Concurrency, Time & Constraints

Frank D. Valencia

CNRS, LIX Ecole Polytechnique de Paris

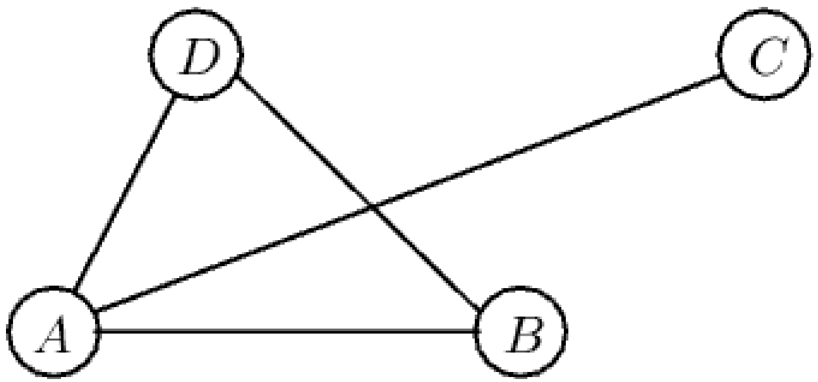
CLEI Cali, Oct 2005

CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

Concurrency

Concurrent Systems: **Agents** (or **processes**) that interact with each other.

- Systems as networks where **arcs represent agent interaction**.



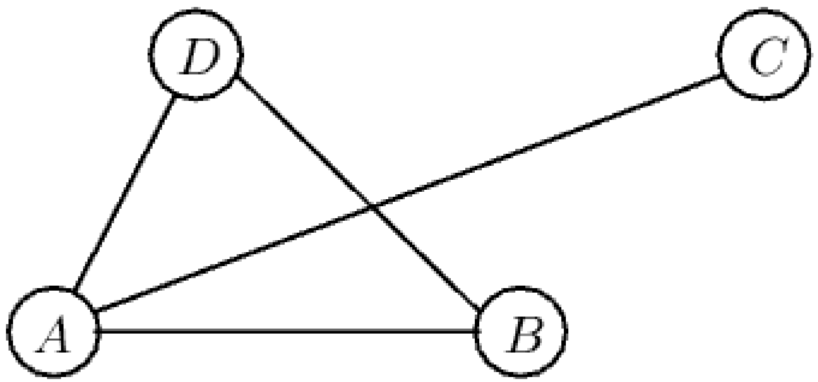
- Models for Concurrency: CCS, Pi-Calculus, CSP. Arcs denote **Links.**

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Concurrency, Constraints

In CCP [Saraswat, '89]: **Agents** interact via **constraints** over shared variables.

- Systems as networks where **arcs represent agent interaction**.



- *Arcs as constraints* on the (shared-variables of) agents.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Concurrency, Constraints, and Time

As other models, CCP has extended for new and wider phenomena

E.g:

- ***Mobility*** [Gilbert and Palamidessi '00, Réty '98, Rueda & Valencia '97].

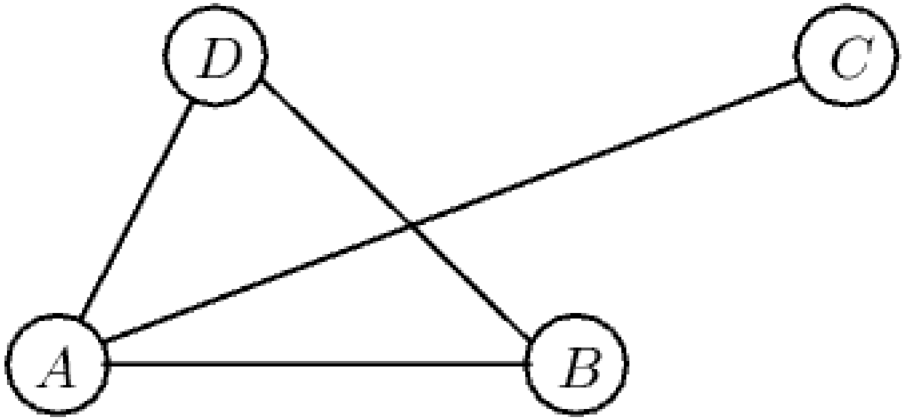
- ***Stochastic Behavior*** [Saraswat, Jagadeesan '98, Gupta-Panangaden-Jagadeesan '99]

- ***Timed Behavior***
  - (Basic) Timed CCP [Gupta-Jagadeesan-Saraswat '94]
  - Timed Default CCP [Gupta-Jagadeesan-Saraswat '95]
  - Hybrid CCP [Gupta-Jagadeesan-Saraswat '96]
  - Timed CCP: the tccp model [DeBoer-Gabbrielli-Meo '00]
  - Nondeterministic (Basic) Timed CCP [Nielsen-Palemidessi-Valencia '01]

Timed CCP: the tcc model

In the tcc model, as time passes, some constraints may ***disappear*** or be created.

E.g.: $\boxed{t = 1}$

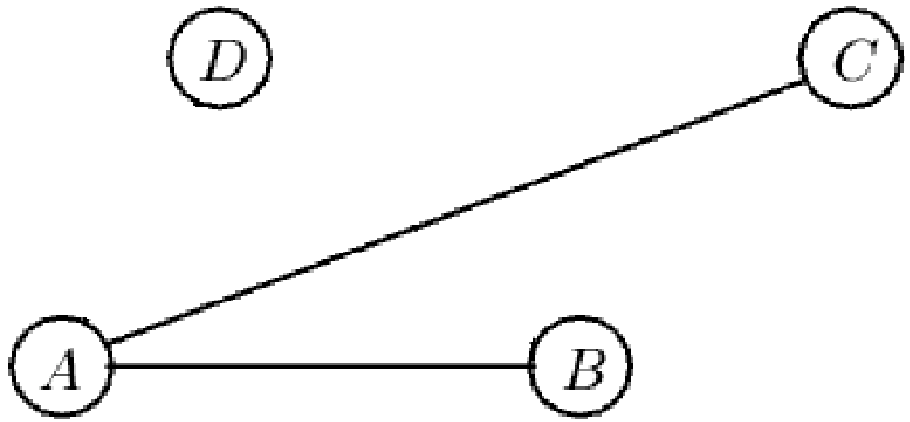


CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Timed CCP: the tcc model

In the tcc model, as time passes, some constraints may ***disappear*** or be created.

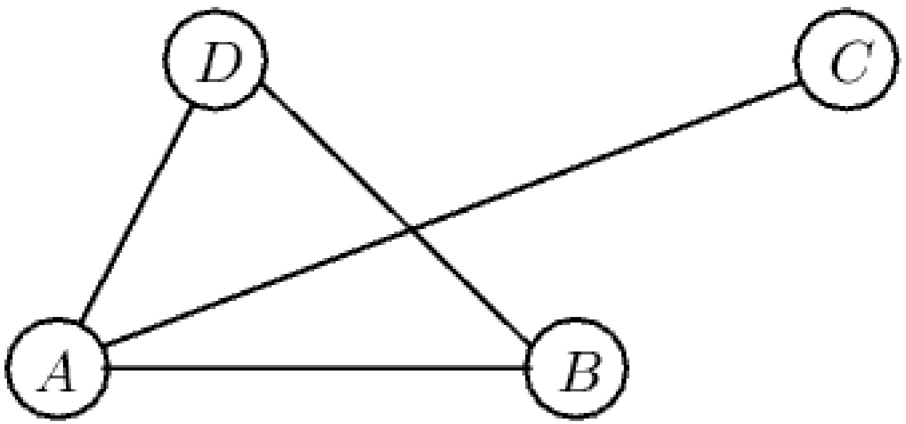
E.g.: $\boxed{t = 2}$



Timed CCP: the tcc model

In the tcc model, as time passes, some constraints may ***disappear*** or be created.

E.g.: $\boxed{t = 3}$

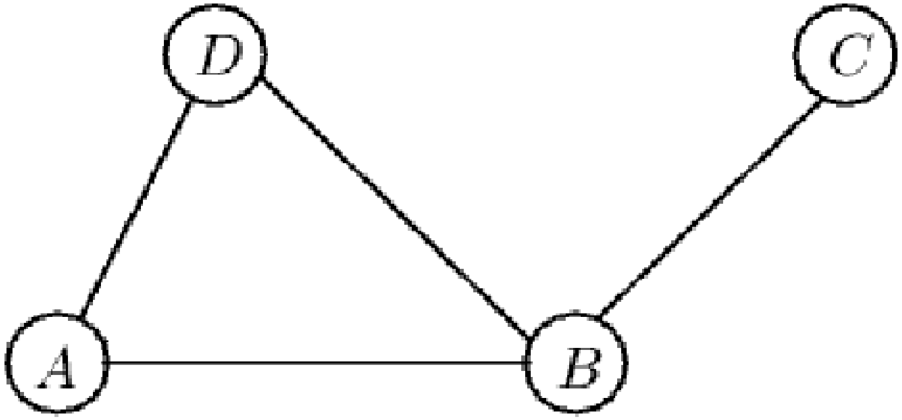


CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Timed CCP: the tcc model

In the tcc model, as time passes, some constraints may ***disappear*** or be created.

E.g.: $\boxed{t = 4}$



CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

The Goal of this Talk

"...One of the outstanding challenges in concurrency is to find the right marriage between ***logic*** and ***behavioural*** *approaches". R. Milner.*

About Timed CCP (I shall argue that):

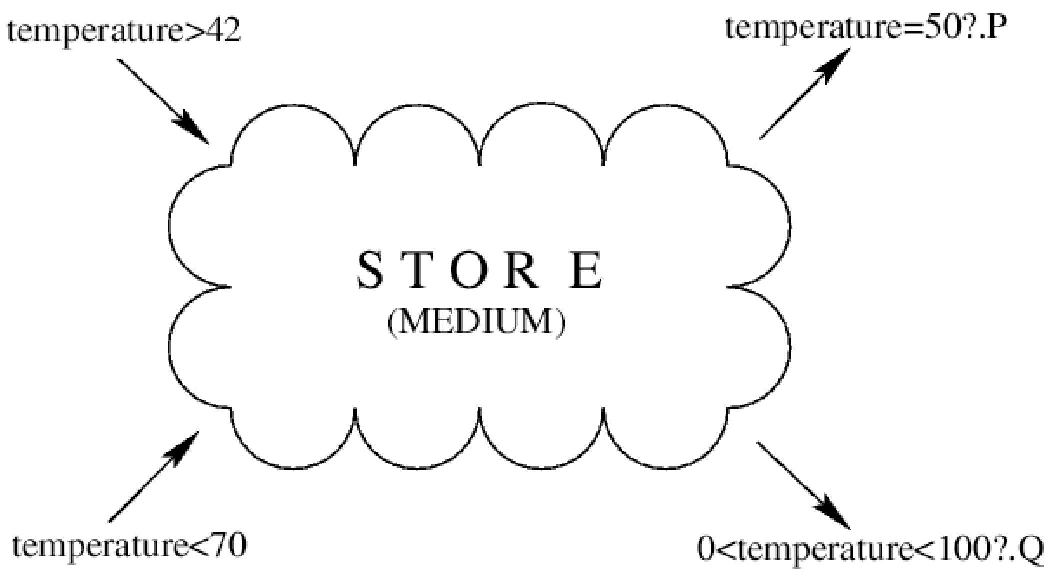
- It is simple.
- It expresses interesting real-world temporal situations.
- It is rigorously formalized upon process algebra and logic.
- It offers reasoning techniques from denotational semantics and process logic.

Agenda

- Basic Timed CCP intuitions.
- Semantics.
- A Logic and Proof System.
- Applications.
- Behavior.
- Hierarchy of temporal CCP languages
- Future Work

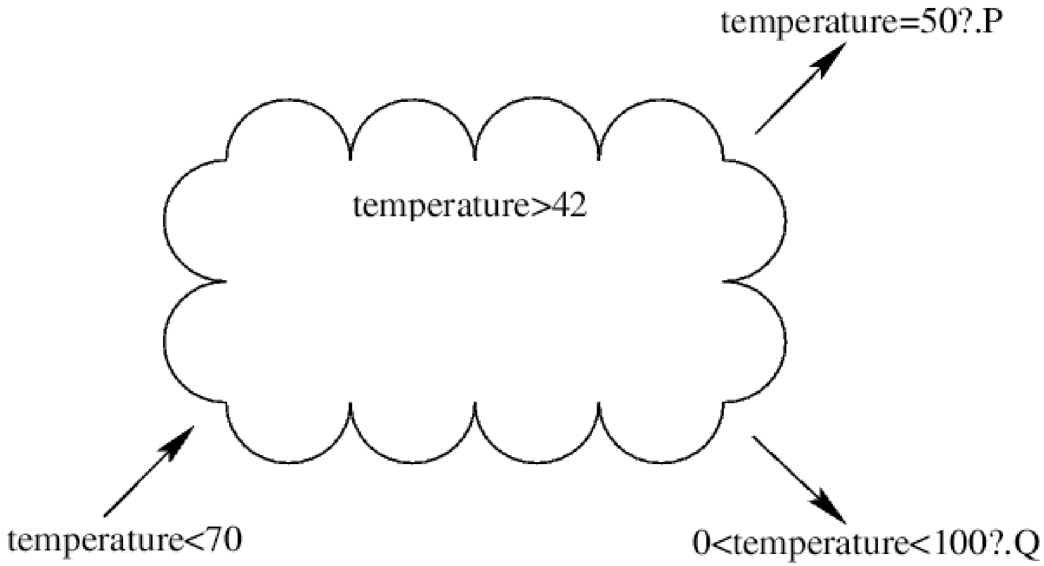
CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

CCP Intuitions: A Typical CCP Scenario

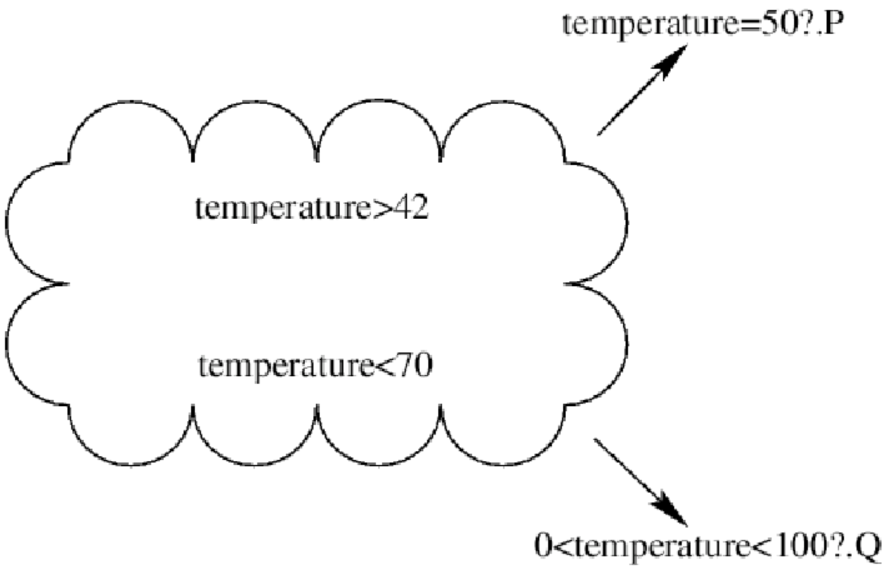


CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

CCP Intuitions: A Typical CCP Scenario

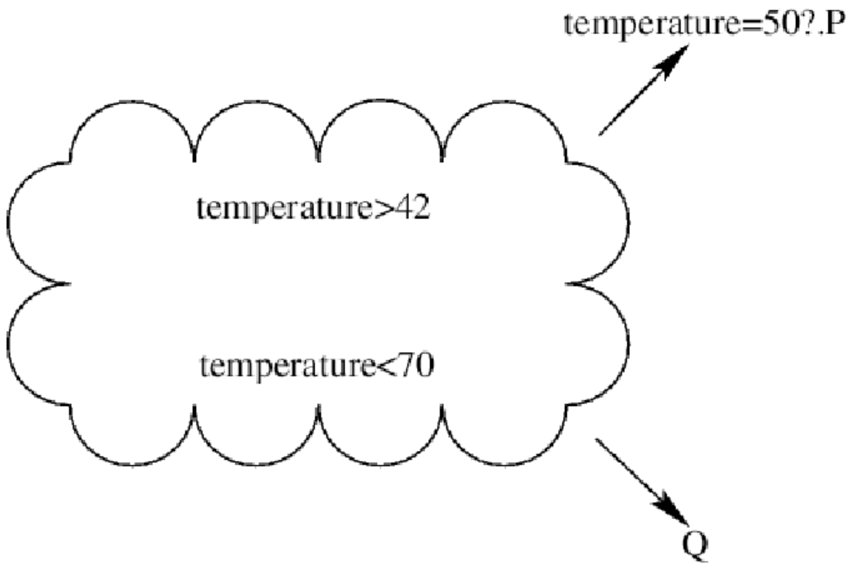


CCP Intuitions: A Typical CCP Scenario



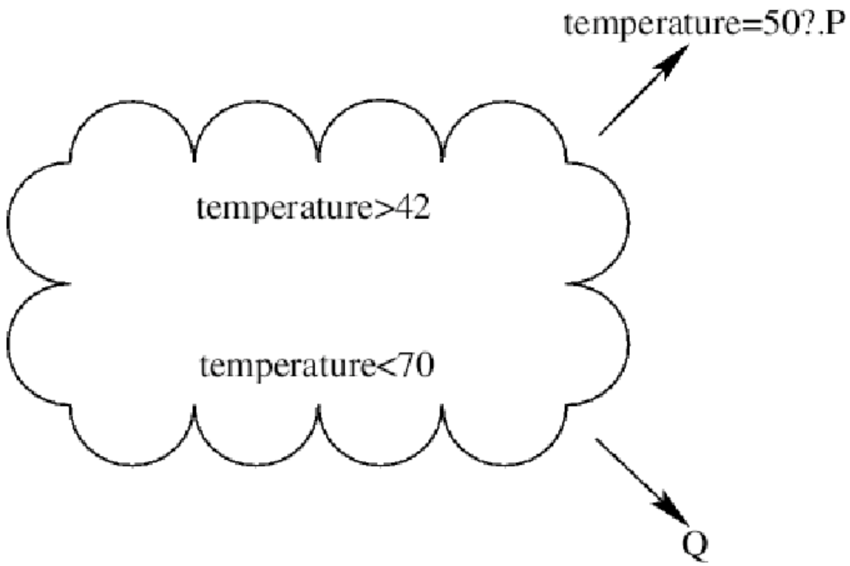
CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

CCP Intuitions: A Typical CCP Scenario



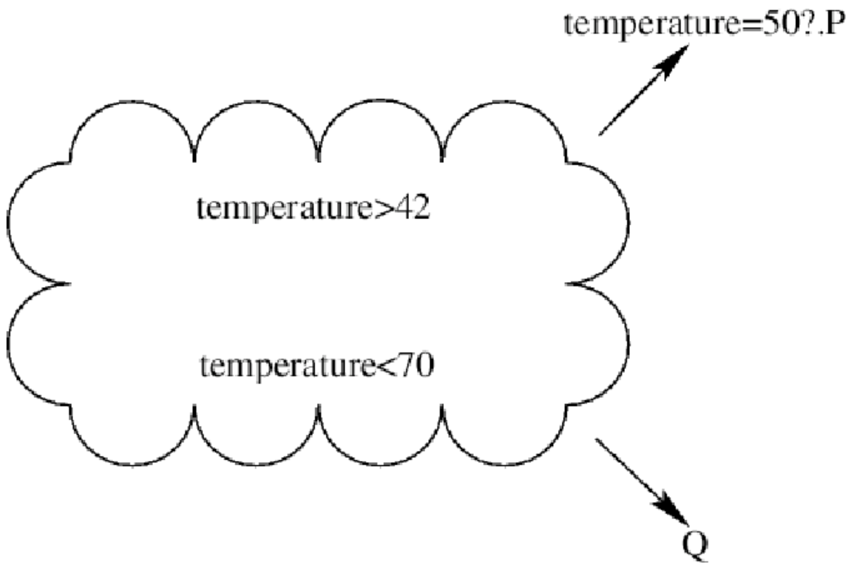
CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

CCP Intuitions: A Typical CCP Scenario



- Partial Information (e.g. `temperature` is some *unknown* value > 20).

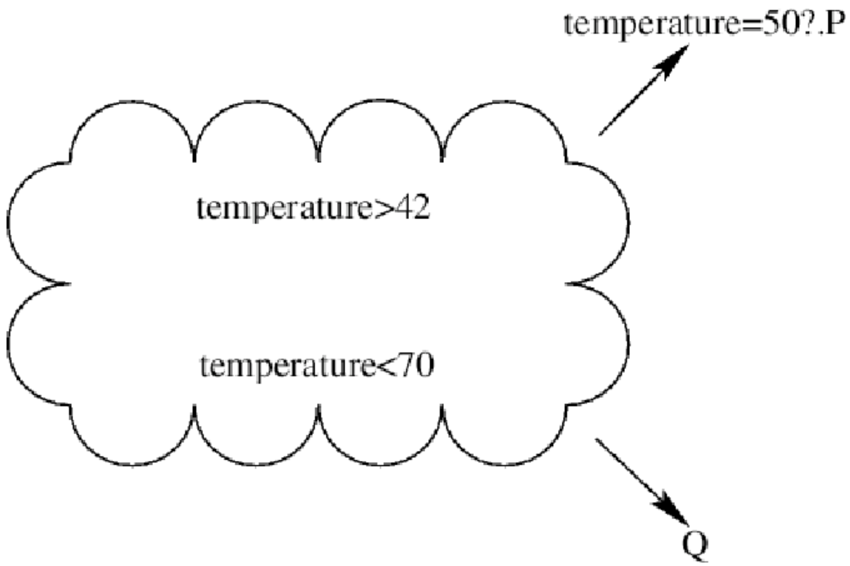
CCP Intuitions: A Typical CCP Scenario



- **Partial Information** (e.g. `temperature` is some *unknown* value > 20).
- **Concurrent Execution** of Processes.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

CCP Intuitions: A Typical CCP Scenario



- **Partial Information** (e.g. `temperature` is some *unknown* value > 20).
- **Concurrent Execution** of Processes.
- **Synchronization** via Blocking-Ask.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

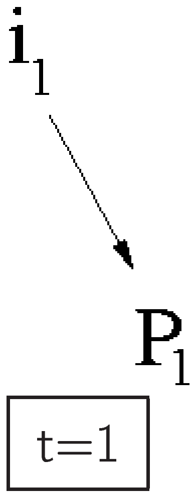
CCP Intuitions: Representing Partial Information

Definition. *A* constraint system *consists of a signature* Σ *and first-order theory* Δ *over* Σ.

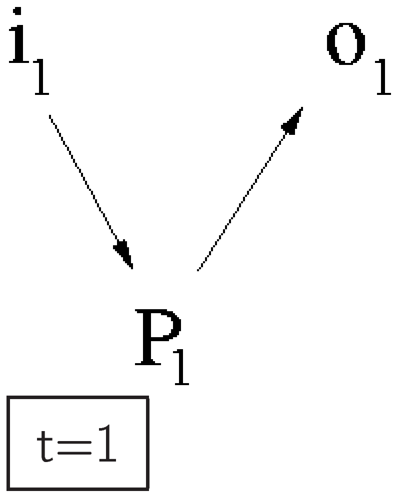
- **Constraints** $a, b, c, \ldots$: formulae over Σ.
- **Relation** $\vdash_\Delta$: decidable entailment relation between constraints.
- $\mathcal{C}$: set of constraints under consideration.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

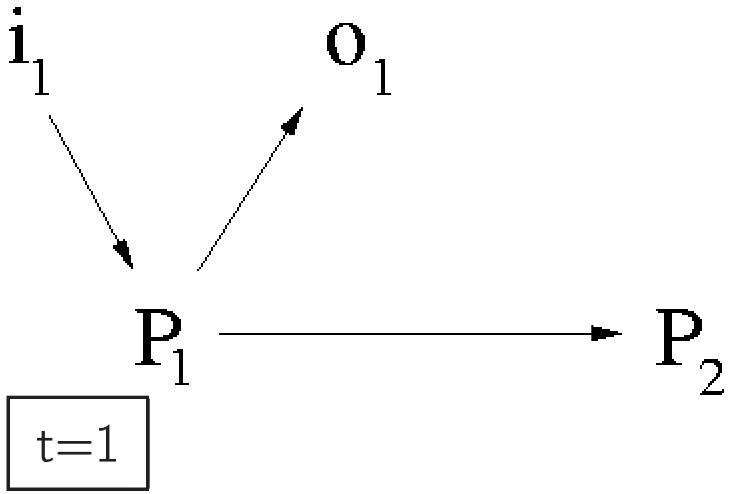
Reactive Systems



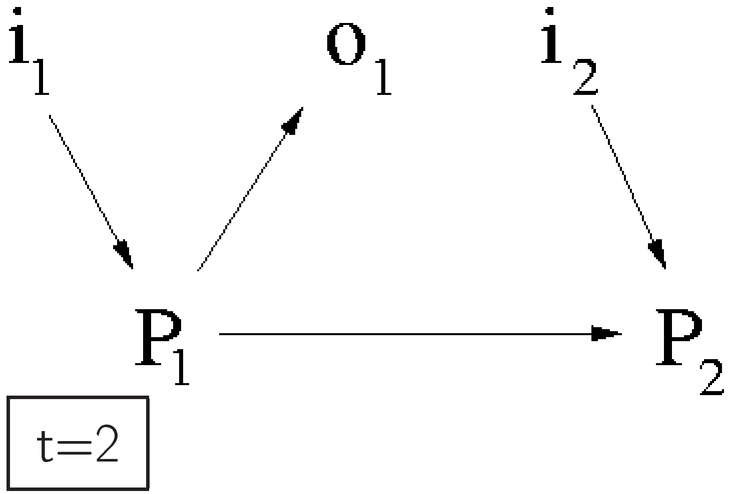
Reactive Systems



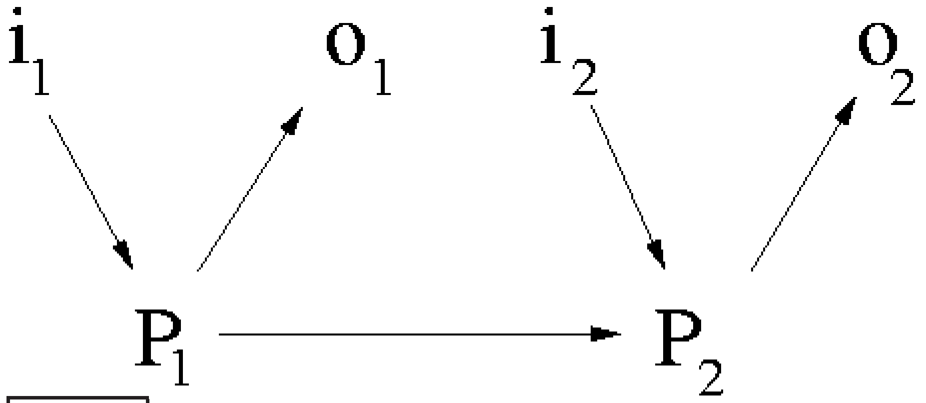
Reactive Systems



Reactive Systems

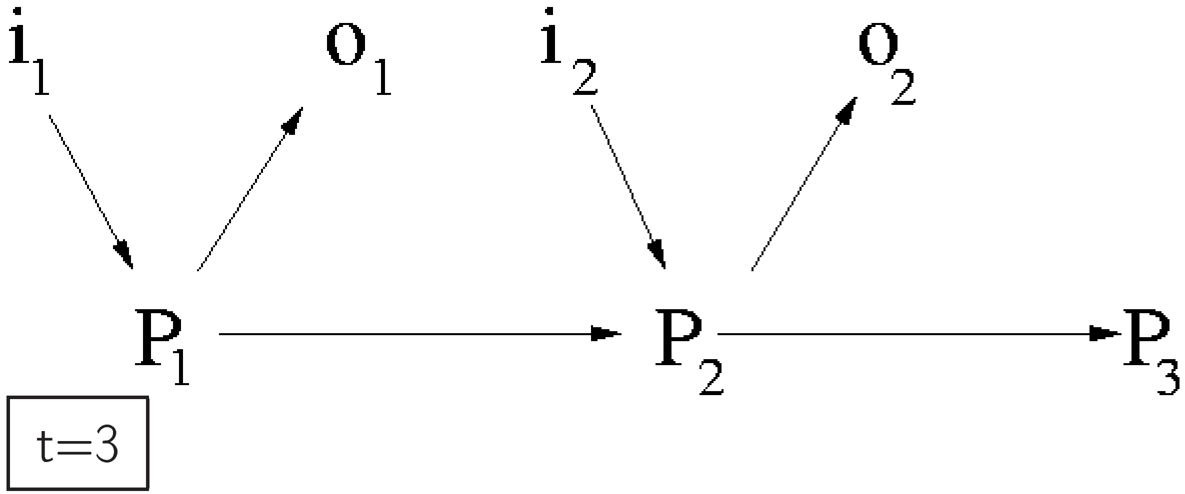


Reactive Systems



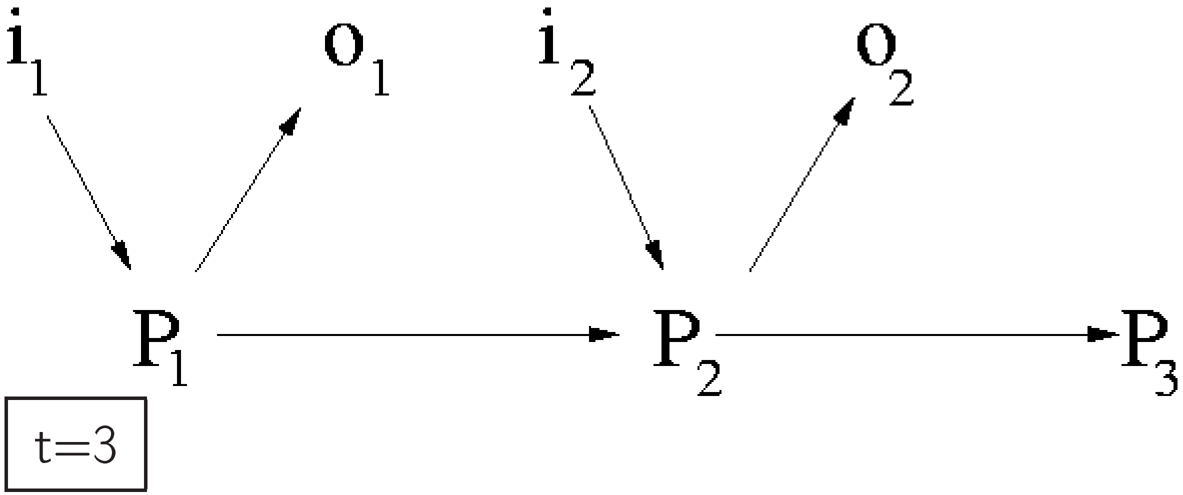
t=2

Reactive Systems



CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Reactive Systems



- **Stimulus** i_i : input information (as a constraint) for P_i .

- **Response** o_i: output information (as a constraint) of P_i .

- **Stimulus-Response** duration: **time interval** (or **time unit**).

Examples: PLC's, RCX Robots, Micro-Controllers, Synchronous Languages.

Basic tcc Processes

Processes	Description	Action within the time interval
$\mathbf{tell}(c)$	telling information	add c to the store

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Basic tcc Processes

Processes	Description	Action within the time interval
• $\mathbf{tell}(c)$	telling information	add c to the store
• $\mathbf{when}\, c\, \mathbf{do}\, P$	asking information	when c in the store execute P

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Basic tcc Processes

Processes	Description	Action within the time interval
• **tell**(c)	telling information	add c to the store
• **when** c **do** P	asking information	when c in the store execute P
• **local** x **in** P	hiding	execute P with local x

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Basic tcc Processes

Processes	Description	Action within the time interval
• **tell**(c)	telling information	add c to the store
• **when** c **do** P	asking information	when c in the store execute P
• **local** x **in** P	hiding	execute P with local x
• **next** P	unit-delay	delay P one time unit.

CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

Basic tcc Processes

Processes	Description	Action within the time interval
• **tell**(c)	telling information	add c to the store
• **when** c **do** P	asking information	when c in the store execute P
• **local** x **in** P	hiding	execute P with local x
• **next** P	unit-delay	delay P one time unit.
• **unless** c **next** P	time-out	unless c now in the store do P next

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Basic tcc Processes

Processes	Description	Action within the time interval
• $\mathbf{tell}(c)$	telling information	add c to the store
• $\mathbf{when}\, c\, \mathbf{do}\, P$	asking information	when c in the store execute P
• $\mathbf{local}\, x\, \mathbf{in}\, P$	hiding	execute P with local x
• $\mathbf{next}\, P$	unit-delay	delay P one time unit.
• $\mathbf{unless}\ c\, \mathbf{next}\, P$	time-out	unless c now in the store do P next
• $P \parallel Q$	parallelism	execute P and Q

Basic tcc Processes

Processes	Description	Action within the time interval
• $\sum_{i \in I} \textbf{when}\, c_i\, \textbf{do}\, P_i$	guarded choice	choose P_i s.t., c_i in the store

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Basic tcc Processes

Processes	Description	Action within the time interval
• $\sum_{i \in I} \textbf{when}\, c_i\, \textbf{do}\, P_i$	guarded choice	choose P_i s.t., c_i in the store
• $\star P$	unbounded delay	delay P undefinitely (not forever)

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Basic tcc Processes

Processes	Description	Action within the time interval
• $\sum_{i\in I}$ **when** c_i **do** P_i	guarded choice	choose P_i s.t., c_i in the store
• $\star P$	unbounded delay	delay P undefinitely (not forever)
• $!P$	replication	execute P each time unit

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Some Derived Constructs

- **Abortion**
  $\mathbf{abort} \stackrel{\text{def}}{=} !\,(\mathbf{tell}(\texttt{false}))$.

- **Asynchronous Parallel**
  $P \,|\, Q \stackrel{\text{def}}{=} (\star P \parallel Q) + (P \parallel \star Q)$

- **Bounded Replication**
  $!_{[t,t']} P \stackrel{\text{def}}{=} \prod_{t \le i \le t'} \mathbf{next}^i P$

- **Bounded Delay**
  $\star_{[t,t']} P \stackrel{\text{def}}{=} \sum_{t \le i \le t'} \mathbf{next}^i P$

CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

Power Saver Example

- A power saver :

$$!(\mathbf{unless}\,(\text{lights} = \text{off})\,\mathbf{next}\ \star\,\mathbf{tell}(\text{lights} = \text{off}))$$

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Power Saver Example

- A refined power saver :

$$!(\textbf{unless}\,(\,\mathrm{lights} = \mathrm{off})\,\textbf{next}\ \star_{[0,60]}\,\textbf{tell}(\mathrm{lights} = \mathrm{off}))$$

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Power Saver Example

- A more refined one; deterministic power saver:

$$!(\mathbf{unless}\ (\mathrm{lights} = \mathrm{off})\ \mathbf{next}\ \mathbf{tell}(\mathrm{lights} = \mathrm{off}))$$

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Operational Semantics

Internal Transitions:

$$RT\ \frac{}{\langle \mathbf{tell}(c), a\rangle \longrightarrow \langle \mathbf{skip}, a \wedge c\rangle} \qquad RG\ \frac{a \vdash c_j}{\langle \sum_{i\in I} \mathbf{when}\ c_i\ \mathbf{do}\ P_i, a\rangle \longrightarrow \langle P_j, a\rangle}$$

$$RB\ \frac{}{\langle !\,P, a\rangle \longrightarrow \langle P \parallel \mathbf{next}\,!\,P, a\rangle} \qquad RS\ \frac{}{\langle \star\,P, a\rangle \longrightarrow \langle \mathbf{next}^n P, a\rangle}\ (n\geq 0)$$

Observable Transition

$$RO\ \frac{\langle P, a\rangle \longrightarrow^* \langle Q, a'\rangle \not\longrightarrow}{P \xRightarrow{(a,a')} \mathbf{F}(Q)} \quad = \begin{cases} Q' & \text{if } Q = \mathbf{next}\ Q' \\ Q' & \text{if } Q = \mathbf{unless}\ (c)\ \mathbf{next}\ Q' \\ \mathbf{F}(Q_1) \parallel \mathbf{F}(Q_2) & \text{if } Q = Q_1 \parallel Q_2 \\ \mathbf{local}\ x\ \mathbf{in}\ \mathbf{F}(Q') & \text{if } Q = \mathbf{local}\ x\ \mathbf{in}\ Q' \\ \mathbf{skip} & \text{otherwise} \end{cases}$$

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Operational Semantics

- **Internal Transitions:**

$$RT\ \frac{}{\langle \mathbf{tell}(c), a\rangle \longrightarrow \langle \mathbf{skip}, a \wedge c\rangle} \qquad RG\ \frac{a \vdash c_j}{\langle \sum_{i\in I} \mathbf{when}\ c_i\ \mathbf{do}\ P_i, a\rangle \longrightarrow \langle P_j, a\rangle}$$

$$RB\ \frac{}{\langle !\, P, a\rangle \longrightarrow \langle P \parallel \mathbf{next}\, !\, P, a\rangle} \qquad RS\ \frac{}{\langle \star\, P, a\rangle \longrightarrow \langle \mathbf{next}^n P, a\rangle}\ (n \geq 0)$$

- **Observable Transition**

$$RO\ \frac{\langle P, a\rangle \longrightarrow^* \langle Q, a'\rangle \not\longrightarrow}{P \overset{(a,a')}{\Longrightarrow} \mathbf{F}(Q)} \qquad = \begin{cases} Q' & \text{if } Q = \mathbf{next}\ Q' \\ Q' & \text{if } Q = \mathbf{unless}\ (c)\ \mathbf{next}\ Q' \\ \mathbf{F}(Q_1) \parallel \mathbf{F}(Q_2) & \text{if } Q = Q_1 \parallel Q_2 \\ \mathbf{local}\ x\ \mathbf{in}\ \mathbf{F}(Q') & \text{if } Q = \mathbf{local}\ x\ \mathbf{in}\ Q' \\ \mathbf{skip} & \text{otherwise} \end{cases}$$

Observations to Make of Processes

- Stimulus-response interaction

$$P = P_1 \xrightarrow{(c_1,c'_1)} P_2 \xrightarrow{(c_2,c'_2)} P_3 \xrightarrow{(c_3,c'_3)} \ldots$$

denoted by $P \xrightarrow{(\alpha,\alpha')}{}^{\omega}$ with $\alpha = c_1.c_2\ldots$ and $\alpha' = c'_1.c'_2\ldots$

Observable Behavior

- Input-Output $io(P) = \{(\alpha, \alpha') \mid P \xrightarrow{(\alpha,\alpha')}{}^{\omega}\}$
- Output $o(P) = \{\alpha' \mid P \xrightarrow{(\mathtt{true}^{\omega},\alpha')}{}^{\omega}\}$
- Strongest Postcondition $sp(P) = \{\alpha' \mid P \xrightarrow{(\_,\alpha')}{}^{\omega}\}$

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Strongest-Postcondition Denotational Semantics

$$[\![\, \mathbf{tell}(a) \,]\!] = \{c \cdot \alpha \in C^\omega \;:\; c \vdash a, \}$$

$$[\![\, P \parallel Q \,]\!] = [\![\, P \,]\!] \cap [\![\, Q \,]\!]$$

$$[\![\, !P \,]\!] = \{\alpha \;:\; \text{for all } \beta \in C^*, \alpha' \in C^\omega : \alpha = \beta.\alpha' \text{ implies } \alpha' \in [\![\, P \,]\!]\}$$

$$[\![\, \star P \,]\!] = \{\beta.\alpha \;:\; \beta \in C^*, \alpha \in [\![\, P \,]\!]\}$$

$$[\![\, \textstyle\sum_{i \in I} \mathbf{when}\,(a_i)\,\mathbf{do}\, P_i \,]\!] = \bigcup_{i \in I}\{c \cdot \alpha \;:\; c \vdash a_i \text{ and } c \cdot \alpha \in [\![\, P_i \,]\!])\cup \\ (\bigcap_{i \in I}\{c \cdot \alpha \;:\; c \not\vdash a_i \,,\, \alpha \in C^\omega\})$$

Definition. P *is* locally-independent *iff its guards depend on no local variables.*

Theorem. $sp(P) \subseteq [\![\, P \,]\!]$ and, if P is a locally-independent, $sp(P) = [\![\, P \,]\!]$

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Related Work & Road Map

- IO Denotation for Timed CCP: [Gupta-Jagadeesan-Saraswat '94]

Related Work & Road Map

IO Denotation for Basic Timed CCP: [Gupta-Jagadeesan-Saraswat '94]

- IO Denotation for *Nondeterministic* Timed CCP [DeBoer-Gabrielli-Meo '01].

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Related Work & Road Map

IO Denotation for Basic Timed CCP: [Gupta-Jagedeesan-Saraswat '94]

IO Denotation *Nondeterministic* Timed CCP [DeBoer-Gabbrielli-Meo '01].

- SP Denotation for *Nondeterministic* Basic Timed CCP [Nielsen-Palimidessi-Valencia '02].

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Related Work & Road Map

IO Denotation for Basic Timed CCP: [Gupta-Jagadeesan-Saraswat'94]
IO Denotation *Nondeterministic* Timed CCP [DeBoer-Gabbrielli-Meo'01].
SP Denotation for *Nondeterministic* Basic Timed CCP [Nielsen-Palamidessi-Valencia'02].

RoadMap:

Operational and Denotational Models for Timed CCP

- Coming Next: Logic & Specification.

An LTL Process Logic

Syntax. $A := c \,|\, A \wedge A \,|\, \neg A \,|\, \exists_x A \,|\, \circ A \,|\, \Diamond A \,|\, \Box A$

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

An LTL Process Logic

Syntax. $A := c \,|\, A \wedge A \,|\, \neg A \,|\, \exists_x A \,|\, \circ A \,|\, \Diamond A \,|\, \Box A$

- c means "c holds in the current time unit"

An LTL Process Logic

Syntax. $A := c \,|\, A \wedge A \,|\, \neg A \,|\, \exists_x A \,|\, \circ A \,|\, \diamond A \,|\, \square A$

- c means "c holds in the current time unit"
- $\square A$ means "A holds always"

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

An LTL Process Logic

Syntax. $A := c \,|\, A \wedge A \,|\, \neg A \,|\, \exists_x A \,|\, \circ A \,|\, \Diamond A \,|\, \Box A$

- c means "c holds in the current time unit"
- $\Box A$ means "A holds always"
- $\Diamond A$ means "A eventually holds"

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

An LTL Process Logic

Syntax. $A := c \,|\, A \wedge A \,|\, \neg A \,|\, \exists_x A \,|\, \circ A \,|\, \diamond A \,|\, \square A$

- c means "c holds in the current time unit"
- $\square A$ means "A holds always"
- $\diamond A$ means "A eventually holds"
- $\circ A$ means "A holds in the next time unit"

.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

An LTL Process Logic

Syntax. $A := c \mid A \wedge A \mid \neg A \mid \exists_x A \mid \circ A \mid \diamond A \mid \Box A$

- c means "c holds in the current time unit".
- $\Box A$ means "A holds always".
- $\diamond A$ means "A eventually holds"
- $\circ A$ means "A holds in the next time unit"

Semantics. Say $\alpha = c_1.c_2.\ldots \models A$ iff $\langle \alpha, 1 \rangle \models A$ where

$\langle \alpha, i \rangle \models c$	iff	$c_i \vdash c$
$\langle \alpha, i \rangle \models \neg A$	iff	$\langle \alpha, i \rangle \not\models A$
$\langle \alpha, i \rangle \models A_1 \wedge A_2$	iff	$\langle \alpha, i \rangle \models A_1$ and $\langle \alpha, i \rangle \models A_2$
$\langle \alpha, i \rangle \models \circ A$	iff	$\langle \alpha, i+1 \rangle \models A$
$\langle \alpha, i \rangle \models \Box A$	iff	for all $j \geq i$ $\langle \alpha, j \rangle \models A$
$\langle \alpha, i \rangle \models \diamond A$	iff	there exists $j \geq i$ s.t. $\langle \alpha, j \rangle \models A$
$\langle \alpha, i \rangle \models \exists_x A$	iff	there is α' $x$variant of α s.t. $\langle \alpha', i \rangle \models A$.

An LTL Process Logic

Syntax. $A := c \mid A \wedge A \mid \neg A \mid \exists_x A \mid \circ A \mid \Diamond A \mid \Box A$

- c means "c holds in the current time unit".
- $\Box A$ means "A holds always".
- $\Diamond A$ means "A eventually holds"
- $\circ A$ means "A holds in the next time unit"

.

For example

- If $\alpha = (x > 1).(x > 2).(x > 3)\ldots.$ then $\alpha \models \Diamond x > 42$

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

An LTL Process Logic

Syntax. $A := c \,|\, A \wedge A \,|\, \neg A \,|\, \exists_x A \,|\, \circ A \,|\, \diamond A \,|\, \Box A$

- c means "c holds in the current time unit".
- $\Box A$ means "A holds always".
- $\diamond A$ means "A eventually holds"
- $\circ A$ means "A holds in the next time unit"

.

For example

- If $\alpha = (x > 1).(x > 2).(x > 3)\ldots.$ then $\alpha \models \diamond x > 42$

- $\Box(A \vee B) \Leftrightarrow \Box A \vee \Box B$??

An LTL Process Logic

Syntax. $A := c \,|\, A \wedge A \,|\, \neg A \,|\, \exists_x A \,|\, \circ A \,|\, \Diamond A \,|\, \Box A$

- c means "c holds in the current time unit".
- $\Box A$ means "A holds always".
- $\Diamond A$ means "A eventually holds"
- $\circ A$ means "A holds in the next time unit"

.

For example

- If $\alpha = (x > 1).(x > 2).(x > 3)\ldots.$ then $\alpha \models \Diamond x > 42$
- $\Box(A \vee B) \Leftrightarrow \Box A \vee \Box B$??
- $\Box\Diamond A \Leftrightarrow \Diamond\Box A$??

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Specification and Satisfaction

Specifications can be expressed as LTL formulae.

Specification and Satisfaction

Specifications can be expressed as LTL formulae. We then say:

P meets A, written $P \models A$, iff all sequences P outputs satisfy A

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Specification and Satisfaction

Specifications can be expressed as LTL formulae. We then say:

P meets A, written $P \models A$, iff all sequences P outputs satisfy A

E.g.,

- $!(\mathbf{unless}\ (\text{LightsOff})\ \mathbf{next}\ \star\ \mathbf{tell}(\text{LightsOff})) \quad \models \quad \Diamond(\text{LightsOff})$

Specification and Satisfaction

Specifications can be expressed as LTL formulae. We then say:

P meets A, written $P \models A$, iff all sequences P outputs satisfy A

E.g.,

- $!(\textbf{unless}\ (\text{LightsOff})\ \textbf{next}\ \star\ \textbf{tell}(\text{LightsOff})) \quad \models \quad \Diamond(\text{LightsOff})$
- $!(\textbf{when}\ (\text{AlarmGoesOff})\ \textbf{do}\ \textbf{tell}(\text{CloseGate})) \quad \models \quad \Box\, \text{AlarmGoesOff} \ \dot{\Rightarrow}\ \Box\text{CloseGate}$

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Specification and Satisfaction

Specifications can be expressed as LTL formulae. We then say:

P meets A, written $P \models A$, iff all sequences P outputs satisfy A

E.g.,

- $!(\textbf{unless}\ (\text{LightsOff})\ \textbf{next}\ \star\ \textbf{tell}(\text{LightsOff})) \quad \models \quad \Diamond(\text{LightsOff})$

- $!(\textbf{when}\ (\text{AlarmGoesOff})\ \textbf{do}\ \textbf{tell}(\text{CloseGate})) \quad \models \quad \Box\,\text{AlarmGoesOff} \ \dot{\Rightarrow}\ \Box\text{CloseGate}$

- **But how can we prove $P \models A$?**

Proof System for $P \models A$

$$\mathbf{tell}(c) \vdash c \ \text{(tell)}$$

$$\frac{P \vdash A \quad Q \vdash B}{P \parallel Q \vdash A \wedge B}\ \text{(par)} \qquad \frac{P \vdash A}{\mathbf{local}\ x\ \mathbf{in}\ P \vdash \exists_x A}\ \text{(hide)}$$

$$\frac{P \vdash A}{\mathbf{next}\ P \vdash \bigcirc A}\ \text{(next)}$$

$$\frac{P \vdash A}{!P \vdash \Box A}\ \text{(rep)} \qquad \frac{P \vdash A}{\star P \vdash \Diamond A}\ \text{(star)}$$

$$\frac{\forall i \in I \ \ P_i \vdash A_i}{\sum_{i \in I} \mathbf{when}\ c_i\ \mathbf{do}\ P_i \vdash \bigvee_{i \in I}(c_i \wedge A_i) \vee \bigwedge_{i \in I} \neg c_i}\ \text{(sum)}$$

$$\frac{P \vdash A \quad A \Rightarrow B}{P \vdash B}\ \text{(rel)}$$

Theorem. *(Completeness) For every locally-independent process P,*

$$P \models A \quad \textit{iff} \quad P \vdash A$$

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Verification $P \models A$

- Can we prove $P \models A$ *automatically* ?

Verification $P \models A$

- Can we prove $P \models A$ *automatically* ?

YES, even for **infinite-state** processes and **first-order** LTL formulae!.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Verification $P \models A$

- **Can we prove** $P \models A$ ***automatically*** ?

YES, even for **infinite-state** processes and **first-order** LTL formulae!.

Theorem. *Given a locally-independent P and a negation-free A, the problem of whether $P \models A$ is* **decidable**.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Verification $P \models A$

- **Can we prove $P \models A$ *automatically* ?**

YES, even for **infinite-state** processes and **first-order** LTL formulae!.

Theorem. *Given a locally-independent P and a negation-free A, the problem of whether $P \models A$ is* **decidable**.

...and the proof uses the ***denotational semantics*** *rather than the operational semantics !.*

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Theoretical Applications: Pnueli's First-Order LTL

Pnueli's First-Order LTL (FOLTL):

- Syntax like that of the Timed CCP Logic.

Theoretical Applications: Pnueli's First-Order LTL

Pnueli's First-Order LTL (FOLTL):

- Syntax like that of the Timed CCP Logic.
- Models are sequences of states

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Theoretical Applications: Pnueli's First-Order LTL

Pnueli's First-Order LTL (FOLTL):

- Syntax like that of the Timed CCP Logic.
- Models are sequences of states
- Variables can be **flexible** (i.e., can change as time passes) or **rigid**.

Theoretical Applications: Pnueli's First-Order LTL

Pnueli's First-Order LTL (FOLTL):

- Syntax like that of the Timed CCP Logic.
- Models are sequences of states
- Variables can be **flexible** (i.e., can change as time passes) or **rigid**.
- [Abadi '89] proved the full-language to be **undecidable.**

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Theoretical Applications: Pnueli's First-Order LTL

Pnueli's First-Order LTL (FOLTL):

- Syntax like that of the Timed CCP Logic.
- Models are sequences of states
- Variables can be **flexible** (i.e., can change as time passes) or **rigid**.
- [Abadi '89] proved the full-language to be **undecidable.**
- Several work identifying decidable fragments of FOLTL.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Theoretical Applications: Pnueli's First-Order LTL

Pnueli's First-Order LTL (FOLTL):

- Syntax like that of the Timed CCP Logic.
- Models are sequences of states
- Variables can be **flexible** (i.e., can change as time passes) or **rigid**.
- [Abadi '89] proved the full-language to be **undecidable.**
- Several work identifying decidable fragments of FOLTL.
- *Without rigid variables, FOLTL is* ***decidable***. Proof by using the theory of Timed CCP.

Programming Applications: Cells

- **Cell** $x : (v)$ models *cell* x *with contents* v.

Programming Applications: Cells

- **Cell** $x : (v)$ models *cell* x *with contents* v.

$$x : (z) \stackrel{\mathrm{def}}{=} \mathbf{tell}(x = z) \parallel \mathbf{unless}\ \mathrm{change}(x)\ \mathbf{next}\ x : (z)$$

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Programming Applications: Cells

- **Cell** $x : (v)$ models *cell* x *with contents* v.

$$x : (z) \stackrel{\text{def}}{=} \textbf{tell}(x = z) \parallel \textbf{unless}\ \mathrm{change}(x)\ \textbf{next}\ x : (z)$$

- **Exchange** $exch_f(x, y)$ models $y := x \ ; x := f(x)$.

Programming Applications: Cells

- **Cell** $x : (v)$ models *cell* x *with contents* v.

$$x : (z) \stackrel{\text{def}}{=} \textbf{tell}(x = z) \parallel \textbf{unless}\ \mathsf{change}(x)\ \textbf{next}\ x : (z)$$

- **Exchange** $exch_f(x, y)$ models $y := x$; $x := f(x)$.

$$exch_f(x, y) \stackrel{\text{def}}{=} \sum_v \textbf{when}\ (x = v)\ \textbf{do}\ (\textbf{tell}(\mathsf{change}(x)) \parallel \textbf{tell}(\mathsf{change}(y)) \parallel \textbf{next}(x : f(v) \parallel y : (v))\ )$$

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Programming Applications: Cells

- **Cell** $x : (v)$ models *cell* x *with contents* v.

$$x : (z) \stackrel{\text{def}}{=} \mathbf{tell}(x = z) \parallel \mathbf{unless}\ \mathsf{change}(x)\ \mathbf{next}\ x : (z)$$

- **Exchange** $exch_f(x, y)$ models $y := x\ ;\ x := f(x)$.

$$exch_f(x, y) \stackrel{\text{def}}{=} \sum_v \mathbf{when}\ (x = v)\ \mathbf{do}\ (\begin{array}{llll} & \mathbf{tell}(\mathsf{change}(x)) & \parallel & \mathbf{tell}(\mathsf{change}(y)) \\ \parallel & \mathbf{next}(x : f(v) & \parallel & y : (v))\ ) \end{array}$$

Example. $x : (3) \parallel y : (5) \parallel exch_7(x, y)$

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Programming Applications: Cells

- **Cell** $x : (v)$ models *cell* x *with contents* v.

$$x : (z) \stackrel{\text{def}}{=} \textbf{tell}(x = z) \parallel \textbf{unless}\ \mathsf{change}(x)\ \textbf{next}\ x : (z)$$

- **Exchange** $exch_f(x, y)$ models $y := x \,;\, x := f(x)$.

$$exch_f(x, y) \stackrel{\text{def}}{=} \sum_v \textbf{when}\ (x = v)\ \textbf{do}\ (\ \textbf{tell}(\mathsf{change}(x)) \parallel \textbf{tell}(\mathsf{change}(y)) \parallel \textbf{next}(x : f(v) \parallel y : (v))\ )$$

Example. $x : (3) \parallel y : (5) \parallel exch_7(x, y) \stackrel{\cdot}{\Longrightarrow} x : (7) \parallel y : (3)$.

Applications: Logic & Proof System at Work

Proposition. $\boxed{exch_f(x,y) \ \vdash \ (x = v) \Rightarrow \circ(x = f(v) \wedge y = v)}$.

Applications: Logic & Proof System at Work

Proposition. $$\boxed{exch_f(x,y) \ \vdash \ (x = v) \Rightarrow \bigcirc(x = f(v) \wedge y = v)}.$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : (g(w)) \vdash x = g(w)}\ Pr.1 \qquad \dfrac{}{y : (w) \vdash y = w}\ Pr.1}{x : (g(w)) \parallel y : (w) \vdash x = g(w) \mathrel{\dot\wedge} y = w}\ \mathit{L}\mathrm{PAR}}{\mathbf{next}\ (x : (g(w)) \parallel y : (w)) \vdash \bigcirc(x = g(w) \mathrel{\dot\wedge} y = w)}\ \mathit{L}\mathrm{NEXT}}{\forall w \in \mathcal{D} \ \ \mathbf{tell}(change(x)) \parallel \mathbf{tell}(change(y)) \parallel \mathbf{next}(x : f(w) \parallel \ y : (w)) \vdash \bigcirc(x = g(w) \mathrel{\dot\wedge} y = w)}\ Lem.(3)$$

$$\dfrac{\forall w \in \mathcal{D} \ \ \mathbf{tell}(change(x)) \parallel \mathbf{tell}(change(y)) \parallel \mathbf{next}(x : f(w) \parallel \ y : (w)) \vdash \bigcirc(x = g(w) \mathrel{\dot\wedge} y = w)}{exch_f(x,y) \vdash \dot{\bigvee_{w \in \mathcal{D}}} (x = w \mathrel{\dot\wedge} \bigcirc(x = g(w) \mathrel{\dot\wedge} y = w)) \mathrel{\dot\vee} \dot{\bigwedge_{w \in \mathcal{D}}} \dot\neg\, x = w}\ \mathit{L}\mathrm{SUM}$$

$$\dfrac{\dfrac{exch_f(x,y) \vdash \dot{\bigvee_{w \in \mathcal{D}}} (x = w \mathrel{\dot\wedge} \bigcirc(x = g(w) \mathrel{\dot\wedge} y = w)) \mathrel{\dot\vee} \dot{\bigwedge_{w \in \mathcal{D}}} \dot\neg\, x = w}{exch_f(x,y) \vdash \dot{\bigwedge_{w \in \mathcal{D}}} (x = w \mathrel{\dot\Rightarrow} \bigcirc(x = g(w) \mathrel{\dot\wedge} y = w))}\ LCONS}{exch_f(x,y) \vdash (x = v \mathrel{\dot\Rightarrow} \bigcirc(x = g(v) \mathrel{\dot\wedge} y = v))}\ LCONS$$

Programming Applications: LEGO Zigzagging

Specification. Go *forward* (`f`), *right* (`r`) or *left* (`l`) but DO NOT go:

- `f` if preceding action was `f`,
- `r` if second-to-last action was `r`, and
- `l` if second-to-last action was `l`.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Programming Applications: LEGO Zigzagging

Specification. Go *forward* (`f`), *right* (`r`) or *left* (`l`) but DO NOT go:

- `f` if preceding action was `f`,
- `r` if second-to-last action was `r`, and
- `l` if second-to-last action was `l`.

$$
\begin{array}{llll}
GoForward & \stackrel{\text{def}}{=} & & \mathtt{f}_{exch}(act_1, act_2) \parallel \mathbf{tell}(\mathtt{forward}) \\
GoRight & \stackrel{\text{def}}{=} & & \mathtt{r}_{exch}(act_1, act_2) \parallel \mathbf{tell}(\mathtt{right}) \\
GoLeft & \stackrel{\text{def}}{=} & & \mathtt{l}_{exch}(act_1, act_2) \parallel \mathbf{tell}(\mathtt{left}) \\
Zigzag & \stackrel{\text{def}}{=} & (& \mathbf{when}\,(act_1 \neq \mathtt{f})\,\mathbf{do}\ GoForward \\
 & & + & \mathbf{when}\,(act_2 \neq \mathtt{r})\,\mathbf{do}\ GoRight \\
 & & + & \mathbf{when}\,(act_2 \neq \mathtt{l})\,\mathbf{do}\ GoLeft\ \ ) \\
 & & \parallel & \mathbf{next}\ Zigzag \\
StartZigzag & \stackrel{\text{def}}{=} & & act_1\!:\!(0) \parallel act_2\!:\!(0) \parallel Zigzag
\end{array}
$$

Programming Applications: LEGO Zigzagging

Specification. Go *forward* (`f`), *right* (`r`) or *left* (`l`) but DO NOT go:

- `f` if preceding action was `f`,
- `r` if second-to-last action was `r`, and
- `l` if second-to-last action was `l`.

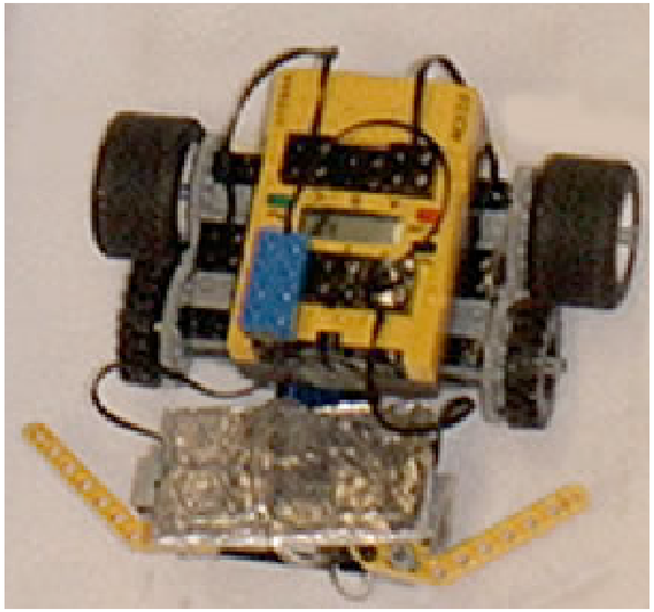
$$
\begin{array}{llll}
\mathit{GoForward} & \stackrel{\text{def}}{=} & & \mathtt{f}_{exch}(act_1, act_2) \parallel \mathbf{tell}(\mathtt{forward}) \\
\mathit{GoRight} & \stackrel{\text{def}}{=} & & \mathtt{r}_{exch}(act_1, act_2) \parallel \mathbf{tell}(\mathtt{right}) \\
\mathit{GoLeft} & \stackrel{\text{def}}{=} & & \mathtt{l}_{exch}(act_1, act_2) \parallel \mathbf{tell}(\mathtt{left}) \\
\mathit{Zigzag} & \stackrel{\text{def}}{=} & (& \mathbf{when}\,(act_1 \neq \mathtt{f})\,\mathbf{do}\ \mathit{GoForward} \\
 & & + & \mathbf{when}\,(act_2 \neq \mathtt{r})\,\mathbf{do}\ \mathit{GoRight} \\
 & & + & \mathbf{when}\,(act_2 \neq \mathtt{l})\,\mathbf{do}\ \mathit{GoLeft}\ ) \\
 & & \parallel & \mathbf{next}\ \mathit{Zigzag} \\
\mathit{StartZigzag} & \stackrel{\text{def}}{=} & & act_1{:}(0) \parallel act_2{:}(0) \parallel \mathit{Zigzag}
\end{array}
$$

Proposition. StartZigzag $\vdash \Box(\Diamond\mathtt{right} \wedge \Diamond\mathtt{left})$

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

A Timed CCP Programming Language for Robots

LMAN (Hurtado&Munoz 2003): A timed ccp reactive programming language for **LEGO RCX** Robots.



Music Applications: Controlled Improvisation.

Music composition and performance is a complex task of defining and controlling concurrent activity. E.g:

▷ There are $M_1, \ldots, M_m$ musicians (or *Voices* if you wish). Each M_i is given a three-notes pattern p_i of delays between each note in the block.

▷ Once her block is played, the musician waits for the others to finish their respective blocks before start playing a new one.

▷ The exact time a new block will be started is not specified, but should not be later than `pdur`; the sum of the durations of all patterns.

▷ Musicians keep playing notes until all of them play a note simultaneously.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Music Applications: Controlled Improvisation

$$M_i \stackrel{\text{def}}{=} \sum_{(j,k,l)\in perm(p_i)} \left(\; Play^i_{(j,k,l)} \parallel \mathbf{next}^{j+k+l}\left(\; \mathit{flag}_i := 1 \parallel \mathbf{whenever}\ (go = 1)\ \mathbf{do}\ \star_{[0,\mathrm{pdur}]} M_i \; \right) \right)$$

$$Play^i_{(j,k,l)} \stackrel{\text{def}}{=} !_{[0,j-1]}\mathbf{tell}(note_i = \texttt{sil}) \parallel \mathbf{next}^j\mathbf{tell}(c_i[note_i]) \parallel !_{[j+1,j+k-1]}\mathbf{tell}(note_i = \texttt{sil}) \parallel \mathbf{next}^{j+k}\mathbf{tell}(c_i[note_i]) \parallel !_{[j+k+1,j+k+l-1]}\mathbf{tell}(note_i = \texttt{sil}) \parallel \mathbf{next}^{j+k+l}\mathbf{tell}(c_i[note_i])$$

$$C \stackrel{\text{def}}{=} !(\mathbf{when} \bigwedge_{i\in[1,m]}(\mathit{flag}_i = 1) \wedge (stop = 0)\ \mathbf{do}\ (\mathbf{tell}(go = 1) \parallel \prod_{i\in[1,m]} \mathit{flag}_i := 0)) \parallel \mathbf{next\ whenever} \bigwedge_{i\in[1,m]}(note_i \neq \texttt{sil})\ \mathbf{do}\ stop := 1$$

CNRS CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Music Applications: Controlled Improvisation

$$Init \stackrel{\text{def}}{=} \prod_{i\in[1,m]}(\mathbf{tell}(c_i[note_i]) \parallel flag_i : 0) \parallel stop : 0$$

$$Sys \stackrel{\text{def}}{=} Init \parallel C \parallel \prod_{i\in[1,m]} M_i$$

▷ Notice that *regardless the musicians' choices* the system always terminates iff

$$Sys \vdash \Diamond stop = 1.$$

▷ Notice that *there are some musicians' choices* on which the system terminates iff

$$Sys \not\vdash \Box stop = 0.$$

The above statements can be effectively verified!

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

More Timed CCP Applications and Languages

- Music Composition and Performance (Rueda&Valencia 2004).
- Biological System (Olarte&Rueda 2005, Gutierrez&Perez&Rueda 2005).
- TimedGentzen (Saraswat 1995): A tcc-based programming language for reactive-systems implemented in Prolog.
- JCC (Saraswat&Gupta 2003): An integration of timed ccp into *JAVA*. See http://www.cse.psu.edu/∼saraswat/jcc.html

Related Work & Road Map

- Logic & Proof System for Timed CCP: [Gupta-Jagadeesan-Saraswat '94,'95]

Related Work & Road Map

Logic & Proof System for Timed CCP: [Gupta-Jagadeesan-Saraswat '94,'95]

- Logic & Proof System for *Nondeterministic* Timed CCP [DeBoer-Gabrielli-Meo '01].

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Related Work & Road Map

Logic & Proof System for Timed CCP: [Gupta-Jagadeesan-Saraswat '94,'95]

Logic & Proof System for *Nondeterministic* Timed CCP [DeBoer-Gabrielli-Meo '01].

- Logic & Proof System *Nondeterministic* Basic Timed CCP [Nielsen-Palamidessi-Valencia '02]. .

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Related Work & Road Map

- Logic & Proof System for Timed CCP: [Gupta-Jagadeesan-Saraswat '94,'95]
- Logic & Proof System for *Nondeterministic* Timed CCP [DeBoer-Gabbrielli-Meo '01].
- Logic & Proof System *Nondeterministic* Basic Timed CCP [Nielsen-Palamidessi-Valencia '02].
- Decidability of Verification [Valencia '03].

Related Work & Road Map

Logic & Proof System for Timed CCP: [Gupta-Jagadeesan-Saraswat '94,'95]

Logic & Proof System for *Nondeterministic* Timed CCP [DeBoer-Gabrielli-Meo '01].

Logic & Proof System *Nondeterministic* Basic Timed CCP [Nielsen-Palamidessi-Valencia '02].

Decidability of Verification [Valencia '03]

- Verification (Model Checking) for Timed CCP [Falaschi and Villanueva '03]

RoadMap:

Operational and Denotational Models for Timed CCP

Timed CCP Logic and its Applications

- Coming Next: Behavioral Equivalences.

Observations to Make of Processes

- Stimulus-response interaction

$$P = P_1 \stackrel{(c_1,c'_1)}{\Longrightarrow} P_2 \stackrel{(c_2,c'_2)}{\Longrightarrow} P_3 \stackrel{(c_3,c'_3)}{\Longrightarrow} \ldots$$

denoted by $P \stackrel{(\alpha,\alpha')}{\Longrightarrow}^{\omega}$ with $\alpha = c_1.c_2\ldots$ and $\alpha' = c'_1.c'_2\ldots$

Observable Behavior

- Input-Output $io(P) = \{(\alpha,\alpha') \mid P \stackrel{(\alpha,\alpha')}{\Longrightarrow}^{\omega}\}$
- Output $o(P) = \{\alpha' \mid P \stackrel{(\mathtt{true}^{\omega},\alpha')}{\Longrightarrow}^{\omega}\}$
- Strongest Postcondition $sp(P) = \{\alpha' \mid P \stackrel{(\_,\alpha')}{\Longrightarrow}^{\omega}\}$

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Behavioral Equivalences

Definition. *Let* $l \in \{o, io, sp\}$*. Define* $P \sim_l Q$ *iff* $l(P) = l(Q)$*.*

Unfortunately, neither $\sim_{io}$ *nor* $\sim_o$ *are congruences. Let* $\approx_{io}$ *and* $\approx_o$ *be the corresponding congruences.*

Theorem. $\approx_{io} = \approx_o \subset \sim_{io} \subset \sim_o$.

Distinguishing Context Characterizations

Theorem. *Given P, Q and $\sim \in \{\approx_o, \sim_{io}, \sim_{sp}\}$, one can construct a context $C_{\sim}^{(P,Q)}[.]$ such that:*

$$P \sim Q \quad \textit{if and only if} \quad C_{\sim}^{(P,Q)}[P] \ \sim_o \ C_{\sim}^{(P,Q)}[Q]$$

- Interesting consequence of the theorem:

> **Decidability** of all $\sim_{io}, \sim_{sp}, \approx_o$ and $\approx_{io}$ reduce to that of $\sim_o$.

- Interesting result introduced for the proof:

> Given P one can **construct a finite set** including all relevant inputs.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Behavioral Equivalence: Decidability.

Definition. *A star-free P is* **locally-deterministic** *iff all its summations occur outside of its local processes.*

Theorem. *Given a locally-deterministic P one can effectively construct a Büchi automaton B_P that recognizes $o(P)$.*

As a corollary,

Theorem. $\approx_o, \approx_{io}, \sim_{io}$, $\sim_{sp}$ ***are all decidable*** *for locally-deterministic processes.*

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Related Work & Road Map

- Decidability of Various Equivalences [Valencia '03]

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Related Work & Road Map

- Decidability of Various Equivalences [Valencia '03]
- Timed CCP Bisimilarity Equivalence and its Axiomatization [Tini '00]

RoadMap:

Operational and Denotational Models for Timed CCP

Timed CCP Logic and its Applications

Behavioral Equivalences

- Coming Next: Timed CCP Language Hierarchy.

Variants and their Expressive Power

Basic Timed CCP with the following alternatives for ***infinite behavior.***

- **tcc[Rec]**
  Recursive definitions $A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} P$ with $fv(P) \subseteq \{x_1, \ldots, x_n\}$.

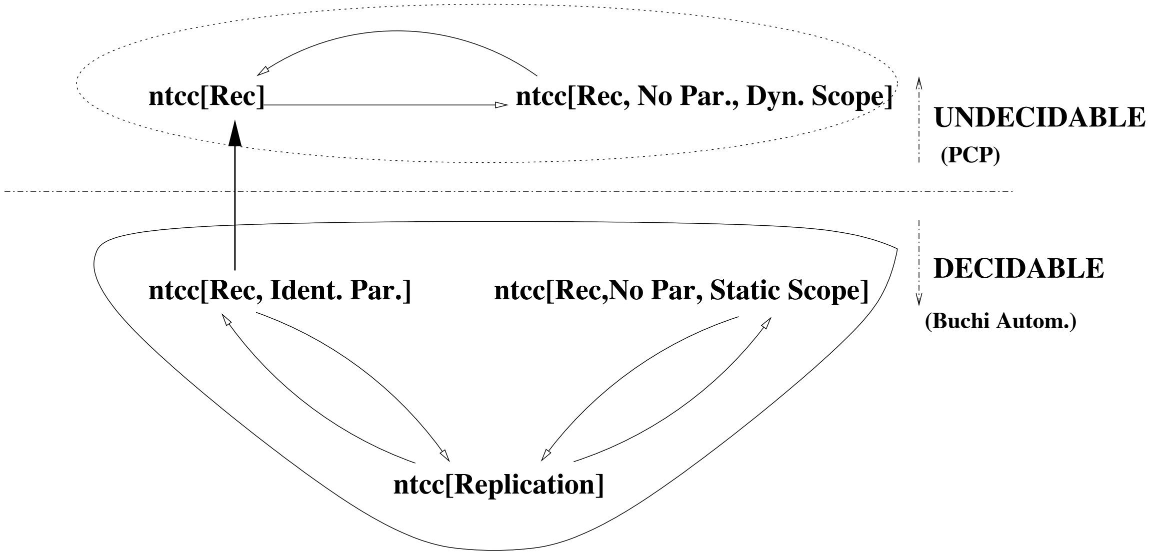
- **tcc[Rec, Identical Parameters]**
  As above but every call of A in P is of the form $A(x_1, \ldots, x_n)$.

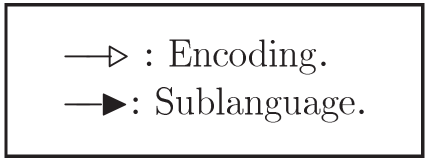
- **tcc[Rec, No Parameters, Dyn. Scoping]**
  Recursive definitions $A \stackrel{\text{def}}{=} P$ with Dynamic Scoping

- **tcc[Rec, No Parameters, Static Scoping]**
  Recursive definitions $A \stackrel{\text{def}}{=} P$ with Static Scoping.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

TCC Hierarchy and $\sim_{io}$ (un)decidability.





- Qualitative distinction between dynamic and static scope.

- The results have inspired similar results for CCS.

Final Remarks

Timed CCP combines the declarative view of *LTL* with the operational-behavioral view from *process calculi* .

Final Remarks

Timed CCP combines the declarative view of *LTL* with the operational-behavioral view from *process calculi* .

"...One of the outstanding challenges in concurrency is to find the right marriage between ***logic*** and ***behavioural*** *approaches". R. Milner.*

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Final Remarks

Timed CCP combines the declarative view of *LTL* with the operational-behavioral view from *process calculi* .

"...One of the outstanding challenges in concurrency is to find the right marriage between ***logic*** and ***behavioural*** *approaches". R. Milner.*

About Timed CCP:

- Simple ideas from concurrency and temporal logic.

Final Remarks

Timed CCP combines the declarative view of *LTL* with the operational-behavioral view from *process calculi* .

"...One of the outstanding challenges in concurrency is to find the right marriage between ***logic*** and ***behavioural*** *approaches". R. Milner.*

About Timed CCP:

- Simple ideas from concurrency and temporal logic.
- It expresses interesting real-world temporal situations.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Final Remarks

Timed CCP combines the declarative view of *LTL* with the operational-behavioral view from *process calculi* .

"...One of the outstanding challenges in concurrency is to find the right marriage between ***logic*** and ***behavioural*** *approaches". R. Milner.*

About Timed CCP:

- Simple ideas from concurrency and temporal logic.
- It expresses interesting real-world temporal situations.
- Formalization upon process algebra and logic.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Final Remarks

Timed CCP combines the declarative view of *LTL* with the operational-behavioral view from *process calculi* .

"...One of the outstanding challenges in concurrency is to find the right marriage between ***logic*** and ***behavioural*** *approaches". R. Milner.*

About Timed CCP:

- Simple ideas from concurrency and temporal logic.
- It expresses interesting real-world temporal situations.
- Formalization upon process algebra and logic.
- Techniques from a denotational semantics and process logic.

Ongoing and Future Work

- Implementation of Automatic Tools for analyzing Timed CCP Processes.
- Probabilistic Timed CCP (Olarte&Rueda 2005, Perez 2005).
- **Secure CCP** *(Ecole Polytechnique, IBM, Univ. Pisa, Javeriana, Univalle).*
- *Timed CCP for reasoning about Biological Systems* (Olarte&Rueda 2005, Gutierrez&Perez 2005).

Examples of Observables

$$\underbrace{\textbf{when } a \textbf{ do next } \begin{array}{l}\textbf{when } b \textbf{ do next tell}(d) \\ + \\ \textbf{when } c \textbf{ do next tell}(e)\end{array}}_{P} \quad , \quad \underbrace{\begin{array}{l}\textbf{when } a \textbf{ do next when } b \textbf{ do next tell}(d) \\ + \\ \textbf{when } a \textbf{ do next when } c \textbf{ do next tell}(e)\end{array}}_{Q}$$

Assuming a, b, c, d and e mutually exclusive:

- $o(P) = o(Q) = \{\texttt{true}^\omega\}$.
- $io(P) \neq io(Q)$: If $\alpha = a.c.\texttt{true}^\omega$ then $(\alpha, \alpha) \in io(Q)$ but $(\alpha, \alpha) \notin io(P)$
- $sp(P) \neq sp(Q)$: If $\alpha = a.c.\texttt{true}^\omega$ then $\alpha \in sp(Q)$ but $\alpha \notin sp(P)$.