An introduction to Concurrency Theory

Communicating and Mobile Systems

Frank D. Valencia

French National Center for Scientific Research (CNRS)
Computer Science Laboratory of École Polytechnique de Paris (LIX)

Oct 2005/CLEI'05

Concurrency is everywhere...

Concurrent Systems

Multiple **agents** (**processes**) that interact among each other.

Some fundamental concurrent systems:

- Reactive Systems.
- Synchronous Communicating Systems
- Mobile Systems.

Concurrent systems can combine:

Example

E.g., ***The Internet*** (a complex one!)

Concurrency is everywhere...

Concurrent Systems

Multiple **agents** (**processes**) that interact among each other.

Some fundamental concurrent systems:

- Reactive Systems.
- Synchronous Communicating Systems
- Mobile Systems.

Concurrent systems can combine:

Example

E.g., ***The Internet*** (a complex one!)

Concurrency is everywhere...

Concurrent Systems

Multiple **agents** (**processes**) that interact among each other.

Some fundamental concurrent systems:

- Reactive Systems.
- Synchronous Communicating Systems
- Mobile Systems.

Concurrent systems can combine:

Example

E.g., ***The Internet*** (a complex one!)

Concurrency is everywhere...

Concurrent Systems

Multiple **agents** (**processes**) that interact among each other.

Some fundamental concurrent systems:

- Reactive Systems.
- Synchronous Communicating Systems
- Mobile Systems.

Concurrent systems can combine:

Example

E.g., ***The Internet*** (a complex one!)

Concurrency is everywhere...

Concurrent Systems

Multiple **agents** (**processes**) that interact among each other.

Some fundamental concurrent systems:

- Reactive Systems.
- Synchronous Communicating Systems
- Mobile Systems.

Concurrent systems can combine:

Example

E.g., ***The Internet*** (a complex one!)

Concurrency is everywhere...

Concurrent Systems

Multiple **agents** (**processes**) that interact among each other.

Some fundamental concurrent systems:

- Reactive Systems.
- Synchronous Communicating Systems
- Mobile Systems.

Concurrent systems can combine:

Example

E.g., ***The Internet*** (a complex one!)

Concurrency is everywhere...

Concurrent Systems

Multiple **agents** (**processes**) that interact among each other.

Some fundamental concurrent systems:

- Reactive Systems.
- Synchronous Communicating Systems
- Mobile Systems.

Concurrent systems can combine:

Example

E.g., ***The Internet*** (a complex one!)

Concurrency: A Serious Challenge...

Models of Concurrency

Formal Models to describe and analyze concurrent systems.

Concurrency: A Serious Challenge...

Models of Concurrency

Formal Models to describe and analyze concurrent systems.

As other models of reality, these models must be

- be simple
- be expressive,
- be formal
- provide reasoning techniques.

Just like $\lambda-$calculus model of sequential computation !

Concurrency: A Serious Challenge...

Models of Concurrency

Formal Models to describe and analyze concurrent systems.

Models for sequential computation don't apply. ***Concurrent computation*** is usually

- *Non-Terminating*
- *Reactive* (or *Interactive*)
- *Nondeterministic* (*Unpredictable*).

Concurrency: A Serious Challenge...

Models of Concurrency

Formal Models to describe and analyze concurrent systems.

In concurrency theory

- Each model focuses in a fundamental phenomenon: E.g., Synchrony and Mobility.
- But there is no yet a "*cannonical model*".

...Probably because concurrency is a very broad (young) area.

Concurrency: A Serious Challenge...

Models of Concurrency

Formal Models to describe and analyze concurrent systems.

Some well established model of concurrency:

- Process Calculi (Like λ-calculi)
  - **CCS (Synchronous communication)**
  - **π-calculus (CCS Extension to Mobility)**
- Petri Nets (Like Automata)

Mobility

What kind of ***mobility*** will we consider in this tutorial ?

- Processes move.
- **Links** move$\surd$

The last one, because it is the π-calculus' choice; for flexibility and simplicity.

- π-calculus has the ability of sending private and public *links* (*names*).

Mobility

What kind of ***mobility*** will we consider in this tutorial ?

- Processes move.
- **Links** move$\surd$

The last one, because it is the π-calculus' choice; for flexibility and simplicity.

- π-calculus has the ability of sending private and public *links* (*names*).

Mobility

What kind of ***mobility*** will we consider in this tutorial ?

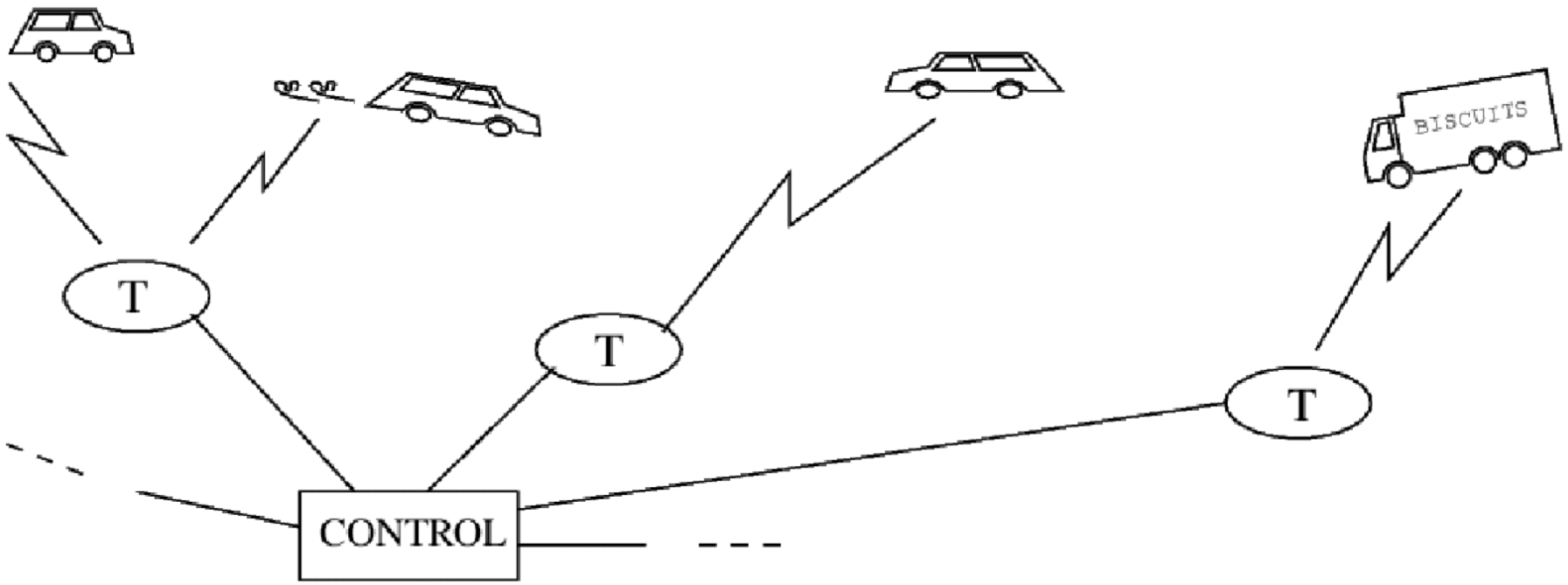
- Processes move.
- ***Links*** move$\surd$

The last one, because it is the π-calculus' choice; for flexibility and simplicity.

- π-calculus has the ability of sending private and public *links* (*names*).

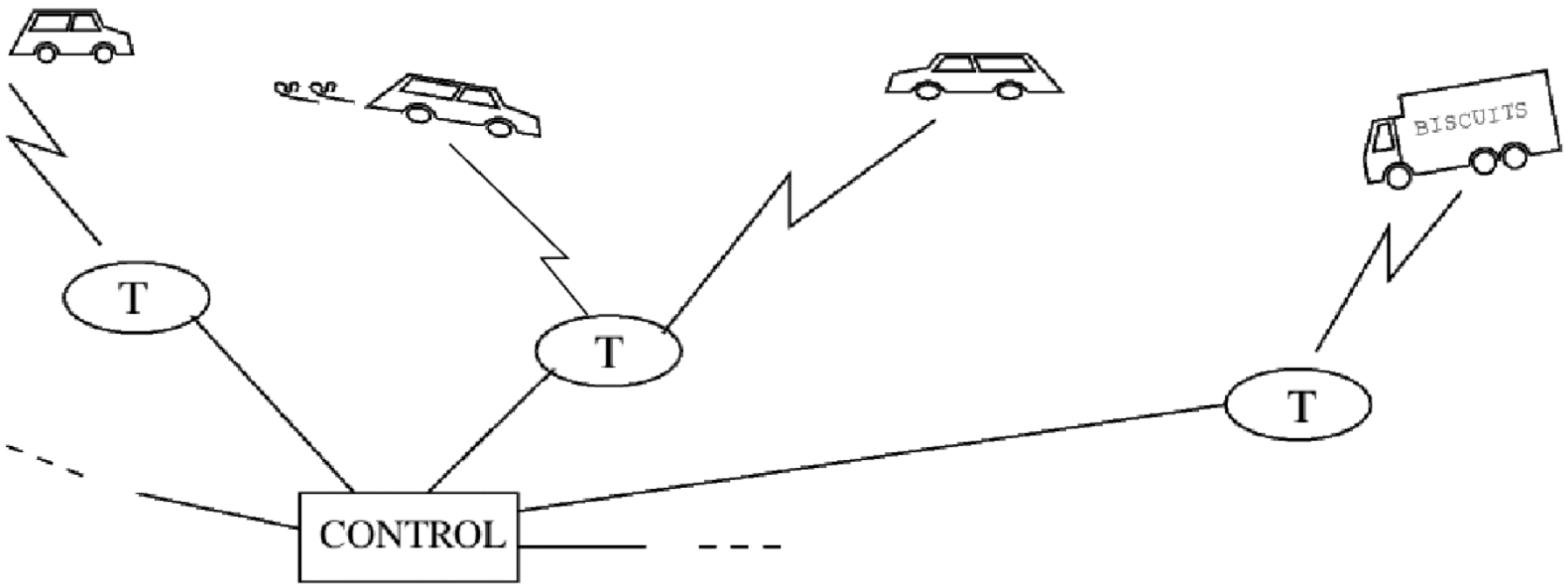
Link Mobility

Mobile phones

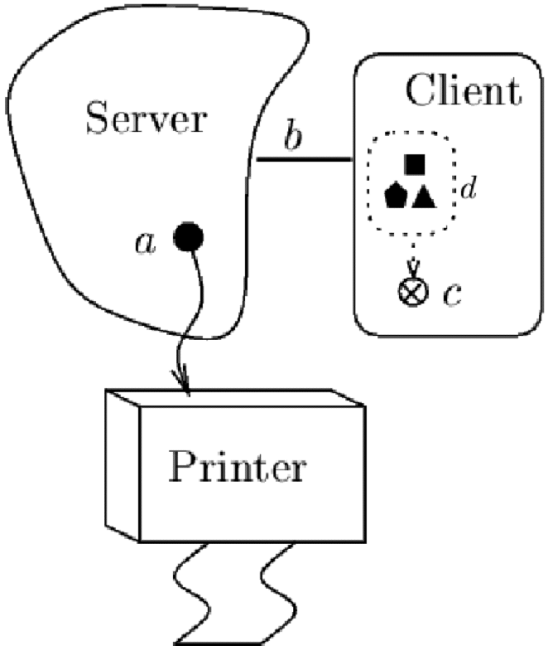


Link Mobility

Mobile phones



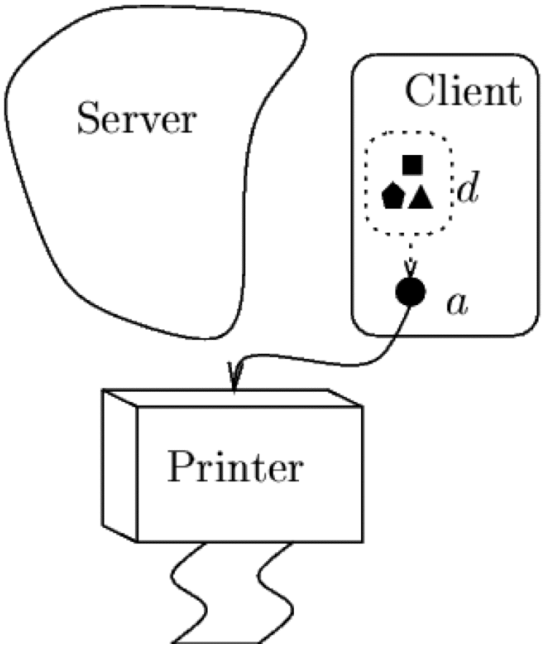
A flavour of the pi-calculus



In the π calculus:

$(\nu a)(\bar{b}a.S \parallel a(x).P) \parallel b(c).\bar{c}d.C$

A flavour of the pi-calculus



In π the pi-calculus:

$$(\nu a)(\bar{b}a.S \parallel a(x).P) \parallel b(c).\bar{c}d.C \longrightarrow S \parallel (\nu a)(a(x).P \parallel \bar{a}d.C)$$

Outline

Outline

Classic Automata Theory

Automata

Definition

An automata A over an alphabet Act is a tuple (Q, Q_0, Q_f, T) where

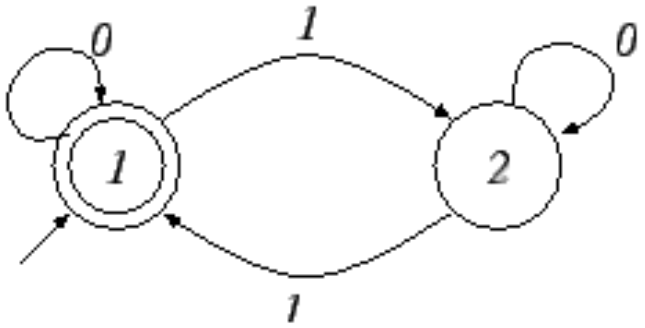
- $S(A) = Q = \{q_0, q_1, \ldots\}$ is the set of **states**,
- $S_0(A) = Q_0 \subseteq Q$ is the set of **initial states**,
- $S_f(A) = Q_f \subseteq Q$ is the set of **accepting** (or final) **states**,
- $T(A) = T \subseteq Q \times Act \times Q$ is the set of **transitions**.

Write $q \xrightarrow{a} q'$ for $(q, a, q') \in T$. Omit Act when it is understood.

Classic Automata Theory

Automata

Example: A Typical Graph Representation



Classic Automata Theory

Regular Sets and Expressions

Definition (Acceptance, Regularity)

- A ***accepts*** $a_1 \ldots a_n$ iff there is a path $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$ in $T(A)$ with $q_0 \in S_o(A)$ and $q_n \in S_f(A)$.
- The ***Language*** of (or recognized by) A, $L(A)$, is the set of sequences accepted by A.
- ***Regular sets*** are those recognized by ***finite-state*** *automata* (**FSA**).

Classic Automata Theory

Regular Sets and Expressions

Definition (Acceptance, Regularity)

- A ***accepts*** $a_1 \ldots a_n$ iff there is a path $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$ in $T(A)$ with $q_0 \in S_o(A)$ and $q_n \in S_f(A)$.
- The ***Language*** of (or recognized by) A, $L(A)$, is the set of sequences accepted by A.
- ***Regular sets*** are those recognized by ***finite-state*** *automata* (**FSA**).

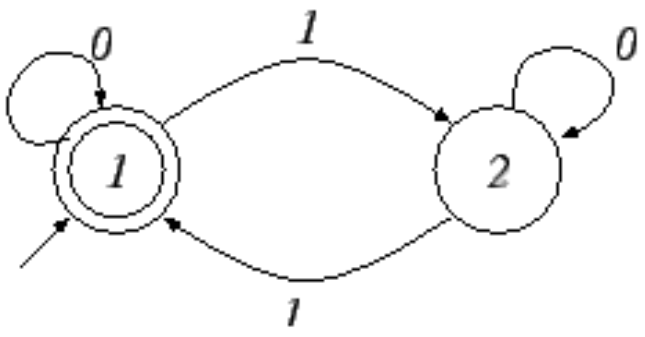
Fact

Regular Expressions *(e.g.* $a.(b+c)^*$*) are* ***as expressive*** *as FSA.*

Classic Automata Theory

Regular Sets and Expressions

Example



- An string accepted: 10001.
- The language accepted: All binary strings with an even number of 1's.
- A Corresponding Regular Expression: 0*(10*10*)*

Classic Automata Theory

Some Nice Properties and Exercises

Fact

1. *Deterministic and NonDeterministic FSA are* ***equally expressive****.*
2. *Regular sets are* ***closed*** *under (a) union, (b) complement, (c) intersection.*

Exercises

1. Prove 2.b and 2.c.
2. Prove that emptiness problem of a given FSA is *decidable*.
3. Prove that language equivalence of two given FSA is *decidable*.

Classic Automata Theory

Some Nice Properties and Exercises

Fact

1. *Deterministic and NonDeterministic FSA are* ***equally expressive****.*
2. *Regular sets are* ***closed*** *under (a) union, (b) complement, (c) intersection.*

Exercises

1. Prove 2.b and 2.c.
2. Prove that emptiness problem of a given FSA is ***decidable***.
3. Prove that language equivalence of two given FSA is ***decidable***.

Classic Automata Theory

What is good for ?

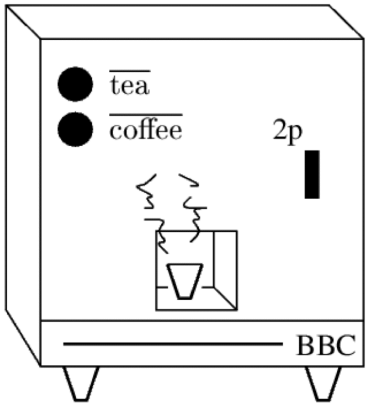
Classic Automata Theory is solid and foundational with several applications in computer science. BUT...

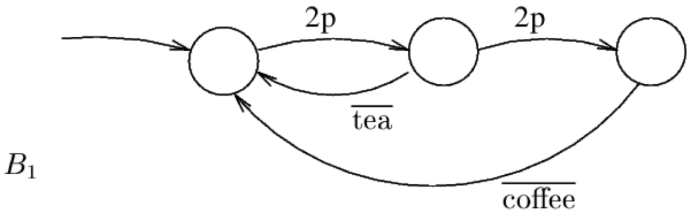
Classic Automata Theory

What is good for ?

Classic Automata Theory is solid and foundational with several applications in computer science. BUT...

Example: A FSA Vending Machine Representation



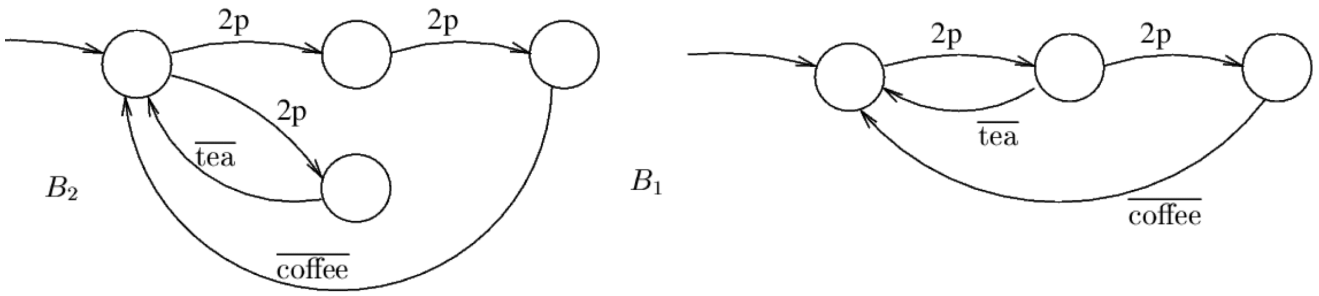


Classic Automata Theory

What is good for ?

Classic Automata Theory is solid and foundational with several applications in computer science. BUT...

Example: A FSA Vending Machine Representation



- $L(B_1) = L(B_2)$ but B_2 would represent an annoying vending

Classic Automata Theory

What is good for ?

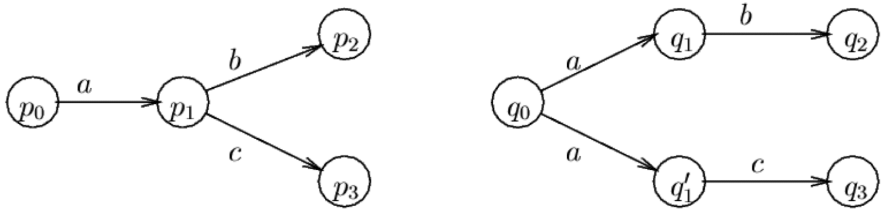
Classic Automata Theory is solid and foundational with several applications in computer science. BUT...

Fact

Language equivalence may be too weak for ***interactive behavior****.*

Classic Automata Theory.

The Problem with Language Equivalence



In automata theory $a \cdot (b + c) = a \cdot b + a \cdot c$. I.e., p_0 and q_o are equivalent.

- We need stronger equivalences that does not validate the above.

Outline

Bisimilarity: The Equivalence of Reactive Systems.

Transition Systems

Transition systems are just automata in which initial and final states are irrelevant.

Bisimilarity: The Equivalence of Reactive Systems.

Simulation

Let T be transition system. A relation $R \subseteq S(T) \times S(T)$ is a ***simulation*** iff for each $(p, q) \in R$:

▷ If $p \xrightarrow{a} p'$ then there is q' s.t. $q \xrightarrow{a} q'$ and $(p', q') \in R$.

Bisimilarity: The Equivalence of Reactive Systems.

Bisimulation

A relation R is a ***bisimulation*** iff R and its *converse* R^{-1} are both simulations.

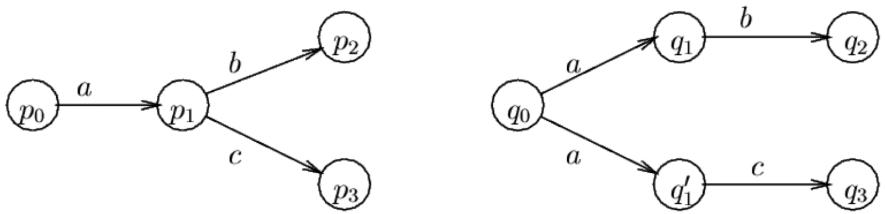
Bisimilarity: The Equivalence of Reactive Systems.

Similaridad & Bisimilaridad

- ▷ p **simulates** q iff there exists a *simulation* R s.t $(p, q) \in R$.
- ▷ p and q are ***bisimilar,*** written $p \sim q$, if there exists a ***bisimulation*** R s.t. $(p, q) \in R$.

Bisimilarity: The Equivalence of Reactive Systems.

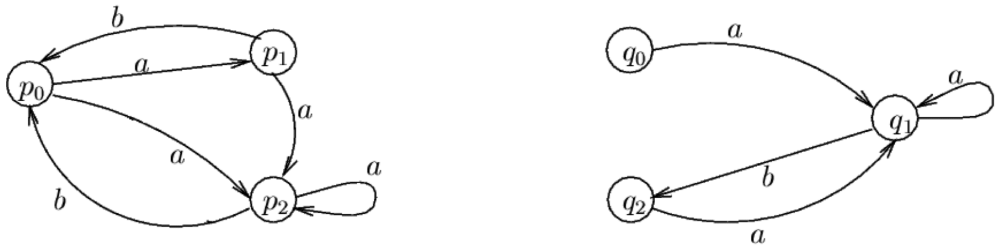
Example: Trace Equivalence is not Bisimilarity



Here p_0 simulates q_0 but q_0 does not simulate p_0, so p_0 and q_0 are not bisimilar.

Bisimilarity: The Equivalence of Reactive Systems.

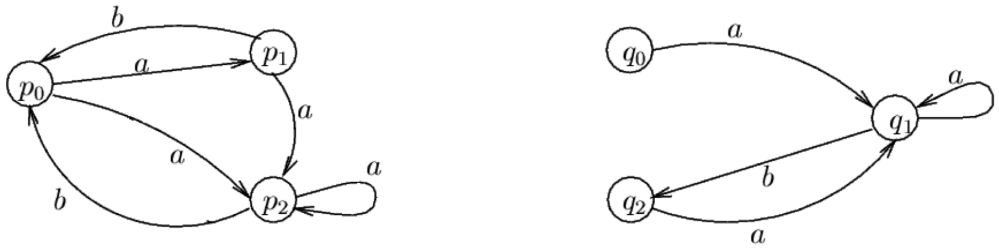
Example: Bisimilarity is not graph equivalence



Bisimilarity equates LTS with different graph structure: E.g., p_0 and q_0 are bisimilar. But how do we prove they are bisimilar ?

Bisimilarity: The Equivalence of Reactive Systems.

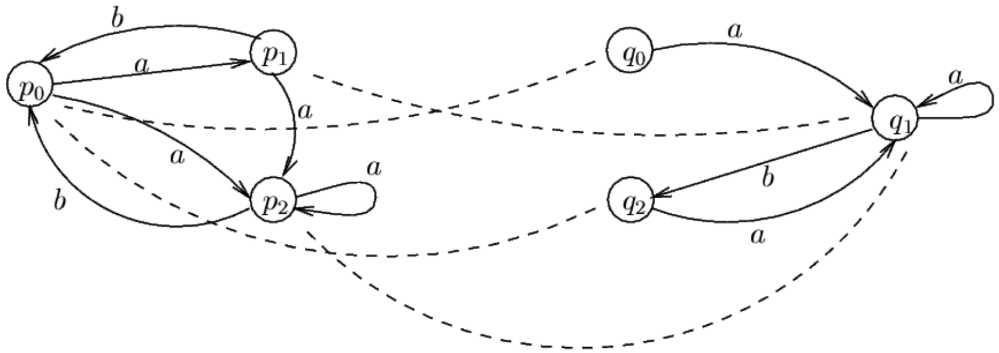
Example: Proving Bisimilarity



To prove that p_0 and q_0 are bisimilar simply exhibit a **bisimulation** R. E.g. $R = \{(p_0, q_0), (p_0, q_2), (p_1, q_1), (p_2, q_1)\}$.

Bisimilarity: The Equivalence of Reactive Systems.

Example: Proving Bisimilarity



To prove that p_0 and q_0 are bisimilar simply exhibit a **bisimulation** R. E.g. $R = \{(p_0, q_0), (p_0, q_2), (p_1, q_1), (p_2, q_1)\}$.

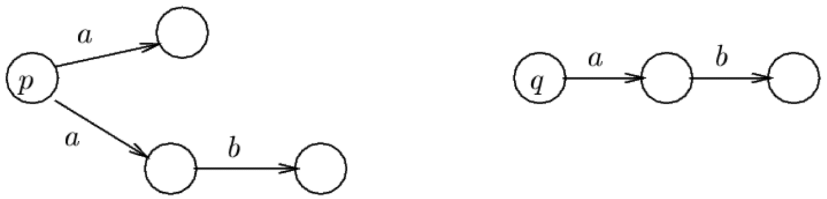
Bisimilarity: The Equivalence of Reactive Systems.

Example: Bisimilarity and simulation

Suppose that p simulates q and q simulates p. Are p and q necesarly bisimilar ?

Bisimilarity: The Equivalence of Reactive Systems.

Example: Bisimilarity and simulation



No! Here p simulates$q$ and q simulates p but they are not bisimilar. Notice that p is less ***reliable*** than q; i.e., it ***deadlocks*** after a.

Outline

Process Calculi

Process calculi *treat processes much like λ-calculus treats computable functions:*

- Structure of ***terms*** represent the structure of processes. E.g.:

$$P \parallel Q$$

- ***Operational Semantics*** to represent computational steps. E.g.:

$$\text{PAR}\ \frac{P \longrightarrow P'}{P \parallel Q \longrightarrow P' \parallel Q}$$

Process Calculi

Process calculi *treat processes much like λ-calculus treats computable functions:*

- Structure of ***terms*** represent the structure of processes. E.g.:

$$P \parallel Q$$

- ***Operational Semantics*** to represent computational steps. E.g.:

$$\text{PAR}\ \frac{P \longrightarrow P'}{P \parallel Q \longrightarrow P' \parallel Q}$$

Process Calculi

Process calculi *treat processes much like λ-calculus treats computable functions:*

- Structure of ***terms*** represent the structure of processes. E.g.:

$$P \parallel Q$$

- ***Operational Semantics*** to represent computational steps. E.g.:

$$\text{PAR}\ \frac{P \longrightarrow P'}{P \parallel Q \longrightarrow P' \parallel Q}$$

Typical Issues in Developing Process Calculi

- **Which process constructs fit the intended phenomena?**
  E.g. atomic actions, parallelism, nondeterminism.
- **How should these constructs be given meaning ?**
  E.g. Operational Semantics, Denotational Semantics.
- **How should processes be compared ?**
  E.g. Observable Behavior, Process Equivalences.
- **How should process properties be specified and proved ?**
  E.g. Logic for expressing process specifications (like in Hoare's Logic).
- **How expressive is the calulus ?**
  E.g. Expressiveness of a nice fragment of the calculus.

Typical Issues in Developing Process Calculi

- **Which process constructs fit the intended phenomena?**
  E.g. atomic actions, parallelism, nondeterminism.
- **How should these constructs be given meaning ?**
  E.g. Operational Semantics, Denotational Semantics.
- **How should processes be compared ?**
  E.g. Observable Behavior, Process Equivalences.
- **How should process properties be specified and proved ?**
  E.g. Logic for expressing process specifications (like in Hoare's Logic).
- **How expressive is the calulus ?**
  E.g. Expressiveness of a nice fragment of the calculus.

Typical Issues in Developing Process Calculi

- **Which process constructs fit the intended phenomena?**
  E.g. atomic actions, parallelism, nondeterminism.
- **How should these constructs be given meaning ?**
  E.g. Operational Semantics, Denotational Semantics.
- **How should processes be compared ?**
  E.g. Observable Behavior, Process Equivalences.
- **How should process properties be specified and proved ?**
  E.g. Logic for expressing process specifications (like in Hoare's Logic).
- **How expressive is the calulus ?**
  E.g. Expressiveness of a nice fragment of the calculus.

Typical Issues in Developing Process Calculi

- **Which process constructs fit the intended phenomena?**
  E.g. atomic actions, parallelism, nondeterminism.
- **How should these constructs be given meaning ?**
  E.g. Operational Semantics, Denotational Semantics.
- **How should processes be compared ?**
  E.g. Observable Behavior, Process Equivalences.
- **How should process properties be specified and proved ?**
  E.g. Logic for expressing process specifications (like in Hoare's Logic).
- **How expressive is the calulus ?**
  E.g. Expressiveness of a nice fragment of the calculus.

Typical Issues in Developing Process Calculi

- **Which process constructs fit the intended phenomena?**
  E.g. atomic actions, parallelism, nondeterminism.
- **How should these constructs be given meaning ?**
  E.g. Operational Semantics, Denotational Semantics.
- **How should processes be compared ?**
  E.g. Observable Behavior, Process Equivalences.
- **How should process properties be specified and proved ?**
  E.g. Logic for expressing process specifications (like in Hoare's Logic).
- **How expressive is the calulus ?**
  E.g. Expressiveness of a nice fragment of the calculus.

Outline

CCS Syntax

Basic Entities of CCS:

- A set $\mathcal{N} = a, b, \ldots$ of **names** and $\overline{\mathcal{N}} = \{\overline{a} \mid a \in \mathcal{N}\}$ of **co-names.**
- A set $\mathcal{L} = \mathcal{N} \cup \overline{\mathcal{N}}$ of **labels** (ranged over by $l, l', \ldots$).
- A set $Act = \mathcal{L} \cup \{\tau\}$ of **actions** (ranged over by $\mathbf{a}, \mathbf{b}, \ldots$).
- Action τ is called the *silent or unobservable action.*
- Actions a and $\overline{a}$ are viewed as being "complementary". I.e., $a = \overline{\overline{a}}$.

CCS Syntax

Definition (CCS Syntax of Processes)

$$P, Q, \ldots := 0 \mid \mathbf{a}.P \mid P \parallel Q \mid P + Q \mid (\nu a)P \mid A(a_1, \ldots, a_n)$$

- **Bound names** of P, $fn(P)$: Those with a bound occurrence in P.
- **Free names of** P, $bn(P)$: Those with a not bound occurrence in P.
- For each (call) $A(a_1, \ldots, a_n)$ there is a unique **process definition** $A(b_1 \ldots b_n) = P$, with $fv(P) \subseteq \{b_1, \ldots, b_n\}$.
- The set of all processes is denoted by $\mathcal{P}$.

CCS: Operational Semantics.

$$\text{ACT}\quad \frac{}{\mathsf{a}.P \xrightarrow{\mathsf{a}} P}$$

$$\text{SUM}_1\quad \frac{P \xrightarrow{\mathsf{a}} P'}{P + Q \xrightarrow{\mathsf{a}} P'} \qquad \text{SUM}_2\quad \frac{Q \xrightarrow{\mathsf{a}} Q'}{P + Q \xrightarrow{\mathsf{a}} Q'}$$

$$\text{COM}_1\quad \frac{P \xrightarrow{\mathsf{a}} P'}{P \parallel Q \xrightarrow{\mathsf{a}} P' \parallel Q} \qquad \text{COM}_2\quad \frac{Q \xrightarrow{\mathsf{a}} Q'}{P \parallel Q \xrightarrow{\mathsf{a}} P \parallel Q'}$$

$$\text{COM}_3\quad \frac{P \xrightarrow{l} P' \quad Q \xrightarrow{\bar{l}} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'}$$

$$\text{RES}\quad \frac{P \xrightarrow{\mathsf{a}} P'}{(\nu a)P \xrightarrow{\mathsf{a}} (\nu a)P'} \quad \text{if } \mathsf{a} \neq a \text{ and } \mathsf{a} \neq \bar{a}$$

$$\text{REC}\quad \frac{P_A[b_1, \ldots, b_n / a_1, \ldots, a_n] \xrightarrow{\mathsf{a}} P'}{A\langle b_1, \ldots, b_n \rangle \xrightarrow{\mathsf{a}} P'} \quad \text{if } A(a_1, \ldots, a_n) \stackrel{\text{def}}{=} P_A$$

Outline

CCS: Bisimilarity

Labeled Transition System of CCS

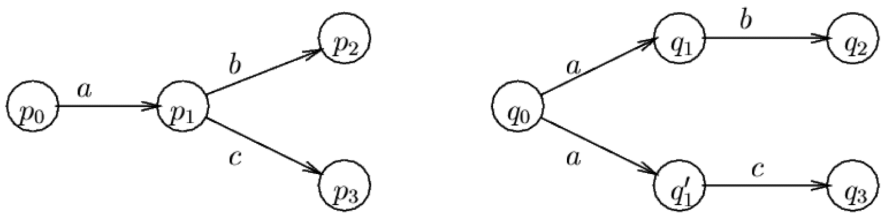
The LTS *of CCS has* $\mathcal{P}$ *as its states and its transitions given by the operational semantics.* Define $P \sim Q$ iff the states corresponding to P and Q are bisimilar.

CCS: Bisimilarity

Labeled Transition System of CCS

The LTS *of CCS has $\mathcal{P}$ as its states and its transitions given by the operational semantics.* Define $P \sim Q$ iff the states corresponding to P and Q are bisimilar.

Example: Processes and LTS states



$P = a.(b.0 + c.0)$ corresponds to p_0 and $Q = a.b.0 + a.c.0$ corresponds to q_0. Hence $P \not\sim Q$.

CCS: Bisimilarity

Labeled Transition System of CCS

The LTS *of CCS has $\mathcal{P}$ as its states and its transitions given by the operational semantics.* Define $P \sim Q$ iff the states corresponding to P and Q are bisimilar.

Some Basic Equalities

- $P \parallel Q \sim Q \parallel P$
- $P \parallel 0 \sim P$
- $(P \parallel Q) \parallel R \sim P \parallel (Q \parallel R)$
- $(\nu a)0 \sim 0$
- $P \parallel (\nu a)Q \sim (\nu a)(P \parallel Q)$?
- $(\nu a)P \sim (\nu b)P[b/a]$?

CCS: The expansion law.

- Notice that $a.0 \parallel b.0$ is bisimilar to $a.b.0 + b.a.0$.
- More generally, we have the **expansion law**:
  $(\nu\vec{a})(P_1 \parallel \ldots \parallel P_n) \quad \sim$

$$\Sigma\{\mathbf{a}_i.(\nu\vec{a})(P_1 \parallel \ldots \parallel P_i' \parallel \ldots \parallel P_n \mid P_i \xrightarrow{\mathbf{a}_i} P_i',\ \overline{\mathbf{a}}, \mathbf{a}_i \notin \vec{a}\}$$
$$+$$
$$\Sigma\{\tau.(\nu\overline{a})(P_1 \parallel \ldots \parallel P_i' \parallel \ldots \parallel P_j' \ldots \parallel P_n \mid P_i \xrightarrow{l} P_i',\ P_j \xrightarrow{\overline{l}} P_j'\}$$

- So, every move in $(\nu\vec{a})(P_1 \parallel \ldots \parallel P_n)$ is either one of the P_i or a communication between some P_i and P_j.

CCS: The expansion law.

- Notice that $a.0 \parallel b.0$ is bisimilar to $a.b.0 + b.a.0$.
- More generally, we have the **expansion law**:
  $(\nu\vec{a})(P_1 \parallel \ldots \parallel P_n) \quad \sim$

$$\Sigma\{\mathbf{a}_i.(\nu\vec{a})(P_1 \parallel \ldots \parallel P'_i \parallel \ldots \parallel P_n \mid P_i \xrightarrow{\mathbf{a}_i} P'_i,\ \overline{\mathbf{a}}, \mathbf{a}_i \notin \vec{a}\}$$
$$+$$
$$\Sigma\{\tau.(\nu\overline{a})(P_1 \parallel \ldots \parallel P'_i \parallel \ldots \parallel P'_j \ldots \parallel P_n \mid P_i \xrightarrow{l} P'_i\,,\ P_j \xrightarrow{\overline{l}} P'_j\}$$

- So, every move in $(\nu\vec{a})(P_1 \parallel \ldots \parallel P_n)$ is either one of the P_i or a communication between some P_i and P_j.

CCS: The expansion law.

- Notice that $a.0 \parallel b.0$ is bisimilar to $a.b.0 + b.a.0$.
- More generally, we have the **expansion law**:
  $(\nu\vec{a})(P_1 \parallel \ldots \parallel P_n) \quad \sim$

$$\Sigma\{\mathbf{a}_i.(\nu\vec{a})(P_1 \parallel \ldots \parallel P'_i \parallel \ldots \parallel P_n \mid P_i \xrightarrow{\mathbf{a}_i} P'_i,\ \overline{\mathbf{a}}, \mathbf{a}_i \notin \vec{a}\}$$
$$+$$
$$\Sigma\{\tau.(\nu\overline{a})(P_1 \parallel \ldots \parallel P'_i \parallel \ldots \parallel P'_j \ldots \parallel P_n \mid P_i \xrightarrow{l} P'_i\,,\ P_j \xrightarrow{\overline{l}} P'_j\}$$

- So, every move in $(\nu\vec{a})(P_1 \parallel \ldots \parallel P_n)$ is either one of the P_i or a communication between some P_i and P_j.

Congruence issues

- Suppose that $P \sim Q$. We would like $P \parallel R \sim Q \parallel R$.
- More generally we would like

$$C[P] \sim C[Q]$$

  where $C[.]$ is a **process context**. I.e., we want $\sim$ to be a **congruence**.
- The notion of congruence allows us to replace "*equals with equals*".
- In fact, $\sim$ is a congruence. How can we prove this ?.

Congruence issues

- Suppose that $P \sim Q$. We would like $P \parallel R \sim Q \parallel R$.
- More generally we would like

$$C[P] \sim C[Q]$$

where $C[.]$ is a **process context**. I.e., we want $\sim$ to be a **congruence**.

- The notion of congruence allows us to replace "*equals with equals*".
- In fact, $\sim$ is a congruence. How can we prove this ?.

Congruence issues

- Suppose that $P \sim Q$. We would like $P \parallel R \sim Q \parallel R$.
- More generally we would like

$$C[P] \sim C[Q]$$

  where $C[.]$ is a **process context**. I.e., we want $\sim$ to be a **congruence**.
- The notion of congruence allows us to replace "*equals with equals*".
- In fact, $\sim$ is a congruence. How can we prove this ?.

Congruence issues

- Suppose that $P \sim Q$. We would like $P \parallel R \sim Q \parallel R$.
- More generally we would like

$$C[P] \sim C[Q]$$

where $C[.]$ is a **process context**. I.e., we want $\sim$ to be a **congruence**.

- The notion of congruence allows us to replace "*equals with equals*".
- In fact, $\sim$ is a congruence. How can we prove this ?.

Observable Behavior

- In principle, P and Q should be ***equivalent*** iff another process (the environment, an observer) cannot ***observe*** any difference in their behavior.
- Notice $\tau.P \not\sim P$, although τ is an **unobservable** action. So $\sim$ could be too *strong*:).
- We look for other notion of equivalence focused in terms of *observable* actions (i.e., actions $\xrightarrow{l}$, $l \in \mathcal{L}$).

Observable Behavior

- In principle, P and Q should be ***equivalent*** iff another process (the environment, an observer) cannot ***observe*** any difference in their behavior.
- Notice $\tau.P \not\sim P$, although τ is an **unobservable** action. So $\sim$ could be too *strong*:).
- We look for other notion of equivalence focused in terms of *observable* actions (i.e., actions $\xrightarrow{l}$, $l \in \mathcal{L}$).

Observable Behavior

- In principle, P and Q should be ***equivalent*** iff another process (the environment, an observer) cannot ***observe*** any difference in their behavior.
- Notice $\tau.P \not\sim P$, although τ is an **unobservable** action. So $\sim$ could be too *strong*:).
- We look for other notion of equivalence focused in terms of *observable* actions (i.e., actions $\xrightarrow{l}$, $l \in \mathcal{L}$).

Outline

Observations

- Think of any $\xrightarrow{l}$ with $l \neq \tau$ as an **observation**.
- Think of an **experiment** e as a sequence $l_1.l_2 \ldots l_n$ of observable actions.
- **Notation:** If $s = \mathbf{a}_1 \ldots \mathbf{a}_n \in Act^*$ then define

$$\stackrel{s}{\Longrightarrow} = (\xrightarrow{\tau})^* \xrightarrow{\mathbf{a}_1} (\xrightarrow{\tau})^* \ldots (\xrightarrow{\tau})^* \xrightarrow{\mathbf{a}_n} (\xrightarrow{\tau})^*$$

- Notice that $\stackrel{e}{\Longrightarrow}$ for $e = l_1.l_2 \ldots l_n \in \mathcal{L}$ denotes a sequence of observable actions inter-spread with τ actions: ***The notion of experiment***.

Observations

- Think of any $\xrightarrow{l}$ with $l \neq \tau$ as an **observation**.
- Think of an **experiment** e as a sequence $l_1.l_2 \ldots l_n$ of observable actions.
- **Notation:** If $s = \mathbf{a}_1 \ldots \mathbf{a}_n \in Act^*$ then define

$$\stackrel{s}{\Longrightarrow} = (\xrightarrow{\tau})^* \xrightarrow{\mathbf{a}_1} (\xrightarrow{\tau})^* \ldots (\xrightarrow{\tau})^* \xrightarrow{\mathbf{a}_n} (\xrightarrow{\tau})^*$$

- Notice that $\stackrel{e}{\Longrightarrow}$ for $e = l_1.l_2 \ldots l_n \in \mathcal{L}$ denotes a sequence of observable actions inter-spread with τ actions: ***The notion of experiment***.

Observations

- Think of any $\xrightarrow{l}$ with $l \neq \tau$ as an **observation**.
- Think of an **experiment** e as a sequence $l_1.l_2 \ldots l_n$ of observable actions.
- **Notation:** If $s = \mathbf{a}_1 \ldots \mathbf{a}_n \in Act^*$ then define

$$\stackrel{s}{\Longrightarrow} = (\xrightarrow{\tau})^* \xrightarrow{\mathbf{a}_1} (\xrightarrow{\tau})^* \ldots (\xrightarrow{\tau})^* \xrightarrow{\mathbf{a}_n} (\xrightarrow{\tau})^*$$

- Notice that $\stackrel{e}{\Longrightarrow}$ for $e = l_1.l_2 \ldots l_n \in \mathcal{L}$ denotes a sequence of observable actions inter-spread with τ actions: ***The notion of experiment***.

Observations

- Think of any $\xrightarrow{l}$ with $l \neq \tau$ as an **observation**.
- Think of an **experiment** e as a sequence $l_1.l_2 \ldots l_n$ of observable actions.
- **Notation:** If $s = \mathbf{a}_1 \ldots \mathbf{a}_n \in Act^*$ then define

$$\stackrel{s}{\Longrightarrow} = (\xrightarrow{\tau})^* \xrightarrow{\mathbf{a}_1} (\xrightarrow{\tau})^* \ldots (\xrightarrow{\tau})^* \xrightarrow{\mathbf{a}_n} (\xrightarrow{\tau})^*$$

- Notice that $\stackrel{e}{\Longrightarrow}$ for $e = l_1.l_2 \ldots l_n \in \mathcal{L}$ denotes a sequence of observable actions inter-spread with τ actions: ***The notion of experiment***.

Trace Equivalence

Definition (Trace Equivalence)

P and Q are ***trace equivalent***, written $P \sim_t Q$, iff for every experiment (here called trace) $e = l_1 \ldots l_n \in \mathcal{L}^*$

$$P \stackrel{e}{\Longrightarrow} \quad \text{iff} \quad Q \stackrel{e}{\Longrightarrow}$$

Observations:

- $\tau.P \sim_t P$ (nice!)
- $a.b.0 + a.c.0 \sim_t a.(b.0 + c.0)$ (not that nice:)
- $a.b.0 + a.0 \sim_t a.b.0$ (not sensitive to deadlocks)
- $a.0 + b.0 \sim_t (\nu c)(c.0 \parallel \overline{c}.a.0 \parallel \overline{c}.b.0)$

Trace Equivalence

Definition (Trace Equivalence)

P and Q are ***trace equivalent***, written $P \sim_t Q$, iff for every experiment (here called trace) $e = l_1 \ldots l_n \in \mathcal{L}^*$

$$P \stackrel{e}{\Longrightarrow} \quad \text{iff} \quad Q \stackrel{e}{\Longrightarrow}$$

Observations:

- $\tau.P \ \sim_t \ P$ (nice!)
- $a.b.0 + a.c.0 \ \sim_t \ a.(b.0 + c.0)$ (not that nice:)
- $a.b.0 + a.0 \ \sim_t \ a.b.0$ (not sensitive to deadlocks)
- $a.0 + b.0 \ \sim_t \ (\nu c)(c.0 \parallel \overline{c}.a.0 \parallel \overline{c}.b.0)$

Failures Equivalence

Definition (Failures Equivalence)

A pair (e, L), where $e \in \mathcal{L}^*$ (a *trace*) and $L \subset \mathcal{L}$, is a **failure** for P iff there is P' s.t.:

$$(1) P \stackrel{e}{\Longrightarrow} P', \ (2)\, P' \not\stackrel{l}{\longrightarrow} \text{ for all } l \in L, \text{ and } (3) P' \not\stackrel{\tau}{\longrightarrow}$$

$P \sim_f Q$ iff P and Q have the same failures.

Observations:

- $\tau.P \ \sim_f \ P$.
- $a.b.0 + a.c.0 \ \not\sim_f \ a.(b.0 + c.0)$ (**Exercise**: why?)
- $a.b.0 + a.0 \ \not\sim_f \ a.b.0$.
- $a.0 + b.0 \ \not\sim_f \ (\nu c)(c.0 \parallel \overline{c}.a.0 \parallel \overline{c}.b.0)$ (**Exercise**)
- $a.(b.c.0 + b.d.0) \ \sim_f \ a.b.c.0 + a.b.d.0$.
- Let $P = \tau.P$. We have $\tau.0 \not\sim_f P$.

Failures Equivalence

Definition (Failures Equivalence)

A pair (e, L), where $e \in \mathcal{L}^*$ (a *trace*) and $L \subset \mathcal{L}$, is a **failure** for P iff there is P' s.t.:

$$(1) P \stackrel{e}{\Longrightarrow} P', \ (2)\, P' \not\stackrel{l}{\longrightarrow} \text{ for all } l \in L, \text{ and } (3) P' \not\stackrel{\tau}{\longrightarrow}$$

$P \sim_f Q$ iff P and Q have the same failures.

Observations:

- $\tau.P \ \sim_f \ P$.
- $a.b.0 + a.c.0 \ \not\sim_f \ a.(b.0 + c.0)$ (**Exercise**: why?)
- $a.b.0 + a.0 \ \not\sim_f \ a.b.0$.
- $a.0 + b.0 \ \not\sim_f \ (\nu c)(c.0 \parallel \overline{c}.a.0 \parallel \overline{c}.b.0)$ (**Exercise**)
- $a.(b.c.0 + b.d.0) \ \sim_f \ a.b.c.0 + a.b.d.0$.
- Let $D = \tau.D$. We have $\tau.0 \not\sim_f D$

Weak Bisimilarity (Observational Equivalence)

Definition (Weak Bisimilarity)

A ***symmetric*** relation R is a ***weak bisimulation*** iff for every $(P, Q) \in R$:

▷ If $P \stackrel{e}{\Longrightarrow} P'$ then there is Q' s.t. $Q \stackrel{e}{\Longrightarrow} Q'$ and $(P', Q') \in R$.

Define $P \approx Q$ iff $(P, Q) \in R$ for some *weak bisimulation* R.

Observations:

- $\tau.P \approx P$, $a.\tau.P \approx a.P$
- However, $a.0 + b.0 \not\approx a.0 + \tau.b.0$
- $a.(b.c.0 + b.d.0) \not\approx a.b.c.0 + a.b.d.0$.(**Exercise**)
- Let $D = \tau.D$. We have $\tau.0 \approx D$.
- $a.0 + b.0 \not\approx (\nu c)(c.0 \parallel \overline{c}.a.0 \parallel \overline{c}.b.0)$ (**Exercise**)

Weak Bisimilarity (Observational Equivalence)

Definition (Weak Bisimilarity)

A ***symmetric*** relation R is a ***weak bisimulation*** iff for every $(P, Q) \in R$:

$\triangleright$ If $P \stackrel{e}{\Longrightarrow} P'$ then there is Q' s.t. $Q \stackrel{e}{\Longrightarrow} Q'$ and $(P', Q') \in R$.

Define $P \approx Q$ iff $(P, Q) \in R$ for some *weak bisimulation* R.

Observations:

- $\tau.P \approx P$, $a.\tau.P \approx a.P$
- However, $a.0 + b.0 \not\approx a.0 + \tau.b.0$
- $a.(b.c.0 + b.d.0) \not\approx a.b.c.0 + a.b.d.0$.(**Exercise**)
- Let $D = \tau.D$. We have $\tau.0 \approx D$.
- $a.0 + b.0 \not\approx (\nu c)(c.0 \parallel \overline{c}.a.0 \parallel \overline{c}.b.0)$ (**Exercise**)

Process Logic: Verification and Specification

- Processes can be used to *specify and verify* behavioural properties:
  E.g., $2p.\overline{tea}.0 + 2p.\overline{coffe}.0$ specifies a machine which does not satisfy the behaviour specified by a $2p.(\overline{tea}.0 + \overline{coffe}.0)$.
- Alternatively, a logic can be used. An example of a property is "P will never not execute a bad action" or "P eventually executes execute a good action".

Process Logic: Verification and Specification

- Processes can be used to *specify and verify* behavioural properties:
  E.g., $2p.\overline{tea}.0 + 2p.\overline{coffe}.0$ specifies a machine which does not satisfy the behaviour specified by a $2p.(\overline{tea}.0 + \overline{coffe}.0)$.
- Alternatively, a logic can be used. An example of a property is "P will never not execute a bad action" or "P eventually executes execute a good action".

Outline

Hennessy&Milner Logic

Definition (Logic Syntax)

$$F \ := \texttt{true} \mid \texttt{false} \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \langle K \rangle F \mid [K]F$$

where K is a set of actions

- The boolean operators are interpreted as in propositional logic.
- $\langle K \rangle F$ (**possibility**) asserts (of a given P): It is *possible* for P to do a $\mathbf{a} \in K$ and then evolve into a Q that satisfy F.
- $[K]F$ (**necessity**) asserts (of a given P): If P can do a $\mathbf{a} \in K$ then it *must* evolve into a Q which satisfies F.

Hennessy&Milner Logic

Definition (Logic Syntax)

$$F \ := \texttt{true} \mid \texttt{false} \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \langle K \rangle F \mid [K]F$$

where K is a set of actions

- The boolean operators are interpreted as in propositional logic.
- $\langle K \rangle F$ (**possibility**) asserts (of a given P): It is *possible* for P to do a $\mathbf{a} \in K$ and then evolve into a Q that satisfy F.
- $[K]F$ (**necessity**) asserts (of a given P): If P can do a $\mathbf{a} \in K$ then it *must* evolve into a Q which satisfies F.

Hennessy&Milner Logic

Definition (Logic Syntax)

$$F \ := \texttt{true} \mid \texttt{false} \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \langle K \rangle F \mid [K]F$$

where K is a set of actions

- The boolean operators are interpreted as in propositional logic.
- $\langle K \rangle F$ (**possibility**) asserts (of a given P): It is *possible* for P to do a $\mathbf{a} \in K$ and then evolve into a Q that satisfy F.
- $[K]F$ (**necessity**) asserts (of a given P): If P can do a $\mathbf{a} \in K$ then it *must* evolve into a Q which satisfies F.

Hennessy&Milner Logic

Definition (Logic Syntax)

$$F \ := \texttt{true} \mid \texttt{false} \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \langle K \rangle F \mid [K]F$$

where K is a set of actions

- The boolean operators are interpreted as in propositional logic.
- $\langle K \rangle F$ (**possibility**) asserts (of a given P): It is *possible* for P to do a $\mathbf{a} \in K$ and then evolve into a Q that satisfy F.
- $[K]F$ (**necessity**) asserts (of a given P): If P can do a $\mathbf{a} \in K$ then it *must* evolve into a Q which satisfies F.

Semantics of Hennessy-Milner Logic

Definition

The compliance of P with the specification F, written $P \models F$, is given by:

$$
\begin{array}{lll}
P \not\models \texttt{false} & & \\
P \models \texttt{true} & & \\
P \models F_1 \wedge F_2 & \text{iff} & P \models F_1 \text{ and } P \models F_2 \\
P \models F_1 \vee F_2 & \text{iff} & P \models F_1 \text{ or } P \models F_2 \\
P \models \langle K \rangle F & \text{iff} & \text{for some } Q,\ P \xrightarrow{\mathbf{a}} Q,\ \mathbf{a} \in K \text{ and } Q \models F \\
P \models [K]F & \text{iff} & \text{if } P \xrightarrow{\mathbf{a}} Q \text{ and } \mathbf{a} \in K \text{ then } Q \models F
\end{array}
$$

Hennessy&Milner Logic and Bisimilarity

Example

Let $P_1 = a.(b.0 + c.0)$, $P_2 = a.b.0 + a.c.0$. Also let

$$F = \langle\{a\}\rangle\,(\langle\{b\}\rangle\,\mathtt{true} \wedge \langle\{c\}\rangle\,\mathtt{true}).$$

Notice that $P_1 \models F$ but $P_2 \not\models F$.

Theorem

$P \sim Q$ if and only, for every F, $P \models F$ iff $Q \models F$.

Hennessy&Milner Logic and Bisimilarity

Example

Let $P_1 = a.(b.0 + c.0)$, $P_2 = a.b.0 + a.c.0$. Also let

$$F = \langle\{a\}\rangle\,(\langle\{b\}\rangle\,\mathtt{true} \wedge \langle\{c\}\rangle\,\mathtt{true}).$$

Notice that $P_1 \models F$ but $P_2 \not\models F$.

Theorem

$P \sim Q$ if and only, for every F, $P \models F$ iff $Q \models F$.

A Linear Temporal Logic.

Definition (Logic Syntax)

$$F := \texttt{true} \mid \texttt{false} \mid L \mid F_1 \vee F_2 \mid F_1 \wedge F_2 \mid \Diamond F \mid \Box F$$

where L is a set of non-silent actions.

- Formulae assert properties of traces.
- Boolean operators are interpreted as usual
- L asserts (of a given trace s) that the first action of s must be in $L \cup \{\tau\}$
- $\Diamond F$ asserts (of a given trace s) that at some point in s, F holds.
- $\Box F$ asserts (of a given trace s) that at every point in s, F holds.

A Linear Temporal Logic.

Definition (Logic Syntax)

$$F := \texttt{true} \mid \texttt{false} \mid L \mid F_1 \vee F_2 \mid F_1 \wedge F_2 \mid \Diamond F \mid \Box F$$

where L is a set of non-silent actions.

- Formulae assert properties of traces.
- Boolean operators are interpreted as usual
- L asserts (of a given trace s) that the first action of s must be in $L \cup \{\tau\}$
- $\Diamond F$ asserts (of a given trace s) that at some point in s, F holds.
- $\Box F$ asserts (of a given trace s) that at every point in s, F holds.

A Linear Temporal Logic.

Definition (Logic Syntax)

$$F := \texttt{true} \mid \texttt{false} \mid L \mid F_1 \vee F_2 \mid F_1 \wedge F_2 \mid \Diamond F \mid \Box F$$

where L is a set of non-silent actions.

- Formulae assert properties of traces.
- Boolean operators are interpreted as usual
- L asserts (of a given trace s) that the first action of s must be in $L \cup \{\tau\}$
- $\Diamond F$ asserts (of a given trace s) that at some point in s, F holds.
- $\Box F$ asserts (of a given trace s) that at every point in s, F holds.

A Linear Temporal Logic.

Definition (Logic Syntax)

$$F := \texttt{true} \mid \texttt{false} \mid L \mid F_1 \vee F_2 \mid F_1 \wedge F_2 \mid \Diamond F \mid \Box F$$

where L is a set of non-silent actions.

- Formulae assert properties of traces.
- Boolean operators are interpreted as usual
- L asserts (of a given trace s) that the first action of s must be in $L \cup \{\tau\}$
- $\Diamond F$ asserts (of a given trace s) that at some point in s, F holds.
- $\Box F$ asserts (of a given trace s) that at every point in s, F holds.

A Linear Temporal Logic.

Definition (Logic Syntax)

$$F := \texttt{true} \mid \texttt{false} \mid L \mid F_1 \vee F_2 \mid F_1 \wedge F_2 \mid \Diamond F \mid \Box F$$

where L is a set of non-silent actions.

- Formulae assert properties of traces.
- Boolean operators are interpreted as usual
- L asserts (of a given trace s) that the first action of s must be in $L \cup \{\tau\}$
- $\Diamond F$ asserts (of a given trace s) that at some point in s, F holds.
- $\Box F$ asserts (of a given trace s) that at every point in s, F holds.

Semantics of the temporal logic.

Definition

An infinite sequence of actions $s = \mathbf{a}_1.\mathbf{a}_2 \ldots$ satisfies (or is a model of) F, written $s \models F$, iff $\langle s, 1\rangle \models F$, where

$$
\begin{array}{lll}
\langle s, i\rangle \models \texttt{true} & & \\
\langle s, i\rangle \not\models \texttt{false} & & \\
\langle s, i\rangle \models L & \text{iff} & \mathbf{a}_i \in L \cup \tau \\
\langle s, i\rangle \models F_1 \vee F_2 & \text{iff} & \langle s, i\rangle \models F_1 \text{ or } \langle s, i\rangle \models F_2 \\
\langle s, i\rangle \models F_1 \wedge F_2 & \text{iff} & \langle s, i\rangle \models F_1 \text{ and } \langle s, i\rangle \models F_2 \\
\langle s, i\rangle \models \Box F & \text{iff} & \text{for all } j \geq i \;\; \langle s, j\rangle \models F \\
\langle s, i\rangle \models \Diamond F & \text{iff} & \text{there is a } j \geq i \text{ s.t. } \langle s, j\rangle \models F
\end{array}
$$

$P \models F$ iff whenever $P \stackrel{s}{\Longrightarrow}$ then $\hat{s} \models F$, where $\hat{s} = s.\tau.\tau.\ldots$

Temporal Logic and Trace Equivalence

Example

Consider the trace equivalent processes
$A(a,b,c) \stackrel{\mathtt{def}}{=} a.(b.A(a,b,c) + c.A(a,b,c))$ and
$B(a,b,c) \stackrel{\mathtt{def}}{=} a.b.B(a,b,c) + a.c.B(a,b,c)$, and the property
$F = \Box\Diamond(b \vee c)$.
We have $A(a,b,c) \models F$ and $B(a,b,c) \models F$.

Theorem

$P \sim_t Q$ then for every linear temporal formula F,

$$P \models F \ \mathit{iff}\ Q \models F$$

Question. Does the other direction of the theorem hold ?

Temporal Logic and Trace Equivalence

Example

Consider the trace equivalent processes
$A(a,b,c) \stackrel{\mathtt{def}}{=} a.(b.A(a,b,c) + c.A(a,b,c))$ and
$B(a,b,c) \stackrel{\mathtt{def}}{=} a.b.B(a,b,c) + a.c.B(a,b,c)$, and the property
$F = \Box\Diamond(b \vee c)$.
We have $A(a,b,c) \models F$ and $B(a,b,c) \models F$.

Theorem

$P \sim_t Q$ then for every linear temporal formula F,

$$P \models F \ \mathit{iff}\ Q \models F$$

Question. Does the other direction of the theorem hold ?

Temporal Logic and Trace Equivalence

Example

Consider the trace equivalent processes
$A(a,b,c) \stackrel{\mathtt{def}}{=} a.(b.A(a,b,c) + c.A(a,b,c))$ and
$B(a,b,c) \stackrel{\mathtt{def}}{=} a.b.B(a,b,c) + a.c.B(a,b,c)$, and the property
$F = \Box\Diamond(b \vee c)$.
We have $A(a,b,c) \models F$ and $B(a,b,c) \models F$.

Theorem

$P \sim_t Q$ then for every linear temporal formula F,

$$P \models F \ \mathit{iff}\ Q \models F$$

Question. Does the other direction of the theorem hold ?

Outline

Mobility and the pi calculus

What kind of ***mobility*** are we talking about?

- Processes move.
- **Links** move$\surd$

The last one, because it is the π-calculus' choice; for flexibility and simplicity.

- π-calculus has the ability of sending private and public *links* (*names*).

Mobility and the pi calculus

What kind of ***mobility*** are we talking about?

- Processes move.
- **Links** move$\surd$

The last one, because it is the π-calculus' choice; for flexibility and simplicity.

- π-calculus has the ability of sending private and public *links* (*names*).

Mobility and the pi calculus

What kind of ***mobility*** are we talking about?

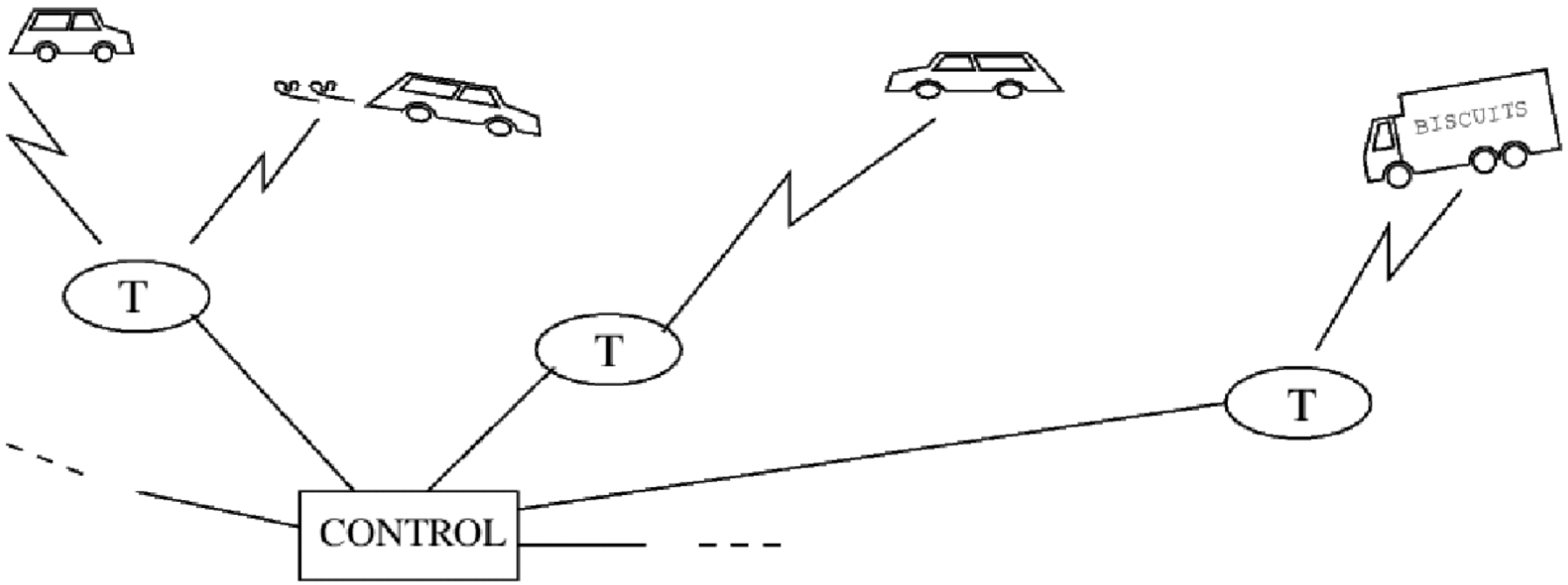
- Processes move.
- ***Links*** move$\surd$

The last one, because it is the π-calculus' choice; for flexibility and simplicity.

- π-calculus has the ability of sending private and public *links* (*names*).

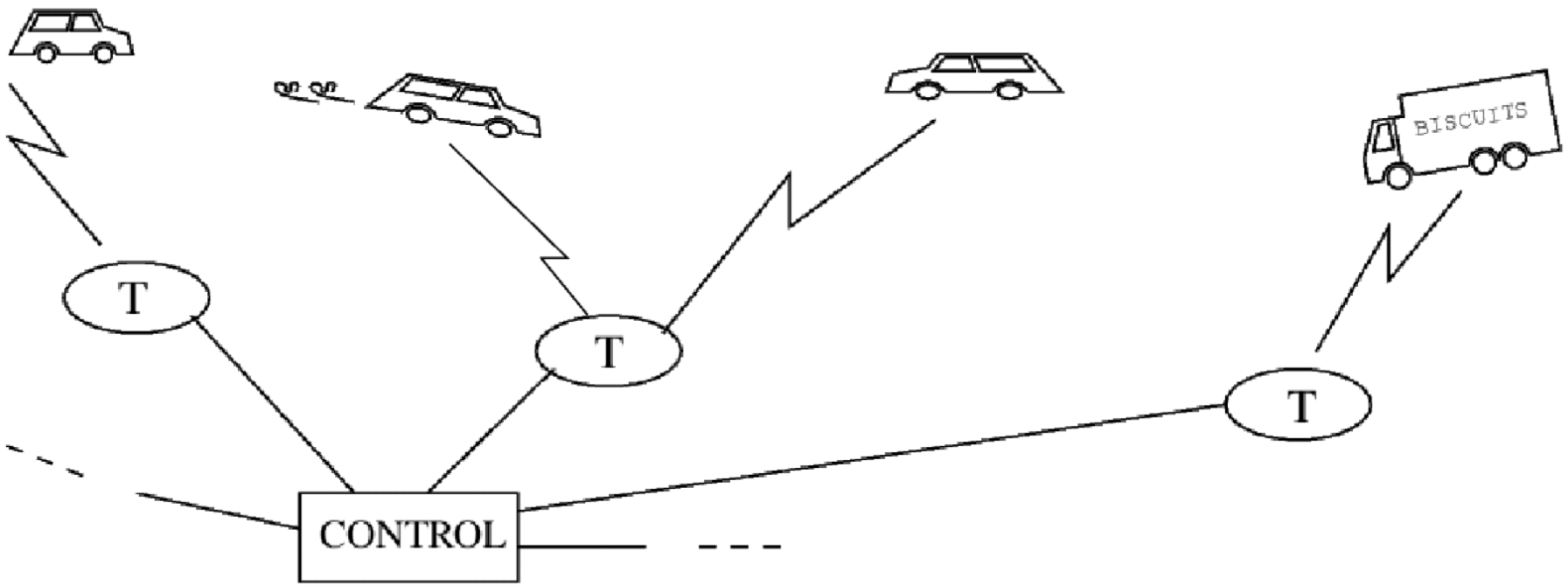
Link Mobility

Mobile phones



Link Mobility

Mobile phones



$\pi-$Calculus: Syntax

Definition (Syntax)

$$P, Q, \ldots := P \parallel Q \mid \Sigma_{i \in I}\, \alpha.P \mid (\nu a)P \mid !P \mid \textbf{if } a = b \textbf{ then } P$$

where $\alpha := \tau \mid \overline{a}b \mid a(x)$.

- Names=Channels=Ports=Links.
- $\overline{a}b.P$: "**send** b on channel a and then activate P"
- $a(x).P$: "**receive** a name on channel a (if any), and replace x with it in P"
- $(\nu a)P$: "**create** a **fresh name** a **private** to P"
- $(\nu a)P$ and $a(x).Q$ are the only binders.
- $!P$: "replicate P" i.e., $!P$ represents $P \parallel P \parallel P \parallel \ldots$

$\pi-$Calculus: Syntax

Definition (Syntax)

$$P, Q, \ldots := P \parallel Q \mid \Sigma_{i \in I}\, \alpha.P \mid (\nu a)P \mid !P \mid \textbf{if } a = b \textbf{ then } P$$

where $\alpha := \tau \mid \overline{a}b \mid a(x)$.

- Names=Channels=Ports=Links.
- $\overline{a}b.P$: "**send** b on channel a and then activate P"
- $a(x).P$: "**receive** a name on channel a (if any), and replace x with it in P"
- $(\nu a)P$: "**create** a **fresh name** a **private** to P"
- $(\nu a)P$ and $a(x).Q$ are the only binders.
- $!P$: "replicate P" i.e., $!P$ represents $P \parallel P \parallel P \parallel \ldots$

$\pi-$Calculus: Syntax

Definition (Syntax)

$$P, Q, \ldots := P \parallel Q \mid \Sigma_{i \in I}\, \alpha.P \mid (\nu a)P \mid !P \mid \textbf{if } a = b \textbf{ then } P$$

where $\alpha := \tau \mid \bar{a}b \mid a(x)$.

- Names=Channels=Ports=Links.
- $\bar{a}b.P$: "**send** b on channel a and then activate P"
- $a(x).P$: "**receive** a name on channel a (if any), and replace x with it in P"
- $(\nu a)P$: "**create** a **fresh name** a **private** to P"
- $(\nu a)P$ and $a(x).Q$ are the only binders.
- $!P$: "replicate P" i.e., $!P$ represents $P \parallel P \parallel P \parallel \ldots$

$\pi-$Calculus: Syntax

Definition (Syntax)

$$P, Q, \ldots := P \parallel Q \mid \Sigma_{i \in I}\, \alpha.P \mid (\nu a)P \mid !P \mid \textbf{if } a = b \textbf{ then } P$$

where $\alpha := \tau \mid \bar{a}b \mid a(x)$.

- Names=Channels=Ports=Links.
- $\bar{a}b.P$: "send b on channel a and then activate P"
- $a(x).P$: "receive a name on channel a (if any), and replace x with it in P"
- $(\nu a)P$: "create a fresh name a private to P"
- $(\nu a)P$ and $a(x).Q$ are the only binders.
- $!P$: "replicate P" i.e., $!P$ represents $P \parallel P \parallel P \parallel \ldots$

$\pi-$Calculus: Syntax

Definition (Syntax)

$$P, Q, \ldots := P \parallel Q \mid \Sigma_{i \in I}\, \alpha.P \mid (\nu a)P \mid !P \mid \textbf{if } a = b \textbf{ then } P$$

where $\alpha := \tau \mid \overline{a}b \mid a(x)$.

- Names=Channels=Ports=Links.
- $\overline{a}b.P$: "**send** b on channel a and then activate P"
- $a(x).P$: "**receive** a name on channel a (if any), and replace x with it in P"
- $(\nu a)P$: "**create** a **fresh name** a **private** to P"
- $(\nu a)P$ and $a(x).Q$ are the only binders.
- $!P$: "replicate P" i.e., $!P$ represents $P \parallel P \parallel P \parallel \ldots$

$\pi-$Calculus: Syntax

Definition (Syntax)

$$P, Q, \ldots := P \parallel Q \mid \Sigma_{i \in I}\, \alpha.P \mid (\nu a)P \mid !P \mid \textbf{if } a = b \textbf{ then } P$$

where $\alpha := \tau \mid \overline{a}b \mid a(x)$.

- Names=Channels=Ports=Links.
- $\overline{a}b.P$: "**send** b on channel a and then activate P"
- $a(x).P$: "**receive** a name on channel a (if any), and replace x with it in P"
- $(\nu a)P$: "**create** a **fresh name** a **private** to P"
- $(\nu a)P$ and $a(x).Q$ are the only binders.
- $!P$: "replicate P" i.e., $!P$ represents $P \parallel P \parallel P \parallel \ldots$

$\pi-$Calculus: Syntax

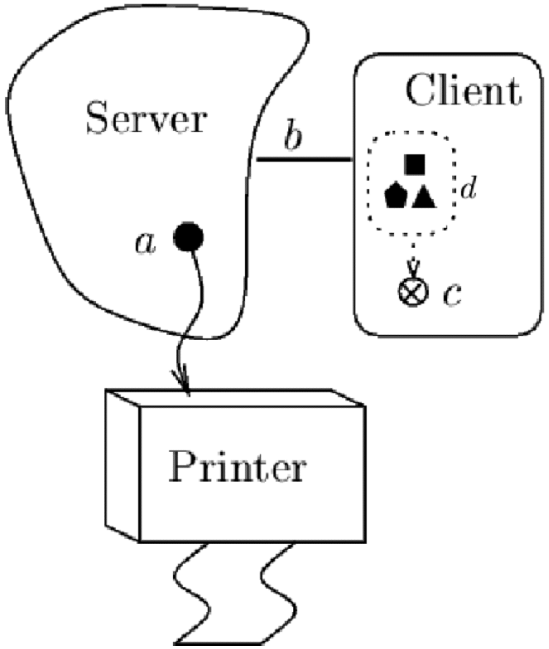
Definition (Syntax)

$$P, Q, \ldots := P \parallel Q \mid \Sigma_{i \in I}\, \alpha.P \mid (\nu a)P \mid !P \mid \textbf{if } a = b \textbf{ then } P$$

where $\alpha := \tau \mid \overline{a}b \mid a(x)$.

- Names=Channels=Ports=Links.
- $\overline{a}b.P$: "**send** b on channel a and then activate P"
- $a(x).P$: "**receive** a name on channel a (if any), and replace x with it in P"
- $(\nu a)P$: "**create** a **fresh name** a **private** to P"
- $(\nu a)P$ and $a(x).Q$ are the only binders.
- $!P$: "replicate P" i.e., $!P$ represents $P \parallel P \parallel P \parallel \ldots$

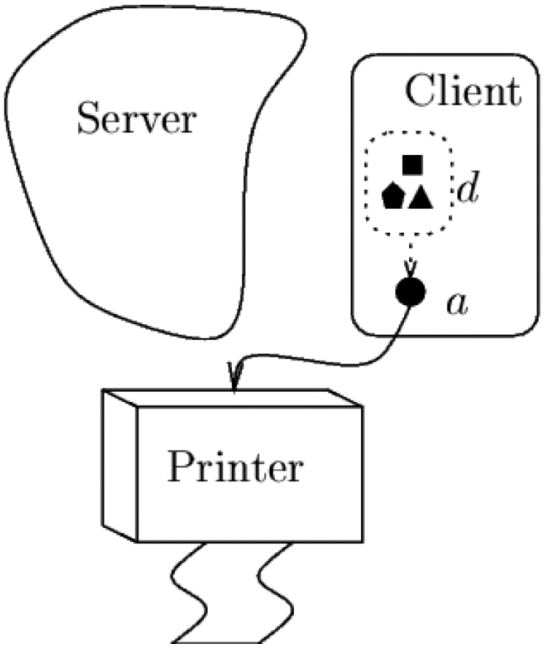
A flavour of the pi-calculus



In the π calculus:

$(\nu a)(\bar{b}a.S \parallel a(x).P) \parallel b(c).\bar{c}d.C$

A flavour of the pi-calculus



In π the pi-calculus:

$$(\nu a)(\bar{b}a.S \parallel a(x).P) \parallel b(c).\bar{c}d.C \longrightarrow S \parallel (\nu a)(a(x).P \parallel \bar{a}d.C)$$

Mobile Exercises

Write an agent that:

- reads something from port a and sends it twice along port b
- generates ***infinitely many*** different names—and send them along channel a.
- encodes biadic-comunication: $\bar{x}z_1z_2.P$ which outputs both y_1 and y_2 on channel x, and $x(y_1, y_2).Q$ which inputs two names sent on channel x.

Mobile Exercises

Write an agent that:

- reads something from port a and sends it twice along port b
- generates ***infinitely many*** different names—and send them along channel a.
- encodes biadic-comunication: $\bar{x}z_1z_2.P$ which outputs both y_1 and y_2 on channel x, and $x(y_1, y_2).Q$ which inputs two names sent on channel x.

Mobile Exercises

Write an agent that:

- reads something from port a and sends it twice along port b
- generates ***infinitely many*** different names—and send them along channel a.
- encodes biadic-comunication: $\bar{x}z_1z_2.P$ which outputs both y_1 and y_2 on channel x, and $x(y_1, y_2).Q$ which inputs two names sent on channel x.

Reaction Semantics of π.

The reactive semantics of π consists of an ***structural congruence*** $\equiv$ and the ***reactive rules***.

- The relation $\equiv$ describes **irrelevant syntactic aspects**
- The reactive semantics describes evolutions due to **synchronous** communication.

Reaction Semantics of π.

The reactive semantics of π consists of an ***structural congruence*** $\equiv$ and the ***reactive rules***.

- The relation$\equiv$ describes **irrelevant syntactic aspects**
- The reactive semantics describes evolutions due to **synchronous** communication.

Reaction Semantics of π.

The reactive semantics of π consists of an ***structural congruence*** $\equiv$ and the ***reactive rules***.

- The relation $\equiv$ describes **irrelevant syntactic aspects**
- The reactive semantics describes evolutions due to **synchronous** communication.

Structural Equivalence

Structural Congruence

Structural Congruence $\equiv$ is the congruence given by:

- $P \equiv Q$ if P can be α-*converted* into Q.
- $P \parallel 0 \equiv P$, $P \parallel Q \equiv Q \parallel P$, $(P \parallel Q) \parallel R = P \parallel (Q \parallel R)$.
- $(\nu a)0 = 0$,$(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P$,
- $(\nu a)(P \parallel Q) \equiv P \parallel (\nu a)Q$ if $a \notin fn(P)$.
- $!P \equiv P \parallel !P$.

Example

(1) $(\nu a)P \equiv P$ if $a \notin fn(P)$?
(2) $x(y).y(z) \equiv x(z).y(y)$?
(3) $x \parallel y \equiv x.y + y.x$?

Structural Equivalence

Structural Congruence

Structural Congruence $\equiv$ is the congruence given by:

- $P \equiv Q$ if P can be α-*converted* into Q.
- $P \parallel 0 \equiv P$, $P \parallel Q \equiv Q \parallel P$, $(P \parallel Q) \parallel R = P \parallel (Q \parallel R)$.
- $(\nu a)0 = 0$,$(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P$,
- $(\nu a)(P \parallel Q) \equiv P \parallel (\nu a)Q$ if $a \notin fn(P)$.
- $!P \equiv P \parallel !P$.

Example

(1) $(\nu a)P \equiv P$ if $a \notin fn(P)$?
(2) $x(y).y(z) \equiv x(z).y(y)$?
(3) $x \parallel y \equiv x.y + y.x$?

Reactive Rules

$$\text{TAU}\quad \frac{}{\tau.P + M \longrightarrow P}$$

$$\text{REACT}\quad \frac{}{(a(x).P + M) \;\parallel\; (\bar{a}b.Q + N) \longrightarrow P[b/x] \parallel Q}$$

$$\text{PAR}\quad \frac{P \longrightarrow P'}{P \parallel Q \longrightarrow P' \parallel P}$$

$$\text{RES}\quad \frac{P \longrightarrow P'}{(\nu a)P \longrightarrow (\nu a)P'}$$

$$\text{STRUCT}\quad \frac{P \equiv P' \longrightarrow Q' \equiv Q}{P \longrightarrow Q}$$

Exercise. Assume $a \notin fn(S) \cup fn(C)$. Write a derivation for our Server example:

$$(\nu a)(\bar{b}a.S \parallel a(x).P) \parallel b(c).\bar{c}d.C \longrightarrow S \parallel (\nu a)(a(x).P \parallel \bar{a}d.C)$$

Outline

Applications: Persistent and Mutable Data Structures

Definition

A ***cell*** is basis for defining mutable and persistent data structures.

$$Cell(u, r, z) \stackrel{\mathtt{def}}{=} u(z').Cell(u, r, z') + r(x).\bar{x}z.Cell(u, r, z)$$

The cell current contents z can be updated via u or read via r.

But we do not have recursive definitions in the π-calculus. Can the above cell be rewritten in with the π-calculus primitives ?

Applications: Persistent and Mutable Data Structures

Definition

A ***cell*** is basis for defining mutable and persistent data structures.

$$Cell(u, r, z) \stackrel{\mathtt{def}}{=} u(z').Cell(u, r, z') + r(x).\bar{x}z.Cell(u, r, z)$$

The cell current contents z can be updated via u or read via r.

But we do not have recursive definitions in the π-calculus. Can the above cell be rewritten in with the π-calculus primitives ?

Recursive Definitions

We wish invocations $A(z)$ to recursive definition (as in CCS) $A(x) \stackrel{\mathtt{def}}{=} P$. Do we need to extend the calculus ?

No! Jus take an $a \notin fn(P)$:

1. Replace in P every invocation $A(u)$ with $\bar{a}u$. Call $\hat{P}$ the resulting processes.
1. Encode $A(x) = P$ as $!a(x).\hat{P}$.
1. Encode $A(z)$ as $(\nu a)(\bar{a}(z) \parallel !a(x).\hat{P})$.

Recursive Definitions

We wish invocations $A(z)$ to recursive definition (as in CCS) $A(x) \stackrel{\mathtt{def}}{=} P$. Do we need to extend the calculus ?
No! Jus take an $a \notin fn(P)$:

1. Replace in P every invocation $A(u)$ with $\bar{a}u$. Call $\hat{P}$ the resulting processes.
1. Encode $A(x) = P$ as $!a(x).\hat{P}$.
1. Encode $A(z)$ as $(\nu a)(\bar{a}(z) \parallel !a(x).\hat{P})$.

Recursive Definitions

We wish invocations $A(z)$ to recursive definition (as in CCS) $A(x) \stackrel{\mathtt{def}}{=} P$. Do we need to extend the calculus ?
No! Jus take an $a \notin fn(P)$:

1. Replace in P every invocation $A(u)$ with $\bar{a}u$. Call $\hat{P}$ the resulting processes.
1. Encode $A(x) = P$ as $!a(x).\hat{P}$.
1. Encode $A(z)$ as $(\nu a)(\bar{a}(z) \parallel !a(x).\hat{P})$.

Recursive Definitions

We wish invocations $A(z)$ to recursive definition (as in CCS) $A(x) \stackrel{\mathtt{def}}{=} P$. Do we need to extend the calculus ?
No! Jus take an $a \notin fn(P)$:

1. Replace in P every invocation $A(u)$ with $\bar{a}u$. Call $\hat{P}$ the resulting processes.
1. Encode $A(x) = P$ as $!a(x).\hat{P}$.
1. Encode $A(z)$ as $(\nu a)(\bar{a}(z) \parallel !a(x).\hat{P})$.

π Equivalences

Should $P = x(z) \parallel \bar{y}u$ be equivalent to $Q = x(z).\bar{y}u + \bar{y}u.x(z)$ in π—like in CCS?

- Take $R = a(x).P$ and $S = a(x).Q$.
- Hence, $\bar{a}y$ can tell R and S apart!. I.e., $\bar{a}y \parallel R$.

Passing links around complicates matters—not unduly, though:).
Need to consider substitutions, contexts, etc!

π Equivalences

Should $P = x(z) \parallel \bar{y}u$ be equivalent to $Q = x(z).\bar{y}u + \bar{y}u.x(z)$ in π—like in CCS?

- Take $R = a(x).P$ and $S = a(x).Q$.
- Hence, $\bar{a}y$ can tell R and S apart!. I.e., $\bar{a}y \parallel R$.

Passing links around complicates matters—not unduly, though:).
Need to consider substitutions, contexts, etc!

π Equivalences

Should $P = x(z) \parallel \bar{y}u$ be equivalent to $Q = x(z).\bar{y}u + \bar{y}u.x(z)$ in π—like in CCS?

- Take $R = a(x).P$ and $S = a(x).Q$.
- Hence, $\bar{a}y$ can tell R and S apart!. I.e., $\bar{a}y \parallel R$.

Passing links around complicates matters—not unduly, though:). Need to consider substitutions, contexts, etc!

Outline

Barbed Equivalence

Intuitively we write $P \downarrow_x$ ($P \downarrow_{\bar{x}}$) iff P can receive (send) a link on channel x.
Formally, $P \downarrow_l$ where $l \in \mathcal{L}$ is given by:

- $x(y).P \downarrow_x$.
- $\bar{x}z.P \downarrow_{\bar{x}}$.
- $\Sigma_{i \in I} \alpha_i.P_i \downarrow_l$ iff for some $i \in I$, $\alpha_i.P_i \downarrow_l$
- $P \parallel Q \downarrow_l$ iff either $P \downarrow_l$ or $Q \downarrow_l$
- $(\nu z)P \downarrow_l$ iff $l \neq z$, $\bar{l} \neq z$ and $P \downarrow_l$

Barbed Equivalence

Definition

R is a ***barbed simulation*** iff for every $(P, Q) \in R$:
(1) If $P \longrightarrow P'$ then $\exists Q'$: $Q \longrightarrow Q' \ \wedge \ (P', Q') \in R$.
(2) If $P \downarrow_l$ then $Q \downarrow_l$.

Definition

Barbed bisimilarity: $P \sim_B Q$ iff there is R such that R and R^{-1} are barbed simulations and $(P, Q) \in R$.

Example

Let $P = x(z) \parallel \bar{y}u$ and $Q = x(z).\bar{y}u + \bar{y}u.x(z)$. Then $P \sim_B Q$ and$a(x).P \ \sim_B \ a(x).Q$.

Barbed Equivalence

Definition

R is a ***barbed simulation*** iff for every $(P, Q) \in R$:
(1) If $P \longrightarrow P'$ then $\exists Q'$: $Q \longrightarrow Q' \ \wedge \ (P', Q') \in R$.
(2) If $P \downarrow_l$ then $Q \downarrow_l$.

Definition

Barbed bisimilarity: $P \sim_B Q$ iff there is R such that R and R^{-1} are barbed simulations and $(P, Q) \in R$.

Example

Let $P = x(z) \parallel \bar{y}u$ and $Q = x(z).\bar{y}u + \bar{y}u.x(z)$. Then $P \sim_B Q$ and$a(x).P \ \sim_B \ a(x).Q$.

Barbed Equivalence

Definition

R is a ***barbed simulation*** iff for every $(P, Q) \in R$:
(1) If $P \longrightarrow P'$ then $\exists Q'$: $Q \longrightarrow Q' \ \wedge \ (P', Q') \in R$.
(2) If $P \downarrow_l$ then $Q \downarrow_l$.

Definition

Barbed bisimilarity: $P \sim_B Q$ iff there is R such that R and R^{-1} are barbed simulations and $(P, Q) \in R$.

Example

Let $P = x(z) \parallel \bar{y}u$ and $Q = x(z).\bar{y}u + \bar{y}u.x(z)$. Then $P \sim_B Q$ and$a(x).P \ \sim_B \ a(x).Q$.

Barbed Congruence.

Let $\approx_B$ be the congruence *induced* by $\sim_B$. I.e:

Definition

$$P \approx_B Q \text{ iff } \text{ for every context } C[.],\ C[P] \sim_B C[Q]$$

Example

Note that $x(z) \parallel \bar{y}u \not\approx_B x(z).\bar{y}u + \bar{y}u.x(z)$.

Barbed Congruence.

Let $\approx_B$ be the congruence *induced* by $\sim_B$. I.e:

Definition

$$P \approx_B Q \text{ iff } \text{ for every context } C[.],\ C[P] \sim_B C[Q]$$

Example

Note that $x(z) \parallel \bar{y}u \not\approx_B x(z).\bar{y}u + \bar{y}u.x(z)$.

A Labelled Semantics for Pi.

Transitions $P \xrightarrow{\alpha} Q$ where $\alpha \in \{\bar{a}b, a(x), \tau, \bar{a}\,\nu u\}$ are given by:

$$\text{PREFIX}\quad \frac{}{\alpha\,.\,P + M \xrightarrow{\alpha} P}$$

$$\text{PAR}\quad \frac{P \xrightarrow{\alpha} P',\ \mathtt{bn}(\alpha) \cap \mathtt{fn}(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P'|Q}$$

$$\text{COM}\quad \frac{P \xrightarrow{a(x)} P',\ Q \xrightarrow{\bar{a}u} Q'}{P|Q \xrightarrow{\tau} P'\{u/x\}|Q'}$$

$$\text{RES}\quad \frac{P \xrightarrow{\alpha} P',\ x \notin \alpha}{(\boldsymbol{\nu} x)P \xrightarrow{\alpha} (\boldsymbol{\nu} x)P'}$$

$$\text{OPEN}\quad \frac{P \xrightarrow{\bar{a}x} P',\ a \neq x}{(\boldsymbol{\nu} x)P \xrightarrow{\bar{a}\boldsymbol{\nu} x} P'}$$

$$\text{CLOSE}\quad \frac{P \xrightarrow{a(u)} P',\ \ Q \xrightarrow{\bar{a}\boldsymbol{\nu} u} Q'}{P|Q \xrightarrow{\tau} (\boldsymbol{\nu} u)(P'|Q')}$$

Early Bisimilarity for Pi

Definition

R is a ***simulation*** iff for $\forall (P, Q) \in R$:
If $P \xrightarrow{\alpha} P'$ then
(1) If α is an input $a(x)$ then $\forall u$, $\exists Q'$:
$Q \xrightarrow{\alpha} Q' \;\wedge\; (P'[u/x], Q'[u/x]) \in R$
(2) If α is not an input then $\exists Q : Q \xrightarrow{\alpha} Q' \;\wedge\; (P', Q') \in R$

.

Definition

(Early) Bisimilarity: $P \sim_E Q$ iff there is R such R and R^{-1} are simulations and $(P, Q) \in R$.

Example: Let $P = x(z) \parallel \bar{y}u$ and $Q = x(z).\bar{y}u + \bar{y}u.x(z)$.
Then $a(x).P \not\sim_E a(x).Q$ but $P \sim_E Q$.

Early Congruence for Pi.

Definition

Define $P \approx_E Q$ iff $P\sigma \sim_E Q\sigma$ for all substitutions σ.

Example: Note that $x(z) \parallel \bar{y}u \not\approx_E x(z).\bar{y}u + \bar{y}u.x(z)$. In fact,

The relation $\approx_E$ is a congruence. Furthermore $\approx_E = \approx_B$!.

Thanks

Example

On first slide.

Example

On second slide.

Thanks

Example

On first slide.

Example

On second slide.

Make Titles Informative.

Theorem

On first slide.

Corollary

On second slide.

Make Titles Informative.

Theorem

On first slide.

Corollary

On second slide.

Make Titles Informative.

Theorem

In left column.

Corollary

In right column.
New line

Make Titles Informative.

Theorem

In left column.

Corollary

In right column.
New line

Summary

- The first main message of your talk in one or two lines.
- The second main message of your talk in one or two lines.
- Perhaps a third message, but not more than that.

- Outlook
  - What we have not done yet.
  - Even more stuff.