off} &= \mathit{press}.\mathit{Light}\\
\mathit{Bright} &= \mathit{press}.\mathit{Off}\\
\mathit{Light} &= \mathit{press}.\mathit{Bright} + \varepsilon(1.5).\tau.\mathit{press}.\mathit{Off}\\
\mathit{SlowUser} &= \overline{\mathit{press}}.\varepsilon(1.7).\overline{\mathit{press}}.\mathit{SlowUser}
\end{aligned}
$$

- As before:
  $(\mathit{SlowUser} \parallel \mathit{Off}) \setminus \mathit{press} \xrightarrow{\tau} (\varepsilon(1.7).\overline{\mathit{press}}.\mathit{SlowUser} \parallel \mathit{Light}) \setminus \mathit{press}$
- Now $\overline{\mathit{press}}$-transition only enabled after 1.7 time units, but Light can only delay for at most 1.5 units:
  $((\varepsilon(1.7).\overline{\mathit{press}}.\mathit{SlowUser}) \parallel \mathit{Light}) \setminus \mathit{press} \xrightarrow{1.5}$
  $((\varepsilon(0.2).\overline{\mathit{press}}.\mathit{SlowUser}) \parallel (\mathit{press}.\mathit{Bright}+\varepsilon(0).\tau.\mathit{press}.\mathit{Off}))\setminus\mathit{press}\ (*)$
- Here the right-hand process of $(*)$ can do a τ-action, disabling further delays and thus avoiding the Bright state:
  $(*) \xrightarrow{\tau} ((\varepsilon(0.2).\overline{\mathit{press}}.\mathit{SlowUser}) \parallel (\mathit{press}.\mathit{Off})) \setminus \mathit{press}$

Outline

Miscellaneous

- Written exams:
  - 1st Friday 21 February 11:30 AH 2
  - 2nd Tuesday 25 March 10:00 AH 1

Miscellaneous

- Written exams:
  - 1st Friday 21 February 11:30 AH 2
  - 2nd Tuesday 25 March 10:00 AH 1
- Teaching in Summer 2014:
  - Seminar Concurrency Theory [Katoen/Noll]
  - Course Advanced Model Checking [Katoen/NN]
  - Course Modelling and Verification of Probabilistic Systems [Katoen/NN]
  - Course Compiler Construction [Noll/NN]