Open Bisimulation, Revisited

S. Briais U. Nestmann

School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne

12th International Workshop on
Expressiveness in Concurrency
27 August, 2005
San Francisco, USA

Outline

Outline

Syntax

- Processes

$$P, Q \quad ::= \quad \mathbf{0} \;\Big|\; \pi.P \;\Big|\; !P \;\Big|\; (\nu z)\,P \;\Big|\; P \| Q \;\Big|\; [x = y]P \;\Big|\; P + Q$$

- Prefixes

$$\pi \quad ::= \quad \tau \;\Big|\; x(z) \;\Big|\; \overline{x}\langle z \rangle$$

Syntax

- Processes

$$P, Q \quad ::= \quad \mathbf{0} \;\Big|\; \pi.P \;\Big|\; !P \;\Big|\; (\nu z)\,P \;\Big|\; P \| Q \;\Big|\; [x = y]P \;\Big|\; P + Q$$

- Prefixes

$$\pi \quad ::= \quad \tau \;\Big|\; x(z) \;\Big|\; \overline{x}\langle z \rangle$$

- Only names

Labelled Semantics

$$\textsc{Input}\ \frac{}{a(x).P \xrightarrow{a(x)} P} \qquad \textsc{Open}\ \frac{P \xrightarrow{\bar{a}z} P'}{(\nu z)P \xrightarrow{(\nu z)\bar{a}z} P'}\ z \neq a$$

$$\textsc{Close-L}\ \frac{P \xrightarrow{a(x)} P' \qquad Q \xrightarrow{(\nu z)\bar{a}z} Q'}{P \| Q \xrightarrow{\tau} (\nu z)(P'\{^{z}/_{x}\} \| Q')}\ z \notin \mathrm{fn}(P)$$

Outline

Bisimulation

- Proof techniques for showing process equivalence

Bisimulation

- Proof techniques for showing process equivalence
- Wide variety of bisimulations: ground, early, late, open, . . .

Bisimulation

- Proof techniques for showing process equivalence
- Wide variety of bisimulations: ground, early, late, open, . . .
- The above cited differ on how they handle substitutions

Bisimulation

- Proof techniques for showing process equivalence
- Wide variety of bisimulations: ground, early, late, open, . . .
- The above cited differ on how they handle substitutions
- For example, ground: no substitutions at all

Bisimulation

- Proof techniques for showing process equivalence
- Wide variety of bisimulations: ground, early, late, open, ...
- The above cited differ on how they handle substitutions
- For example, ground: no substitutions at all

$$P \mid Q$$

Bisimulation

- Proof techniques for showing process equivalence
- Wide variety of bisimulations: ground, early, late, open, ...
- The above cited differ on how they handle substitutions
- For example, ground: no substitutions at all

$$\begin{array}{ccc} P & \xrightarrow{\alpha} & P' \\ | & & \\ Q & & \end{array}$$

Bisimulation

- Proof techniques for showing process equivalence
- Wide variety of bisimulations: ground, early, late, open, ...
- The above cited differ on how they handle substitutions
- For example, ground: no substitutions at all

$$\begin{array}{ccc} P & \xrightarrow{\alpha} & P' \\ | & & \\ Q & \xrightarrow{\alpha} & \end{array}$$

Bisimulation

- Proof techniques for showing process equivalence
- Wide variety of bisimulations: ground, early, late, open, ...
- The above cited differ on how they handle substitutions
- For example, ground: no substitutions at all

$$\begin{array}{ccc} P & \xrightarrow{\alpha} & P' \\ | & & \\ Q & \xrightarrow{\alpha} & Q' \end{array}$$

Bisimulation

- Proof techniques for showing process equivalence
- Wide variety of bisimulations: ground, early, late, open, ...
- The above cited differ on how they handle substitutions
- For example, ground: no substitutions at all

$$\begin{array}{ccc} P & \xrightarrow{\alpha} & P' \\ | & & | \\ Q & \xrightarrow{\alpha} & Q' \end{array}$$

Substitutions

A substitution

- has finite domain

Substitutions

A substitution

- has finite domain
- replaces something (a name) with something (e.g.: a name)

Substitutions

A substitution

- has finite domain
- replaces something (a name) with something (e.g.: a name)
- can be lifted to bigger entities (e.g.: messages)

Substitutions

A substitution

- has finite domain
- replaces something (a name) with something (e.g.: a name)
- can be lifted to bigger entities (e.g.: messages)

Some questions when designing a bisimulation:

Substitutions

A substitution

- has finite domain
- replaces something (a name) with something (e.g.: a name)
- can be lifted to bigger entities (e.g.: messages)

Some questions when designing a bisimulation:

- When should substitutions be applied?

Substitutions

A substitution

- has finite domain
- replaces something (a name) with something (e.g.: a name)
- can be lifted to bigger entities (e.g.: messages)

Some questions when designing a bisimulation:

- When should substitutions be applied?
- Which names are substitutable?

Substitutions

A substitution

- has finite domain
- replaces something (a name) with something (e.g.: a name)
- can be lifted to bigger entities (e.g.: messages)

Some questions when designing a bisimulation:

- When should substitutions be applied?
- Which names are substitutable?
- By what?

Late and open

- Late bisimulation

Late and open

- Late bisimulation

$$P \mid Q$$

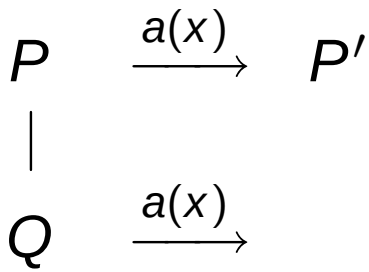
Late and open

- Late bisimulation

$$\begin{array}{ccc} P & \xrightarrow{a(x)} & P' \\ | & & \\ Q & & \end{array}$$

Late and open

- Late bisimulation



Late and open

- Late bisimulation

$$\begin{array}{ccc} P & \xrightarrow{a(x)} & P' \\ | & & \\ Q & \xrightarrow{a(x)} & Q' \end{array}$$

Late and open

- Late bisimulation

$$\begin{array}{ccccc} P & \xrightarrow{a(x)} & P' & & P'\{^{z}/_{x}\} \\ | & & & & | \\ Q & \xrightarrow{a(x)} & Q' & & Q'\{^{z}/_{x}\} \end{array}$$

Late and open

- Late bisimulation

$$\begin{array}{ccccc} P & \xrightarrow{a(x)} & P' & & P'\{^{z}/_{x}\} \\ | & & & & | \\ Q & \xrightarrow{a(x)} & Q' & & Q'\{^{z}/_{x}\} \end{array}$$

- Open bisimulation

Late and open

- Late bisimulation

$$\begin{array}{ccccc} P & \xrightarrow{a(x)} & P' & & P'\{^{z}/_{x}\} \\ | & & & & | \\ Q & \xrightarrow{a(x)} & Q' & & Q'\{^{z}/_{x}\} \end{array}$$

- Open bisimulation

$$\begin{array}{c} P \\ | \\ Q \end{array}$$

Late and open

- Late bisimulation

$$\begin{array}{ccccc} P & \xrightarrow{a(x)} & P' & & P'\{^{z}/_{x}\} \\ | & & & & | \\ Q & \xrightarrow{a(x)} & Q' & & Q'\{^{z}/_{x}\} \end{array}$$

- Open bisimulation

$$\begin{array}{cc} P & P\sigma \\ | & \\ Q & \end{array}$$

Late and open

- Late bisimulation

$$\begin{array}{ccccc} P & \xrightarrow{a(x)} & P' & & P'\{^{z}/_{x}\} \\ | & & & & | \\ Q & \xrightarrow{a(x)} & Q' & & Q'\{^{z}/_{x}\} \end{array}$$

- Open bisimulation

$$\begin{array}{ccccc} P & & P\sigma & \xrightarrow{\alpha} & P' \\ | & & & & \\ Q & & & & \end{array}$$

Late and open

- Late bisimulation

$$\begin{array}{ccccc} P & \xrightarrow{a(x)} & P' & & P'\{^{z}/_{x}\} \\ | & & & & | \\ Q & \xrightarrow{a(x)} & Q' & & Q'\{^{z}/_{x}\} \end{array}$$

- Open bisimulation

$$\begin{array}{ccccc} P & & P\sigma & \xrightarrow{\alpha} & P' \\ | & & & & \\ Q & & Q\sigma & & \end{array}$$

Late and open

- Late bisimulation

$$\begin{array}{ccccc} P & \xrightarrow{a(x)} & P' & & P'\{^{z}/_{x}\} \\ | & & & & | \\ Q & \xrightarrow{a(x)} & Q' & & Q'\{^{z}/_{x}\} \end{array}$$

- Open bisimulation

$$\begin{array}{ccccc} P & & P\sigma & \xrightarrow{\alpha} & P' \\ | & & & & \\ Q & & Q\sigma & \xrightarrow{\alpha} & Q' \end{array}$$

Late and open

- Late bisimulation

$$\begin{array}{ccccc} P & \xrightarrow{a(x)} & P' & & P'\{^{z}/_{x}\} \\ | & & & & | \\ Q & \xrightarrow{a(x)} & Q' & & Q'\{^{z}/_{x}\} \end{array}$$

- Open bisimulation

$$\begin{array}{ccccc} P & & P\sigma & \xrightarrow{\alpha} & P' \\ | & & & & | \\ Q & & Q\sigma & \xrightarrow{\alpha} & Q' \end{array}$$

Late and open

- Late bisimulation

$$\begin{array}{ccc} P & \xrightarrow{a(x)} & P' \\ | & & | \\ Q & \xrightarrow{a(x)} & Q' \end{array} \qquad \begin{array}{c} P'\{^{z}/_{x}\} \\ | \\ Q'\{^{z}/_{x}\} \end{array}$$

- Open bisimulation

$$\sigma \triangleright D \qquad \begin{array}{ccc} P & P\sigma & \xrightarrow{\alpha} & P' \\ | & & & | \\ Q & Q\sigma & \xrightarrow{\alpha} & Q' \end{array}$$

Indexed by a *distinction* D.

Late and open

- Late bisimulation

$$\begin{array}{ccc} P & \xrightarrow{a(x)} & P' \qquad P'\{^{z}/_{x}\} \\ | & & | \\ Q & \xrightarrow{a(x)} & Q' \qquad Q'\{^{z}/_{x}\} \end{array}$$

- Open bisimulation

$$\sigma \triangleright D \quad \begin{array}{ccc} P & P\sigma & \xrightarrow{\alpha} P' \\ | & & | \\ Q & Q\sigma & \xrightarrow{\alpha} Q' \end{array}$$

Indexed by a *distinction* D.
In the following, we concentrate on open.

The lazy flavour of open

$$
\begin{aligned}
P &\stackrel{\text{def}}{=} c(x).(\tau + \tau.\tau + \tau.[x = a]\tau) \\
Q &\stackrel{\text{def}}{=} c(x).(\tau + \tau.\tau)
\end{aligned}
$$

The lazy flavour of open

$$P \stackrel{\text{def}}{=} c(x).(\tau + \tau.\tau + \tau.[x = a]\tau)$$
$$Q \stackrel{\text{def}}{=} c(x).(\tau + \tau.\tau)$$

- P and Q are late bisimilar but not open

The lazy flavour of open

$$P \stackrel{\text{def}}{=} c(x).(\tau + \tau.\tau + \tau.[x = a]\tau)$$
$$Q \stackrel{\text{def}}{=} c(x).(\tau + \tau.\tau)$$

- P and Q are late bisimilar but not open
- In open, the instantiation of x can be delayed until x is used

The lazy flavour of open

$$P \stackrel{\text{def}}{=} c(x).(\tau + \tau.\tau + \tau.[x = a]\tau)$$
$$Q \stackrel{\text{def}}{=} c(x).(\tau + \tau.\tau)$$

- P and Q are late bisimilar but not open
- In open, the instantiation of x can be delayed until x is used
- Open is “very late”

Some properties of open

Some properties of open

- Contrary to early or late, it is a full congruence

Some properties of open

- Contrary to early or late, it is a full congruence
- It is easily implementable (Mobility Workbench, ABC)

Some properties of open

- Contrary to early or late, it is a full congruence
- It is easily implementable (Mobility Workbench, ABC)

For these reasons, we wanted to extend open to the spi-calculus.

Outline

The spi-calculus

The spi-calculus

- To model and study cryptographic protocols.

The spi-calculus

- To model and study cryptographic protocols.
- Messages

$$M, N \quad ::= \quad x \quad \mid \quad (M\,.\,N) \quad \mid \quad \mathrm{E}_N(M)$$

The spi-calculus

- To model and study cryptographic protocols.
- Messages

$$M, N \quad ::= \quad x \quad | \quad (M\,.\,N) \quad | \quad \mathrm{E}_N(M)$$

- Expressions

$$\begin{array}{lll} E, F & ::= & x \\ & | & (E\,.\,F) \quad | \quad \pi_1(E) \quad | \quad \pi_2(E) \\ & | & \mathrm{E}_F(E) \quad | \quad \mathrm{D}_F(E) \end{array}$$

The spi-calculus

- To model and study cryptographic protocols.
- Messages

$$M, N \quad ::= \quad x \quad | \quad (M\,.\,N) \quad | \quad \mathrm{E}_N(M)$$

- Expressions

$$\begin{array}{lll} E, F \quad ::= \quad x & & \\ \quad | \quad (E\,.\,F) & | \quad \pi_1(E) & | \quad \pi_2(E) \\ \quad | \quad \mathrm{E}_F(E) & | \quad \mathrm{D}_F(E) & \end{array}$$

- Guards

$$\phi \quad ::= \quad [E = F] \quad | \quad [E : \mathcal{N}]$$

Open in spi?

- Consider

$$P \stackrel{\text{def}}{=} (\nu k)\,(\nu m)\,\overline{a}\langle \mathsf{E}_k(m)\rangle.a(x).(\overline{a}\langle k\rangle \| [x=k]\overline{a}\langle a\rangle)$$

Open in spi?

- Consider

$$P \stackrel{\text{def}}{=} (\nu k)\,(\nu m)\,\overline{a}\langle \mathsf{E}_k(m)\rangle.a(x).(\overline{a}\langle k\rangle \| [x=k]\overline{a}\langle a\rangle)$$

- The guard $[x=k]$ can never be true.

Open in spi?

- Consider

$$P \stackrel{\text{def}}{=} (\nu k)\,(\nu m)\,\overline{a}\langle \mathsf{E}_k(m)\rangle.a(x).(\overline{a}\langle k\rangle \| [x=k]\overline{a}\langle a\rangle)$$

- The guard $[x=k]$ can never be true.
- The name k has been extruded when performing $\overline{a}\,\mathsf{E}_k(m)$.

Open in spi?

- Consider

$$P \stackrel{\text{def}}{=} (\nu k)\,(\nu m)\,\overline{a}\langle \mathsf{E}_k(m)\rangle.a(x).(\overline{a}\langle k\rangle \| [x=k]\overline{a}\langle a\rangle)$$

- The guard $[x=k]$ can never be true.
- The name k has been extruded when performing $\overline{a}\,\mathsf{E}_k(m)$.
- What are the possible values for x?

Open in spi?

- Consider

$$P \stackrel{\text{def}}{=} (\nu k)\,(\nu m)\,\overline{a}\langle \mathsf{E}_k(m)\rangle.a(x).(\overline{a}\langle k\rangle \| [x=k]\overline{a}\langle a\rangle)$$

- The guard $[x=k]$ can never be true.
- The name k has been extruded when performing $\overline{a}\,\mathsf{E}_k(m)$.
- What are the possible values for x?
  a

Open in spi?

- Consider

$$P \stackrel{\text{def}}{=} (\nu k)\,(\nu m)\,\overline{a}\langle \mathsf{E}_k(m)\rangle.a(x).(\overline{a}\langle k\rangle \| [x=k]\overline{a}\langle a\rangle)$$

- The guard $[x=k]$ can never be true.
- The name k has been extruded when performing $\overline{a}\,\mathsf{E}_k(m)$.
- What are the possible values for x?
  a, z for any z fresh

Open in spi?

- Consider

$$P \stackrel{\text{def}}{=} (\nu k)\,(\nu m)\,\overline{a}\langle \mathsf{E}_k(m)\rangle.a(x).(\overline{a}\langle k\rangle \| [x=k]\overline{a}\langle a\rangle)$$

- The guard $[x=k]$ can never be true.
- The name k has been extruded when performing $\overline{a}\,\mathsf{E}_k(m)$.
- What are the possible values for x?
  a, z for any z fresh(not in $\{k, m, a\}$)

Open in spi?

- Consider

$$P \stackrel{\text{def}}{=} (\nu k)\,(\nu m)\,\overline{a}\langle \mathsf{E}_k(m)\rangle.a(x).(\overline{a}\langle k\rangle \| [x=k]\overline{a}\langle a\rangle)$$

- The guard $[x=k]$ can never be true.
- The name k has been extruded when performing $\overline{a}\,\mathsf{E}_k(m)$.
- What are the possible values for x?
  a, z for any z fresh(not in $\{k, m, a\}$), $\mathsf{E}_k(m)$

Open in spi?

- Consider

$$P \stackrel{\text{def}}{=} (\nu k)\,(\nu m)\,\overline{a}\langle \mathsf{E}_k(m)\rangle.a(x).(\overline{a}\langle k\rangle \| [x=k]\overline{a}\langle a\rangle)$$

- The guard $[x=k]$ can never be true.
- The name k has been extruded when performing $\overline{a}\,\mathsf{E}_k(m)$.
- What are the possible values for x?
  a, z for any z fresh(not in $\{k, m, a\}$), $\mathsf{E}_k(m)$
  and any message built with these “bricks”

Bisimulations in spi

- Bisimulations of π-calculus are two strong

Bisimulations in spi

- Bisimulations of π-calculus are two strong

$$P(m) \stackrel{\text{def}}{=} (\nu k)\, \overline{a}\langle \mathsf{E}_k(m) \rangle$$

Bisimulations in spi

- Bisimulations of π-calculus are two strong

$$P(m) \stackrel{\text{def}}{=} (\nu k)\, \overline{a}\langle \mathsf{E}_k(m) \rangle$$

For any m and n, we want $P(m)$ and $P(n)$ equivalent.

Bisimulations in spi

- Bisimulations of π-calculus are two strong

$$P(m) \stackrel{\text{def}}{=} (\nu k)\, \overline{a}\langle \mathsf{E}_k(m) \rangle$$

  For any m and n, we want $P(m)$ and $P(n)$ equivalent.
- Abadi and Gordon have introduced environment-sensitive bisimulation.

Outline

Different kinds of free names

$$P \stackrel{\text{def}}{=} a(x).(\nu k)\,\overline{b}\langle k\rangle.\overline{x}\langle k\rangle.\,\mathbf{0}$$

A free name is

Different kinds of free names

$$P \stackrel{\text{def}}{=} a(x).(\nu k)\,\overline{b}\langle k\rangle.\overline{x}\langle k\rangle.\,\mathbf{0}$$

A free name is

- either initially free

Different kinds of free names

$$P \stackrel{\text{def}}{=} a(x).(\nu k)\,\overline{b}\langle k\rangle.\overline{x}\langle k\rangle.\,\mathbf{0}$$

A free name is

- either initially free
- or becomes free after an input

Different kinds of free names

$$P \stackrel{\text{def}}{=} a(x).(\nu k)\,\overline{b}\langle k\rangle.\overline{x}\langle k\rangle.\,\mathbf{0}$$

A free name is

- either initially free
- or becomes free after an input
- or becomes free by scope extrusion

Different kinds of free names

$$P \stackrel{\text{def}}{=} a(x).(\nu k)\,\overline{b}\langle k\rangle.\overline{x}\langle k\rangle.\,\mathbf{0}$$

A free name is

- either initially free
- or becomes free after an input
- or becomes free by scope extrusion
- The first two kinds are substitutable:

Different kinds of free names

$$P \stackrel{\text{def}}{=} a(x).(\nu k)\,\overline{b}\langle k\rangle.\overline{x}\langle k\rangle.\mathbf{0}$$

A free name is

- either initially free
- or becomes free after an input
- or becomes free by scope extrusion
- The first two kinds are substitutable:
  - by any name that was known at the moment they became free or

Different kinds of free names

$$P \stackrel{\text{def}}{=} a(x).(\nu k)\,\overline{b}\langle k\rangle.\overline{x}\langle k\rangle.\mathbf{0}$$

A free name is

- either initially free
- or becomes free after an input
- or becomes free by scope extrusion
- The first two kinds are substitutable:
  - by any name that was known at the moment they became free or
  - any fresh name.

Refining distinctions

- A distinction is a finite list of *in*equalities between names.

Refining distinctions

- A distinction is a finite list of *in*equalities between names.
- We take a dual approach for constraining admissible substitutions.

Refining distinctions

- A distinction is a finite list of *in*equalities between names.
- We take a dual approach for constraining admissible substitutions.
- $e = (C, V, \prec)$

Refining distinctions

- A distinction is a finite list of *in*equalities between names.
- We take a dual approach for constraining admissible substitutions.
- $e = (C, V, \prec)$
  - C contains the emitted names (or messages) not in V

Refining distinctions

- A distinction is a finite list of *in*equalities between names.
- We take a dual approach for constraining admissible substitutions.
- $e = (C, V, \prec)$
  - C contains the emitted names (or messages) not in V
  - V contains the input names and the initially free ones

Refining distinctions

- A distinction is a finite list of *in*equalities between names.
- We take a dual approach for constraining admissible substitutions.
- $e = (C, V, \prec)$
  - C contains the emitted names (or messages) not in V
  - V contains the input names and the initially free ones
  - $\prec$ indicates for each $x \in V$ which names in C were known before

Environments

$$P \stackrel{\text{def}}{=} (\nu k)\,\overline{a}\langle k\rangle.a(x).((\nu l)\,\overline{b}\langle l\rangle \| [x=k]\overline{a}\langle a\rangle)$$

C	V	$\prec$
$\emptyset$	$\{a, b\}$	$\emptyset$

$$D = \emptyset$$

Environments

$$P \stackrel{\text{def}}{=} (\nu k)\,\overline{a}\langle k\rangle.a(x).((\nu l)\,\overline{b}\langle l\rangle \| [x = k]\overline{a}\langle a\rangle)$$

C	V	$\prec$
$\emptyset$	$\{a, b\}$	$\emptyset$
$\{k\}$	$\{a, b\}$	$\emptyset$

$$D = k \neq a, k \neq b$$

Environments

$$P \stackrel{\text{def}}{=} (\nu k)\,\overline{a}\langle k\rangle.a(x).((\nu l)\,\overline{b}\langle l\rangle \| [x=k]\overline{a}\langle a\rangle)$$

C	V	$\prec$
$\emptyset$	$\{a,b\}$	$\emptyset$
$\{k\}$	$\{a,b\}$	$\emptyset$
$\{k\}$	$\{a,b,x\}$	$\{(k,x)\}$

$$D = k \neq a, k \neq b$$

Environments

$$P \stackrel{\text{def}}{=} (\nu k)\,\overline{a}\langle k\rangle.a(x).((\nu l)\,\overline{b}\langle l\rangle \| [x = k]\overline{a}\langle a\rangle)$$

C	V	$\prec$
$\emptyset$	$\{a, b\}$	$\emptyset$
$\{k\}$	$\{a, b\}$	$\emptyset$
$\{k\}$	$\{a, b, x\}$	$\{(k, x)\}$

$$D = k \neq a, k \neq b$$

Environments

$$P \stackrel{\text{def}}{=} (\nu k)\,\overline{a}\langle k\rangle.a(x).((\nu l)\,\overline{b}\langle l\rangle \| [x = l]\overline{a}\langle a\rangle)$$

C	V	$\prec$
$\emptyset$	$\{a, b\}$	$\emptyset$
$\{k\}$	$\{a, b\}$	$\emptyset$
$\{k\}$	$\{a, b, x\}$	$\{(k, x)\}$

$$D = k \neq a, k \neq b$$

Environments

$$P \stackrel{\text{def}}{=} (\nu k)\,\overline{a}\langle k\rangle.a(x).((\nu l)\,\overline{b}\langle l\rangle \| [x=k]\overline{a}\langle a\rangle)$$

C	V	$\prec$
$\emptyset$	$\{a,b\}$	$\emptyset$
$\{k\}$	$\{a,b\}$	$\emptyset$
$\{k\}$	$\{a,b,x\}$	$\{(k,x)\}$
$\{k,l\}$	$\{a,b,x\}$	$\{(k,x)\}$

$$D = k \neq a, k \neq b, l \neq a, l \neq b, l \neq x, k \neq l$$

Environments

$$P \stackrel{\text{def}}{=} (\nu k)\,\overline{a}\langle k\rangle.a(x).((\nu l)\,\overline{b}\langle l\rangle \| [x = k]\overline{a}\langle a\rangle)$$

C	V	$\prec$
$\emptyset$	$\{a, b\}$	$\emptyset$
$\{k\}$	$\{a, b\}$	$\emptyset$
$\{k\}$	$\{a, b, x\}$	$\{(k, x)\}$
$\{k, l\}$	$\{a, b, x\}$	$\{(k, x)\}$

$$D = k \neq a, k \neq b, l \neq a, l \neq b, l \neq x, k \neq l$$

Environments

$$P \stackrel{\text{def}}{=} (\nu k)\,\overline{a}\langle k\rangle.a(x).((\nu l)\,\overline{b}\langle l\rangle \| [x=l]\overline{a}\langle a\rangle)$$

C	V	$\prec$
$\emptyset$	$\{a,b\}$	$\emptyset$
$\{k\}$	$\{a,b\}$	$\emptyset$
$\{k\}$	$\{a,b,x\}$	$\{(k,x)\}$
$\{k,l\}$	$\{a,b,x\}$	$\{(k,x)\}$

$$D = k \neq a, k \neq b, l \neq a, l \neq b, l \neq x, k \neq l$$

Environments

$$P \stackrel{\text{def}}{=} (\nu k)\,\overline{a}\langle k\rangle.a(x).((\nu l)\,\overline{b}\langle l\rangle \| [x=k]\overline{a}\langle a\rangle)$$

C	V	$\prec$
$\emptyset$	$\{a,b\}$	$\emptyset$
$\{k\}$	$\{a,b\}$	$\emptyset$
$\{k\}$	$\{a,b,x\}$	$\{(k,x)\}$
$\{k,l\}$	$\{a,b,x\}$	$\{(k,x)\}$

$$D = k \neq a, k \neq b, l \neq a, l \neq b, l \neq x, k \neq l$$

Refining distinctions

- A distinction is a finite list of *in*equalities between names.
- We take a dual approach for constraining admissible substitutions.
- $e = (C, V, \prec)$
  - C contains the emitted names (or messages) not in V
  - V contains the input names and the initially free ones
  - $\prec$ indicates for each $x \in V$ which names in C were known before
- A substitution σ respects e if
  $supp(\sigma) \subseteq V$ and σ does not "contradict" $\prec$

Refining distinctions

- A distinction is a finite list of *in*equalities between names.
- We take a dual approach for constraining admissible substitutions.
- $e = (C, V, \prec)$
  - C contains the emitted names (or messages) not in V
  - V contains the input names and the initially free ones
  - $\prec$ indicates for each $x \in V$ which names in C were known before
- A substitution σ respects e if
  $supp(\sigma) \subseteq V$ and σ does not "contradict" $\prec$
- The corresponding distinction is

$$D(C, V, \prec) \stackrel{\text{def}}{=} C^{\neq} \cup \{n \neq x \mid n \in C \wedge \neg(n \prec x)\}$$

Some results

We have

Some results

We have

- $$P \approx_{\mathrm{K}}^{(C,V,\prec)} Q \Rightarrow P \approx_{\mathrm{O}}^{D(C,V,\prec)} Q$$

Some results

We have

- $$P \approx_{\mathrm{K}}^{(C,V,\prec)} Q \Rightarrow P \approx_{\mathrm{O}}^{D(C,V,\prec)} Q$$
- $$P \approx_{\mathrm{O}}^{D(C,V,\prec)} Q \Rightarrow P \approx_{\mathrm{K}}^{(C,V,\prec)} Q$$

Some results

We have

- $$P \approx_{\mathsf{K}}^{(C,V,\prec)} Q \Rightarrow P \approx_{\mathsf{O}}^{D(C,V,\prec)} Q$$
- $$P \approx_{\mathsf{O}}^{D(C,V,\prec)} Q \Rightarrow P \approx_{\mathsf{K}}^{(C,V,\prec)} Q$$
- In particular
$$P \approx_{\mathsf{O}}^{\emptyset} Q \Leftrightarrow P \approx_{\mathsf{K}}^{(\emptyset,\mathsf{fn}(P+Q),\emptyset)} Q$$

Conclusion

Conclusion

- We have defined K-open bisimulation and proved it coincides with open whenever it is defined

Conclusion

- We have defined K-open bisimulation and proved it coincides with open whenever it is defined
- We conjecture that if D can be expressed in terms of $(C, V, \prec)$, then $\approx_{\mathrm{O}}^{D}$ is more than a D-congruence

Conclusion

- We have defined K-open bisimulation and proved it coincides with open whenever it is defined
- We conjecture that if D can be expressed in terms of $(C, V, \prec)$, then $\approx_{\mathrm{O}}^{D}$ is more than a D-congruence
- We have a proposal for an extension of K-open to spi which is sound w.r.t. barbed equivalence via late hedged bisimulation

Future work

Future work

- Study the congruence properties of K-open

Future work

- Study the congruence properties of K-open
- Study the extension of K-open to spi

Future work

- Study the congruence properties of K-open
- Study the extension of K-open to spi
  - Link with symbolic bisimulation of [BBN04]
  - Congruence properties

Thank you!

Thank you!
Questions?

Bibliography

D. Sangiorgi
A Theory of Bisimulation for the π-calculus.

J. Borgström, S. Briais and U. Nestmann
Symbolic Bisimulations in the Spi Calculus