Open Bisimulation, Revisited

S. Briais  U. Nestmann

School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne

COMETE-PARSIFAL Seminar
11 May, 2006
Paris, FRANCE
Outline

1. The pi-calculus
2. Bisimulations
3. The spi-calculus
4. K-open bisimulation
5. Open hedged bisimulation
Outline

1. The pi-calculus
2. Bisimulations
3. The spi-calculus
4. K-open bisimulation
5. Open hedged bisimulation
The pi-calculus

Syntax

Processes

\[ P, Q ::= 0 \mid ! P \mid \pi.P \mid (\nu z) P \mid P|Q \mid P + Q \]

Prefixes

\[ \pi ::= \tau \mid x(z) \mid \overline{x}(z) \]
Syntax

- **Processes**

  \[
  P, Q ::= \emptyset \mid ! P \mid \pi.P \mid (\nu z) P \mid P \parallel Q \mid P + Q
  \]

- **Prefixes**

  \[
  \pi ::= \tau \mid x(z) \mid \overline{x}(z)
  \]

- **Only names**
Late Labelled Semantics

**Input**

\[ a(x).P \xrightarrow{a(x)} P \]

**Open**

\[ P \xrightarrow{\overline{a}z} P' \]

\[ (\nu z) P \xrightarrow{(\nu z)\overline{a}z} P' \quad z \neq a \]

**Close-L**

\[ P \xrightarrow{a(x)} P' \]

\[ Q \xrightarrow{(\nu z)\overline{a}z} Q' \]

\[ P \parallel Q \xrightarrow{\tau} (\nu z) (P' \{z/x\} \parallel Q') \]

\[ z \notin \text{fn}(P) \]
Outline

1. The pi-calculus
2. Bisimulations
3. The spi-calculus
4. K-open bisimulation
5. Open hedged bisimulation
Bisimulation

- Proof techniques for showing process equivalence
Bisimulation

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

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

\[ P \xrightarrow{\alpha} P' \]

\[ Q \]
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

\[
P \xrightarrow{\alpha} P' \\
\vert \\
Q \xrightarrow{\alpha}
\]
Bisimulations

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

\[
P \xrightarrow{\alpha} P' \quad \quad Q \xrightarrow{\alpha} Q'
\]
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

\[
P \xrightarrow{\alpha} P' \\
\downarrow \quad \downarrow \\
Q \xrightarrow{\alpha} Q'
\]
Substitutions

A substitution

- has finite domain
Substitutions

A substitution

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

A substitution

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

A substitution

- has finite domain
- replaces something (a name) by 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) by 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) by 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) by 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?
Early and late bisimulation

- The *symmetric* relation $R \subseteq \mathcal{P} \times \mathcal{P}$ is

  i) if $\alpha$ is not an input, there exists $Q'$ such that $Q \xrightarrow{\alpha} Q'$ and $P' R Q'$

  ii) if $\alpha = a(x)$, then for all $u \in N$, there exists $Q'$ such that $Q \xrightarrow{\alpha} Q'$ and $(P'\{u/x\}, Q'\{u/x\}) \in R$
Early and late bisimulation

- The symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is
  - an early bisimulation if for all $(P, Q) \in \mathcal{R}$, if $P \xrightarrow{\alpha} P'$ then
    - if $\alpha$ is not an input, there exists $Q' \in \mathcal{P}$ such that $Q \xrightarrow{\alpha} Q'$ and $(P',\{u/x\}, Q',\{u/x\}) \in \mathcal{R}$
    - if $\alpha = a(x)$, then for all $u \in N$, there exists $Q' \in \mathcal{P}$ such that $Q \xrightarrow{\alpha} Q'$ and $(P',\{u/x\}, Q',\{u/x\}) \in \mathcal{R}$
  - a late bisimulation if instead of ii), it satisfies
    - ii') if $\alpha = a(x)$, then there exists $Q' \in \mathcal{P}$ such that $Q \xrightarrow{\alpha} Q'$ and for all $u \in N$, $(P',\{u/x\}, Q',\{u/x\}) \in \mathcal{R}$
Early and late bisimulation

- The symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is
  - an early bisimulation if for all $(P, Q) \in \mathcal{R}$, if $P \xrightarrow{\alpha} P'$ then
    i) if $\alpha$ is not an input,
Early and late bisimulation

The symmetric relation $\mathcal{R} \subset \mathcal{P} \times \mathcal{P}$ is

- an early bisimulation if for all $(P, Q) \in \mathcal{R}$, if $P \xrightarrow{\alpha} P'$ then
  - i) if $\alpha$ is not an input, there exists $Q'$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \mathcal{R} Q'$
Early and late bisimulation

- The symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is
  - an early bisimulation if for all $(P, Q) \in \mathcal{R}$, if $P \xrightarrow{\alpha} P'$ then
    i) if $\alpha$ is not an input,
       there exists $Q'$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \mathcal{R} Q'$
    ii) if $\alpha = a(x)$,
Early and late bisimulation

- The *symmetric* relation $\mathcal{R} \subset \mathcal{P} \times \mathcal{P}$ is
  - an *early* bisimulation if for all $(P, Q) \in \mathcal{R}$, if $P \xrightarrow{\alpha} P'$ then
    i) if $\alpha$ is not an input, there exists $Q'$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \mathcal{R} Q'$
    ii) if $\alpha = a(x)$, then for all $u \in \mathcal{N}$, there exists $Q'$ such that $Q \xrightarrow{\alpha} Q'$ and
        $(P'\{u/x\}, Q'\{u/x\}) \in \mathcal{R}$
Early and late bisimulation

- The *symmetric* relation $R \subseteq P \times P$ is
  - an *early* bisimulation if for all $(P, Q) \in R$, if $P \xrightarrow{\alpha} P'$ then
    i) if $\alpha$ is not an input,
       there exists $Q'$ such that $Q \xrightarrow{\alpha} Q'$ and $P' R Q'$
    ii) if $\alpha = a(x)$,
        then for all $u \in \mathcal{N}$, there exists $Q'$ such that $Q \xrightarrow{\alpha} Q'$ and
        $(P' \{u/x\}, Q' \{u/x\}) \in R$
  - a *late* bisimulation if instead of ii), it satisfies
Early and late bisimulation

- The **symmetric** relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is
  - an *early* bisimulation if for all $(P, Q) \in \mathcal{R}$, if $P \xrightarrow{\alpha} P'$ then
    - i) if $\alpha$ is not an input,
      there exists $Q'$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \mathcal{R} Q'$
    - ii) if $\alpha = a(x)$,
      then for all $u \in \mathcal{N}$, there exists $Q'$ such that $Q \xrightarrow{\alpha} Q'$ and $(P'\{u/x\}, Q'\{u/x\}) \in \mathcal{R}$
  - a *late* bisimulation if instead of ii), it satisfies
    - ii') if $\alpha = a(x)$,
      then there exists $Q'$ such that $Q \xrightarrow{\alpha} Q'$ and for all $u \in \mathcal{N}$,
      $(P'\{u/x\}, Q'\{u/x\}) \in \mathcal{R}$
From late to open

- Late bisimulation
From late to open

- Late bisimulation

\[
P \xrightarrow{a(x)} P' = \{ z / x \}
\]

\[
Q \xrightarrow{a(x)} Q' = \{ z / x \}
\]

Indexed by a distinction D.
From late to open

- Late bisimulation

\[ P \xrightarrow{a(x)} P' \]

\[ Q \]

Indexed by a distinction \( D \).
From late to open

- Late bisimulation

\[
P \xrightarrow{a(x)} P' \\
\mid \\
Q \xrightarrow{a(x)}
\]
From late to open

- Late bisimulation

\[
P \xrightarrow{a(x)} P' \\
\mid
\]

\[
Q \xrightarrow{a(x)} Q'
\]

Indexed by a distinction \(D\).

Briais, Nestmann (EPFL)

Open Bisimulation, Revisited
From late to open

- Late bisimulation

\[
\begin{align*}
P \xrightarrow{a(x)} & \quad P' \quad P'\{\frac{z}{x}\} \\
\mid & \quad \mid \\
Q \xrightarrow{a(x)} & \quad Q' \quad Q'\{\frac{z}{x}\}
\end{align*}
\]
From late to open

- Late bisimulation

\[ P \xrightarrow{a(x)} P' \quad P' \{z/x\} \]
\[ Q \xrightarrow{a(x)} Q' \quad Q' \{z/x\} \]

- Open bisimulation
From late to open

- **Late bisimulation**

\[
\begin{align*}
P & \xrightarrow{a(x)} P' & & P' \{z/x\} \\
\vert & & \vert \\
Q & \xrightarrow{a(x)} Q' & & Q' \{z/x\}
\end{align*}
\]

- **Open bisimulation**

\[
\begin{align*}
P & \\
\vert & \\
Q
\end{align*}
\]
From late to open

- Late bisimulation
  \[ P \xrightarrow{a(x)} P' \quad P' \{z/x\} \]
  \[ Q \xrightarrow{a(x)} Q' \quad Q' \{z/x\} \]

- Open bisimulation
  \[ P \xrightarrow{a} P_\sigma \]
  \[ Q \]

Briais, Nestmann (EPFL)
From late to open

- Late bisimulation

\[ P \xrightarrow{a(x)} P' \quad P'\{\frac{z}{x}\} \]
\[ Q \xrightarrow{a(x)} Q' \quad Q'\{\frac{z}{x}\} \]

- Open bisimulation

\[ P \xrightarrow{\alpha} P' \quad P' \]
\[ Q \]
From late to open

- Late bisimulation

\[
P \xrightarrow{a(x)} P' \quad | \quad P' \{z/x\} \\
| \quad | \\
Q \xrightarrow{a(x)} Q' \quad | \quad Q' \{z/x\}
\]

- Open bisimulation

\[
P \xrightarrow{\alpha} P' \\
| \quad | \\
Q \xrightarrow{\alpha} Q'
\]
From late to open

- Late bisimulation

\[ P \xrightarrow{a(x)} P' \quad P'\{z/x\} \]
\[ Q \xrightarrow{a(x)} Q' \quad Q'\{z/x\} \]

- Open bisimulation

\[ P \xrightarrow{\alpha} P' \]
\[ Q \xrightarrow{\alpha} Q' \]
From late to open

- **Late bisimulation**

  \[ P \xrightarrow{a(x)} P' \quad P' \{z/x\} \]

  \[ Q \xrightarrow{a(x)} Q' \quad Q' \{z/x\} \]

- **Open bisimulation**

  \[ P \quad P_\sigma \xrightarrow{\alpha} P' \]

  \[ Q \quad Q_\sigma \xrightarrow{\alpha} Q' \]
From late to open

- Late bisimulation

\[ P \xrightarrow{a(x)} P' \quad P' \{z/x\} \]

\[ Q \xrightarrow{a(x)} Q' \quad Q' \{z/x\} \]

- Open bisimulation

\[ P \xrightarrow{P\sigma \ x D} P' \]

\[ Q \xrightarrow{Q\sigma \ x D} Q' \]

Indexed by a *distinction D*. 
A distinction $D$ is an irreflexive and symmetric relation between names (finite list of inequalities)
Distinctions

- A distinction $D$ is an irreflexive and symmetric relation between names (finite list of inequalities)
- A substitution $\sigma$ respects $D$ ($\sigma \triangleright D$) if $x \sigma \neq y \sigma$ for all $(x, y) \in D$. 

▶ Briais, Nestmann (EPFL) Open Bisimulation, Revisited 11 / 39
Distinctions

- A distinction $D$ is an irreflexive and symmetric relation between names (finite list of inequalities).
- A substitution $\sigma$ respects $D$ ($\sigma \triangleright D$) if $x\sigma \neq y\sigma$ for all $(x, y) \in D$.
- If $\sigma \triangleright D$, we define the updated distinction $D\sigma$. 

For example, if $D = \{(x, y), (x, z), (y, x), (z, x)\}$ then $\xrightarrow{x \mapsto u, y \mapsto u}$ respects $D$ and the updated distinction is $\{(u, y), (u, z), (y, u), (z, u)\}$. On the contrary, $\xrightarrow{x \mapsto u, y \mapsto u}$ does not respect $D$. 

Distinctions

- A distinction \( D \) is an irreflexive and symmetric relation between names (finite list of inequalities).
- A substitution \( \sigma \) respects \( D (\sigma \triangleright D) \) if \( x_\sigma \neq y_\sigma \) for all \( (x, y) \in D \).
- If \( \sigma \triangleright D \), we define the updated distinction \( D_\sigma \).
- For example, if \( D = \{(x, y), (x, z), (y, x), (z, x)\} \) then
  - \( x \mapsto u \) respects \( D \) and
Distinctions

A distinction $D$ is an irreflexive and symmetric relation between names (finite list of inequalities).

A substitution $\sigma$ respects $D$ ($\sigma \triangleright D$) if $x\sigma \neq y\sigma$ for all $(x, y) \in D$.

If $\sigma \triangleright D$, we define the updated distinction $D\sigma$.

For example, if $D = \{(x, y), (x, z), (y, x), (z, x)\}$ then

- $x \mapsto u$ respects $D$ and the updated distinction is $\{(u, y), (u, z), (y, u), (z, u)\}$
Distinctions

- A distinction $D$ is an irreflexive and symmetric relation between names (finite list of inequalities)
- A substitution $\sigma$ respects $D$ ($\sigma \triangleright D$) if $x\sigma \neq y\sigma$ for all $(x, y) \in D$.
- If $\sigma \triangleright D$, we define the updated distinction $D\sigma$.
- For example, if $D = \{(x, y), (x, z), (y, x), (z, x)\}$ then
  - $x \mapsto u$ respects $D$ and the updated distinction is $\{(u, y), (u, z), (y, u), (z, u)\}$
  - On the contrary, $x \mapsto u, y \mapsto u$ does not respect $D$
An open bisimulation is a “symmetric” relation $\mathcal{R} \subset D \times P \times P$ such that
Open Bisimulation

- An open bisimulation is a "symmetric" relation $R \subseteq D \times P \times P$ such that for all $(D, P, Q) \in R$ and $\sigma \triangleright D$, if $P\sigma \xrightarrow{\alpha} P'$ then

  ▶ if $\alpha$ is not a bound output, then $(D\sigma, P', Q') \in R$

  ▶ otherwise, if $\alpha = (\nu z)\ a z$, then $(D', P', Q') \in R$ where $D' = D\sigma \cup \{z\} \otimes (fn((P+Q)\sigma)))$
An open bisimulation is a “symmetric” relation $R \subseteq D \times P \times P$ such that for all $(D, P, Q) \in R$ and $\sigma \triangleright D$, if $P_\sigma \xrightarrow{\alpha} P'$ then $Q_\sigma \xrightarrow{\alpha} Q'$ and
An open bisimulation is a “symmetric” relation $\mathcal{R} \subset D \times P \times P$ such that for all $(D, P, Q) \in \mathcal{R}$ and $\sigma \triangleright D$, if $P_\sigma \xrightarrow{\alpha} P'$ then $Q_\sigma \xrightarrow{\alpha} Q'$ and
- if $\alpha$ is not a bound output, then $(D_\sigma, P', Q') \in \mathcal{R}$
An open bisimulation is a “symmetric” relation \( R \subseteq D \times P \times P \) such that for all \((D, P, Q) \in R\) and \(\sigma \triangleright D\), if \(P\sigma \xrightarrow{\alpha} P'\) then \(Q\sigma \xrightarrow{\alpha} Q'\) and

- if \(\alpha\) is not a bound output, then \((D\sigma, P', Q') \in R\)
- otherwise, if \(\alpha = (\nu z) \bar{a} z\), then
Open Bisimulation

An open bisimulation is a “symmetric” relation $\mathcal{R} \subset D \times P \times P$ such that for all $(D, P, Q) \in \mathcal{R}$ and $\sigma \triangleright D$, if $P_\sigma \xrightarrow{\alpha} P'$ then $Q_\sigma \xrightarrow{\alpha} Q'$ and

- if $\alpha$ is not a bound output, then $(D_\sigma, P', Q') \in \mathcal{R}$
- otherwise, if $\alpha = (\nu z) \bar{a} z$, then $(D', P', Q') \in \mathcal{R}$ where $D' = D_\sigma$
Open Bisimulation

An open bisimulation is a “symmetric” relation $\mathcal{R} \subseteq D \times P \times P$ such that for all $(D, P, Q) \in \mathcal{R}$ and $\sigma \triangleright D$, if $P_\sigma \xrightarrow{\alpha} P'$ then $Q_\sigma \xrightarrow{\alpha} Q'$ and

- if $\alpha$ is not a bound output, then $(D_\sigma, P', Q') \in \mathcal{R}$
- otherwise, if $\alpha = (\nu z) \overline{a} z$, then $(D', P', Q') \in \mathcal{R}$ where $D' = D_\sigma \cup \{z\} \otimes (\text{fn}((P + Q)_\sigma))$
Open Bisimulation

- An open bisimulation is a “symmetric” relation $\mathcal{R} \subset D \times P \times P$ such that for all $(D, P, Q) \in \mathcal{R}$ and $\sigma > D$, if $P\sigma \xrightarrow{\alpha} P'$ then $Q\sigma \xrightarrow{\alpha} Q'$ and
  - if $\alpha$ is not a bound output, then $(D\sigma, P', Q') \in \mathcal{R}$
  - otherwise, if $\alpha = (\nu z) \bar{a} z$, then $(D', P', Q') \in \mathcal{R}$ where $D' = D\sigma \cup \{z\} \otimes (\text{fn}((P + Q)\sigma))$

- Distinctions are used to forbid the fusing of fresh names with other names
The lazy flavour of open

\[ P \overset{\text{def}}{=} c(x).(\tau + \tau.\tau + \tau.[x=a]_{\tau}) \]
\[ Q \overset{\text{def}}{=} c(x).(\tau + \tau.\tau) \]
The lazy flavour of open

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

- \( P \) and \( Q \) are late bisimilar but not open
The lazy flavour of open

\[ P \overset{\text{def}}{=} \ c(x).\left(\tau + \tau.\tau + \tau.[x = a]\tau\right) \]
\[ Q \overset{\text{def}}{=} \ c(x).\left(\tau + \tau.\tau\right) \]

- \( P \) and \( Q \) are late bisimilar but not open
- In open, the instantiation of \( x \) can be delayed until \( x \) is used
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
- More precisely, open $D$-bisimilarity is a $D$-congruence
Some properties of open

- Contrary to early or late, it is a full congruence
- More precisely, open $D$-bisimilarity is a $D$-congruence
- It is easily implementable (Mobility Workbench, ABC)
Some properties of open

- Contrary to early or late, it is a full congruence
- More precisely, open $D$-bisimilarity is a $D$-congruence
- It is easily implementable (Mobility Workbench, ABC)

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

1. The pi-calculus
2. Bisimulations
3. The spi-calculus
4. K-open bisimulation
5. Open hedged bisimulation
The spi-calculus
The spi-calculus

- To model and study cryptographic protocols.
The spi-calculus

- To model and study cryptographic protocols.

Messages

\[ M, N ::= x \mid (M \cdot N) \mid E_N(M) \]
The spi-calculus

- To model and study cryptographic protocols.

Messages

\[ M, N ::= x \mid (M \cdot N) \mid E_N(M) \]

Expressions

\[ E, F ::= x \mid (E \cdot F) \mid \pi_1(E) \mid \pi_2(E) \]
\[ \quad \mid E_F(E) \mid D_F(E) \]
The spi-calculus

- To model and study cryptographic protocols.

Messages

\[ M, N ::= x \mid (M \cdot N) \mid E_N(M) \]

Expressions

\[ E, F ::= x \mid (E \cdot F) \mid \pi_1(E) \mid \pi_2(E) \]

\[ E_F(E) \mid D_F(E) \]

Guards

\[ \phi ::= [E = F] \mid [E : \mathcal{N}] \]
Evaluation of expressions and formulae

\[
\begin{align*}
\llbracket a \rrbracket & \overset{\text{def}}{=} a \\
\llbracket E_F(E) \rrbracket & \overset{\text{def}}{=} E_N(M) \quad \text{if} \quad \llbracket E \rrbracket = M \in \mathcal{M} \quad \text{and} \quad \llbracket F \rrbracket = N \in \mathcal{M} \\
\llbracket D_F(E) \rrbracket & \overset{\text{def}}{=} M \quad \text{if} \quad \llbracket E \rrbracket = E_N(M) \in \mathcal{M} \quad \text{and} \quad \llbracket F \rrbracket = N \in \mathcal{M} \\
\llbracket E \rrbracket & \overset{\text{def}}{=} \bot \quad \text{in all other cases}
\end{align*}
\]
Evaluation of expressions and formulae

\[
\begin{align*}
[[a]] & \overset{\text{def}}{=} a \\
[[E_F(E)]] & \overset{\text{def}}{=} E_N(M) \quad \text{if } [[E]] = M \in \mathcal{M} \text{ and } [[F]] = N \in \mathcal{M} \\
[[D_F(E)]] & \overset{\text{def}}{=} M \quad \text{if } [[E]] = E_N(M) \in \mathcal{M} \text{ and } [[F]] = N \in \mathcal{M} \\
[[E]] & \overset{\text{def}}{=} \bot \quad \text{in all other cases}
\end{align*}
\]

\[
\begin{align*}
[[tt]] & \overset{\text{def}}{=} \text{true} \\
[[\phi \land \psi]] & \overset{\text{def}}{=} [[\phi]] \text{ and } [[\psi]] \\
[[E = F]] & \overset{\text{def}}{=} \text{true} \quad \text{if } [[E]] = [[F]] = M \in \mathcal{M} \\
[[E : \mathcal{N}]] & \overset{\text{def}}{=} \text{true} \quad \text{if } [[E]] = a \in \mathcal{N} \\
[[\phi]] & \overset{\text{def}}{=} \text{false} \quad \text{in all other cases}
\end{align*}
\]
The wide-mouthed frog protocol

\[ A \rightarrow S : (A.E_k AS (B.k AB)) \]

\[ S \rightarrow B : E_k BS ((A.B).k AB)) \]

\[ A \rightarrow B : E_k AB (m) \]
The wide-mouthed frog protocol

1. \( A \rightarrow S : (A \cdot E_{k_{AS}}((B \cdot k_{AB}))) \)
The wide-mouthed frog protocol

1. \( A \rightarrow S : (A \cdot E_{k_{AS}}((B \cdot k_{AB}))) \)
2. \( S \rightarrow B : E_{k_{BS}}(((A \cdot B) \cdot k_{AB})) \)
The wide-mouthed frog protocol

1. $A \rightarrow S : (A \cdot E_{k_A S}((B \cdot k_{AB})))$
2. $S \rightarrow B : E_{k_{BS}}(((A \cdot B) \cdot k_{AB}))$
3. $A \rightarrow B : E_{k_{AB}}(m)$
.. in spi-calculus

\[(\nu k_{AS}, k_{BS}) \quad (\nu k_{AB}) S\langle (A . E_{k_{AS}}((B . k_{AB}))) \rangle . \overline{B}\langle E_{k_{AB}}(m) \rangle . 0 \]

\[\parallel B(x_1).\phi_1 B(x_2).\phi_2 0 \]

\[\parallel S(x_0).\phi_0 \overline{B}\langle E_{k_{BS}}(((A . B) . \pi_2(D_{k_{AS}}(\pi_2(x_0)))))) \rangle . 0 \]
\[ \begin{align*}
& (\nu k_{AS}, k_{BS}) \\
& ((\nu k_{AB}) \bar{S} \langle (A \cdot E_{k_{AS}}((B \cdot k_{AB}))) \rangle \bar{B} \langle E_{k_{AB}}(m) \rangle \cdot 0 \\
& \parallel B(x_1) \cdot \phi_1 B(x_2) \cdot \phi_2 0 \\
& \parallel S(x_0) \cdot \phi_0 \bar{B} \langle E_{k_{BS}}(((A \cdot B) \cdot \pi_2(D_{k_{AS}}(\pi_2(x_0))))) \rangle \cdot 0
\end{align*} \]

\[ \begin{align*}
\phi_0 &= \left[ B = \pi_1(D_{k_{AS}}(\pi_2(x_0))) \right] \land \left[ A = \pi_1(x_0) \right] \\
\phi_1 &= \left[ B = \pi_1(\pi_2(D_{k_{BS}}(x_1))) \right] \land \left[ A = \pi_1(D_{k_{BS}}(x_1)) \right] \\
\phi_2 &= \left[ D_{\pi_2(\pi_2(D_{k_{BS}}(x_1))(x_2))} : \mathcal{M} \right]
\end{align*} \]
Open in spi?

Consider

\[ P \overset{\text{def}}{=} (\nu k) (\nu m) \overline{a} \langle E_k(m) \rangle . a(x) . (\overline{a} \langle k \rangle \parallel [x = k] \overline{a} \langle a \rangle ) \]
Open in spi?

Consider

\[ P \overset{\text{def}}{=} (\nu k)(\nu m) \overline{a}(E_k(m)).a(x).(\overline{a}(k) || [x = k] \overline{a}(a)) \]

The guard \([x = k]\) can never be true.
Open in spi?

Consider

\[ P \overset{\text{def}}{=} (\nu k) (\nu m) \bar{a} \langle E_k(m) \rangle . a(x). (\bar{a} \langle k \rangle \parallel [x = k] \bar{a} \langle a \rangle) \]

- The guard \( [x = k] \) can never be true.
- The name \( k \) has been extruded when performing \( \bar{a} E_k(m) \).
Open in spi?

Consider

\[ P \overset{\text{def}}{=} (\nu k) (\nu m) \bar{a} \langle E_k(m) \rangle. a(x). (\bar{a} \langle k \rangle \parallel [x = k] \bar{a} \langle a \rangle) \]

- The guard \([x = k]\) can never be true.
- The name \(k\) has been extruded when performing \(\bar{a} E_k(m)\).
- What are the possible values for \(x\)?
Open in spi?

Consider

$$P \overset{\text{def}}{=} (\nu k) (\nu m) \overline{a} \langle E_k(m) \rangle . a(x). (\overline{a} \langle k \rangle \parallel [x = k] \overline{a} \langle a \rangle )$$

The guard \([x = k]\) can never be true.

The name \(k\) has been extruded when performing \(\overline{a} E_k(m)\).

What are the possible values for \(x\)?

\(a\)
Consider

\[ P \overset{\text{def}}{=} (\nu k)(\nu m) \overline{a}\langle E_k(m)\rangle. a(x). (\overline{a}\langle k\rangle \parallel [x = k] \overline{a}\langle a\rangle) \]

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

Consider

\[ P \overset{\text{def}}{=} (\nu k) (\nu m) \bar{a} E_k(m) \cdot a(x) . (\bar{a} k \parallel [x = k] \bar{a} a) \]

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

- Consider

\[ P \overset{\text{def}}{=} (\nu k) (\nu m) \overline{a}\langle E_k(m)\rangle.a(x).(\overline{a}\langle k\rangle \parallel [x = k] \overline{a}\langle a\rangle) \]

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

\[
P \overset{\text{def}}{=} (\nu k) (\nu m) \overline{a} \langle E_k(m) \rangle . a(x). (\overline{a} \langle k \rangle \parallel [x = k] \overline{a} \langle a \rangle)
\]

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

- Bisimulations of \( \pi \)-calculus are two strong
Bisimulations in spi

- Bisimulations of $\pi$-calculus are two strong

$$P(m) \overset{\text{def}}{=} (\nu k) \bar{a} \langle E_k(m) \rangle$$
Bisimulations of $\pi$-calculus are two strong

$$P(m) \overset{\text{def}}{=} (\nu k) \bar{a}\langle E_k(m) \rangle$$

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

- Bisimulations of $\pi$-calculus are two strong

\[ P(m) \overset{\text{def}}{=} (\nu k) \bar{a} \langle E_k(m) \rangle \]

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

- Abadi and Gordon have introduced environment-sensitive bisimulation.
Outline

1. The pi-calculus
2. Bisimulations
3. The spi-calculus
4. K-open bisimulation
5. Open hedged bisimulation
Different kinds of free names

\[ P \overset{\text{def}}{=} a(x).(\nu k) \overline{b}\langle k\rangle.\overline{x}\langle k\rangle.0 \]

A free name is
Different kinds of free names

\[ P \overset{\text{def}}{=} a(x).(\nu k) \overline{b}(k).\overline{x}(k).0 \]

A free name is
- either initially free
Different kinds of free names

\[ P \overset{\text{def}}{=} a(x). (\nu k) b(k). x(k). 0 \]

A free name is
- either initially free
- or becomes free after an input
Different kinds of free names

\[ P \overset{\text{def}}{=} a(x). (\nu k) \overline{b}(k). \overline{x}(k).0 \]

A free name is
- either initially free
- or becomes free after an input
- or becomes free by scope extrusion
Different kinds of free names

\[ P \overset{\text{def}}{=} a(x).(\nu k) \overline{b\langle k\rangle}. \overline{x\langle k\rangle}.0 \]

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

\[ P \overset{\text{def}}{=} a(x). (\nu k) \overline{b(k)} . \overline{x(k)} . 0 \]

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

\[ P \overset{\text{def}}{=} a(x). (\nu k) \overline{b}(k) \overline{x}(k) \cdot 0 \]

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 *inequalities* between names.
Refining distinctions

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

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

- A distinction is a finite list of *inequalities* 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 *inequalities* 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 *inequalities* 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

\[ P \overset{\text{def}}{=} (\nu k) a(k) . a(x) . ((\nu l) b(l) \parallel [x = k] a(a)) \]

<table>
<thead>
<tr>
<th>C</th>
<th>V</th>
<th>\sim</th>
</tr>
</thead>
<tbody>
<tr>
<td>\emptyset</td>
<td>{a, b}</td>
<td>\emptyset</td>
</tr>
</tbody>
</table>

\[ D = \emptyset \]
K-open bisimulation

Environments

\[ P \overset{\text{def}}{=} (\nu k) \overline{a}(k).a(x).((\nu l) \overline{b}(l) || [x = k] \overline{a}(a)) \]

<table>
<thead>
<tr>
<th>C</th>
<th>V</th>
<th>\prec</th>
</tr>
</thead>
<tbody>
<tr>
<td>\emptyset</td>
<td>{a, b}</td>
<td>\emptyset</td>
</tr>
<tr>
<td>{k}</td>
<td>{a, b}</td>
<td>\emptyset</td>
</tr>
</tbody>
</table>

\[ D = k \neq a, k \neq b \]
Environments

\[ P \overset{\text{def}}{=} (\nu k) \overline{a}(k).a(x).((\nu l) \overline{b}(l) \parallel [x = k] \overline{a}(a)) \]

<table>
<thead>
<tr>
<th>C</th>
<th>V</th>
<th>\prec</th>
</tr>
</thead>
<tbody>
<tr>
<td>\emptyset</td>
<td>{a, b}</td>
<td>\emptyset</td>
</tr>
<tr>
<td>{k}</td>
<td>{a, b}</td>
<td>\emptyset</td>
</tr>
<tr>
<td>{k}</td>
<td>{a, b, x}</td>
<td>{(k, x)}</td>
</tr>
</tbody>
</table>

\[ D = k \neq a, k \neq b \]
Environments

\[ P \overset{\text{def}}{=} (\nu k) \bar{a}(k).a(x).((\nu l) \bar{b}(l) \parallel [x = k] \bar{a}(a)) \]

\[
\begin{array}{cccc}
C & V & \prec \\
\emptyset & \{a, b\} & \emptyset \\
\{k\} & \{a, b\} & \emptyset \\
\{k\} & \{a, b, x\} & \{(k, x)\}
\end{array}
\]

\[ D = k \neq a, k \neq b \]
Environments

\[ P \overset{\text{def}}{=} (\nu k) \overline{a}(k).a(x).((\nu l) \overline{b}(l) \parallel [x = l] \overline{a}(a)) \]

<table>
<thead>
<tr>
<th>(C)</th>
<th>(V)</th>
<th>(&lt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>{a, b}</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>{k}</td>
<td>{a, b}</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>{k}</td>
<td>{a, b, x}</td>
<td>{(k, x)}</td>
</tr>
</tbody>
</table>

\[ D = k \not\equiv a, k \not\equiv b \]
Environments

\[ P \overset{\text{def}}{=} (\nu k) \overline{a}(k).a(x).((\nu l) \overline{b}(l) \parallel [x = k] \overline{a}(a)) \]

<table>
<thead>
<tr>
<th>C</th>
<th>V</th>
<th>≺</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅</td>
<td>{a, b}</td>
<td>∅</td>
</tr>
<tr>
<td>{k}</td>
<td>{a, b}</td>
<td>∅</td>
</tr>
<tr>
<td>{k}</td>
<td>{a, b, x}</td>
<td>{(k, x)}</td>
</tr>
<tr>
<td>{k, l}</td>
<td>{a, b, x}</td>
<td>{(k, x)}</td>
</tr>
</tbody>
</table>

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

\[ P \overset{\text{def}}{=} (\nu k) \bar{a}(k).a(x).((\nu l) \bar{b}(l) \parallel [x=k] \bar{a}(a)) \]

<table>
<thead>
<tr>
<th>C</th>
<th>V</th>
<th>\prec</th>
</tr>
</thead>
<tbody>
<tr>
<td>\emptyset</td>
<td>{a, b}</td>
<td>\emptyset</td>
</tr>
<tr>
<td>{k}</td>
<td>{a, b}</td>
<td>\emptyset</td>
</tr>
<tr>
<td>{k}</td>
<td>{a, b, x}</td>
<td>{(k, x)}</td>
</tr>
<tr>
<td>{k, l}</td>
<td>{a, b, x}</td>
<td>{(k, x)}</td>
</tr>
</tbody>
</table>

\[ D = k \neq a, k \neq b, l \neq a, l \neq b, l \neq x, k \neq l \]
Environments

\[ P \overset{\text{def}}{=} (\nu k) \overline{a}(k) . a(x) . ((\nu l) \overline{b}(l) \parallel [x = l] \overline{a}(a)) \]

<table>
<thead>
<tr>
<th>( C )</th>
<th>( V )</th>
<th>( \prec )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>( {a, b} )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( {k} )</td>
<td>( {a, b} )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( {k} )</td>
<td>( {a, b, x} )</td>
<td>( {(k, x)} )</td>
</tr>
<tr>
<td>( {k, l} )</td>
<td>( {a, b, x} )</td>
<td>( {(k, x)} )</td>
</tr>
</tbody>
</table>

\[ D = k \neq a, k \neq b, l \neq a, l \neq b, l \neq x, k \neq l \]
Environments

\[ P \overset{\text{def}}{=} (\nu k) \overline{a}(k).a(x).((\nu l) \overline{b}(l) \parallel [x = k] \overline{a}(a)) \]

<table>
<thead>
<tr>
<th>C</th>
<th>V</th>
<th>\prec</th>
</tr>
</thead>
<tbody>
<tr>
<td>\emptyset</td>
<td>{a, b}</td>
<td>\emptyset</td>
</tr>
<tr>
<td>{k}</td>
<td>{a, b}</td>
<td>\emptyset</td>
</tr>
<tr>
<td>{k}</td>
<td>{a, b, x}</td>
<td>{(k, x)}</td>
</tr>
<tr>
<td>{k, l}</td>
<td>{a, b, x}</td>
<td>{(k, x)}</td>
</tr>
</tbody>
</table>

\[ D = k \neq a, k \neq b, l \neq a, l \neq b, l \neq x, k \neq l \]
Refining distinctions

- A distinction is a finite list of *inequalities* 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 \( \sigma \) respects \( e \) if
  - \( \text{supp}(\sigma) \subseteq V \) and \( \sigma \) does not “contradict” \( \prec \)
Refining distinctions

- A distinction is a finite list of inequalities 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 \( \sigma \) respects \( e \) if \( \text{supp}(\sigma) \subseteq V \) and \( \sigma \) does not “contradict” \( \prec \)
- The corresponding distinction is

\[
D(C, V, \prec) \overset{\text{def}}{=} C \neq \cup \{ n \neq x \mid n \in C \land \neg(n \prec x) \}
\]
Some results

We have

\[ P \sim (C, V, \preceq) K Q \Rightarrow P \sim D (C, V, \preceq) O Q \Rightarrow P \sim (C, V, \preceq) K Q \]

In particular

\[ P \sim \emptyset O Q \iff P \sim (\emptyset, fn(P + Q), \emptyset) K Q \]

if \( e \) is an environment, then open \( D(e) \)-bisimilarity is an \( e \)-congruence.
Some results

We have

\[ P \sim_K^{(C, V, \prec)} Q \Rightarrow P \sim_O^{D(C, V, \prec)} Q \]
Some results

We have

\[ P \sim^K_{(C, V, \prec)} Q \Rightarrow P \sim^D_{(C, V, \prec)} Q \]

\[ P \sim^D_{(C, V, \prec)} Q \Rightarrow P \sim^K_{(C, V, \prec)} Q \]
Some results

We have

\[ P \sim_{K}^{(C, V, \prec)} Q \Rightarrow P \sim_{O}^{D(C, V, \prec)} Q \]

\[ P \sim_{O}^{D(C, V, \prec)} Q \Rightarrow P \sim_{K}^{(C, V, \prec)} Q \]

In particular

\[ P \sim_{O}^{\emptyset} Q \Leftrightarrow P \sim_{K}^{(\emptyset, \text{fn}(P+Q), \emptyset)} Q \]
Some results

We have

\[ P \sim_K^{(C, V, \prec)} Q \Rightarrow P \sim_O^{D(C, V, \prec)} Q \]

\[ P \sim_O^{D(C, V, \prec)} Q \Rightarrow P \sim_K^{(C, V, \prec)} Q \]

In particular

\[ P \sim_0 Q \iff P \sim_K^{(\emptyset, \text{fn}(P+Q), \emptyset)} Q \]

if e is an environment, then open \( D(e) \)-bisimilarity is an e-congruence
Outline

1. The pi-calculus
2. Bisimulations
3. The spi-calculus
4. K-open bisimulation
5. Open hedged bisimulation
The intruder knowledge (1/2)

- A hedge $h$ is a finite set of pairs of message
The intruder knowledge (1/2)

- A hedge $h$ is a finite set of pairs of message.
- The synthesis $S(h)$ is the smallest set that contains $h$ and satisfies:

$$(SYN-ENC) \quad \frac{(M, N) \in S(h) \quad (K, L) \in S(h)}{(E_K(M), E_L(N)) \in S(h)}$$
The intruder knowledge (1/2)

- A hedge $h$ is a finite set of pairs of message.
- The synthesis $S(h)$ is the smallest set that contains $h$ and satisfies
  \[(\text{SYN-ENC}) \quad \frac{(M, N) \in S(h) \quad (K, L) \in S(h)}{(E_K(M), E_L(N)) \in S(h)}\]

- For example, if $h = \{(a, a), (k, k)\}$, we have $(E_k(a), E_k(a)) \in S(h)$.
A hedge $h$ is a finite set of pairs of message.

The synthesis $S(h)$ is the smallest set that contains $h$ and satisfies

$$(\text{SYN-ENC}) \quad \frac{(M, N) \in S(h) \quad (K, L) \in S(h)}{(E_K(M), E_L(N)) \in S(h)}$$

For example, if $h = \{(a, a), (k, k)\}$, we have $(E_k(a), E_k(a)) \in S(h)$.

In general, $S(h)$ is not a hedge since it is not finite.
The intruder knowledge (2/2)

- The analysis $A(h)$ is the smallest hedge that contains $h$ and satisfies

$$
(E_K(M), E_L(N)) \in A(h) \quad (K, L) \in S(A(h))
$$

$$(M, N) \in A(h)
$$
The intruder knowledge (2/2)

- The analysis $\mathcal{A}(h)$ is the smallest hedge that contains $h$ and satisfies

$$\frac{(E_K(M), E_L(N)) \in \mathcal{A}(h) \quad (K, L) \in S(\mathcal{A}(h))}{(M, N) \in \mathcal{A}(h)}$$

- For example, if $h = \{(k, k), (E_k(a), E_k(a))\}$, we have $\mathcal{A}(h) = \{(k, k), (E_k(a), E_k(a)), (a, a)\}$. 
The intruder knowledge (2/2)

- The analysis $\mathcal{A}(h)$ is the smallest hedge that contains $h$ and satisfies

$$
\begin{align*}
(\text{ANA-DEC}) & \quad (E_K(M), E_L(N)) \in \mathcal{A}(h) \quad (K, L) \in S(\mathcal{A}(h)) \\
& \quad (M, N) \in \mathcal{A}(h)
\end{align*}
$$

- For example, if $h = \{(k, k), (E_k(a), E_k(a))\}$, we have $\mathcal{A}(h) = \{(k, k), (E_k(a), E_k(a)), (a, a)\}$.

- The irreducibles $\mathcal{I}(h)$ is a “minimal” hedge “equivalent” to $\mathcal{A}(h)$.
A “symmetric” relation $\mathcal{R} \subseteq \mathcal{H} \times \mathcal{P} \times \mathcal{P}$ is a late hedged bisimulation if for all $(h, P, Q) \in \mathcal{R}$,
Late hedged bisimulation

- A “symmetric” relation $\mathcal{R} \subset \mathcal{H} \times \mathcal{P} \times \mathcal{P}$ is a late hedged bisimulation if for all $(h, P, Q) \in \mathcal{R}$,
  - $h$ is consistent
Late hedged bisimulation

- A “symmetric” relation $\mathcal{R} \subset \mathcal{H} \times \mathcal{P} \times \mathcal{P}$ is a late hedged bisimulation if for all $(h, P, Q) \in \mathcal{R}$,
  - $h$ is consistent
  - if $P \xrightarrow{\alpha} P'$
Late hedged bisimulation

A “symmetric” relation $\mathcal{R} \subseteq \mathcal{H} \times \mathcal{P} \times \mathcal{P}$ is a late hedged bisimulation if for all $(h, P, Q) \in \mathcal{R}$,

- $h$ is consistent
- if $P \xrightarrow{\alpha} P'$
  - if $\alpha = \tau$, 
- if $\alpha = a(x)$,
Late hedged bisimulation

A “symmetric” relation $\mathcal{R} \subseteq \mathcal{H} \times \mathcal{P} \times \mathcal{P}$ is a late hedged bisimulation if for all $(h, P, Q) \in \mathcal{R}$,

- $h$ is consistent
- if $P \xrightarrow{\alpha} P'$ then $Q \xrightarrow{\beta} Q'$ and
  - if $\alpha = \tau$,
A “symmetric” relation \( R \subseteq \mathcal{H} \times \mathcal{P} \times \mathcal{P} \) is a late hedged bisimulation if for all \((h, P, Q) \in R\),

- \( h \) is consistent
- if \( P \xrightarrow{\alpha} P' \) then \( Q \xrightarrow{\beta} Q' \) and
  - if \( \alpha = \tau \), then \( \beta = \tau \) and \((h, P', Q') \in R\)
Late hedged bisimulation

A “symmetric” relation $\mathcal{R} \subseteq \mathcal{H} \times \mathcal{P} \times \mathcal{P}$ is a late hedged bisimulation if for all $(h, P, Q) \in \mathcal{R}$,

1. $h$ is consistent
2. if $P \xrightarrow{\alpha} P'$ then $Q \xrightarrow{\beta} Q'$ and
   1. if $\alpha = \tau$, then $\beta = \tau$ and $(h, P', Q') \in \mathcal{R}$
   2. if $\alpha = a(x)$,
Late hedged bisimulation

A “symmetric” relation $\mathcal{R} \subseteq \mathcal{H} \times \mathcal{P} \times \mathcal{P}$ is a late hedged bisimulation if for all $(h, P, Q) \in \mathcal{R}$,

- $h$ is consistent
- if $P \xrightarrow{\alpha} P'$ and $\text{ch}(\alpha) \in \pi_1(h)$ then $Q \xrightarrow{\beta} Q'$ and
  1. if $\alpha = \tau$, then $\beta = \tau$ and $(h, P', Q') \in \mathcal{R}$
  2. if $\alpha = a(x)$,
Late hedged bisimulation

- A “symmetric” relation $\mathcal{R} \subset \mathcal{H} \times \mathcal{P} \times \mathcal{P}$ is a late hedged bisimulation if for all $(h, P, Q) \in \mathcal{R},$
  - $h$ is consistent
  - if $P \xrightarrow{\alpha} P'$ and $\text{ch}(\alpha) \in \pi_1(h)$ then $Q \xrightarrow{\beta} Q'$ and
    1. if $\alpha = \tau$, then $\beta = \tau$ and $(h, P', Q') \in \mathcal{R}$
    2. if $\alpha = a(x)$, then $\beta = b(x)$ and $(a, b) \in S(h)$
Late hedged bisimulation

- A “symmetric” relation $\mathcal{R} \subset \mathcal{H} \times \mathcal{P} \times \mathcal{P}$ is a late hedged bisimulation if for all $(h, P, Q) \in \mathcal{R}$,
  - $h$ is consistent
  - if $P \xrightarrow{\alpha} P'$ and $\text{ch}(\alpha) \in \pi_1(h)$ then $Q \xrightarrow{\beta} Q'$ and
    1. if $\alpha = \tau$, then $\beta = \tau$ and $(h, P', Q') \in \mathcal{R}$
    2. if $\alpha = a(x)$, then $\beta = b(x)$ and $(a, b) \in S(h)$

  for all $(M, N) \in S(h)$, $(h, P'\{M/x\}, Q'\{N/x\}) \in \mathcal{R}$
Late hedged bisimulation

- A “symmetric” relation $\mathcal{R} \subseteq \mathcal{H} \times \mathcal{P} \times \mathcal{P}$ is a late hedged bisimulation if for all $(h, P, Q) \in \mathcal{R}$,
  - $h$ is consistent
  - if $P \xrightarrow{\alpha} P'$ and $\text{ch}(\alpha) \in \pi_1(h)$ then $Q \xrightarrow{\beta} Q'$ and
    1. if $\alpha = \tau$, then $\beta = \tau$ and $(h, P', Q') \in \mathcal{R}$
    2. if $\alpha = a(x)$, then $\beta = b(x)$ and
       $(a, b) \in S(h)$
       for all $B \subseteq \mathcal{N} \times \mathcal{N}$ consistent (and minimal, and fresh)
       for all $(M, N) \in S(h \cup B)$, $(h \cup B, P'\{^M/x\}, Q'\{^N/x\}) \in \mathcal{R}$
Late hedged bisimulation

A “symmetric” relation $\mathcal{R} \subset \mathcal{H} \times \mathcal{P} \times \mathcal{P}$ is a late hedged bisimulation if for all $(h, P, Q) \in \mathcal{R}$,

- $h$ is consistent
- if $P \xrightarrow{\alpha} P'$ and $\text{ch}(\alpha) \in \pi_1(h)$ then $Q \xrightarrow{\beta} Q'$ and
  1. if $\alpha = \tau$, then $\beta = \tau$ and $(h, P', Q') \in \mathcal{R}$
  2. if $\alpha = a(x)$, then $\beta = b(x)$ and $(a, b) \in S(h)$
     for all $B \subset \mathcal{N} \times \mathcal{N}$ consistent (and minimal, and fresh)
     for all $(M, N) \in S(h \cup B)$, $(h \cup B, P'\{^M_x\}, Q'\{^N_x\}) \in \mathcal{R}$
  3. if $\alpha = (\nu \bar{c}) \bar{a} M$,
Late hedged bisimulation

A “symmetric” relation $\mathcal{R} \subset \mathcal{H} \times \mathcal{P} \times \mathcal{P}$ is a late hedged bisimulation if for all $(h, P, Q) \in \mathcal{R}$,

- $h$ is consistent
- if $P \xrightarrow{\alpha} P'$ and $\text{ch}(\alpha) \in \pi_1(h)$ then $Q \xrightarrow{\beta} Q'$ and
  1. if $\alpha = \tau$, then $\beta = \tau$ and $(h, P', Q') \in \mathcal{R}$
  2. if $\alpha = a(x)$, then $\beta = b(x)$ and $(a, b) \in S(h)$
  for all $B \subset \mathcal{N} \times \mathcal{N}$ consistent (and minimal, and fresh)
  for all $(M, N) \in S(h \cup B)$, $(h \cup B, P'\{^M/_x\}, Q'\{^N/_x\}) \in \mathcal{R}$
  3. if $\alpha = (\nu \tilde{c}) \overline{a} M$, then $\beta = (\nu \tilde{d}) \overline{b} N$ and
     $(a, b) \in S(h)$
A “symmetric” relation $\mathcal{R} \subseteq \mathcal{H} \times \mathcal{P} \times \mathcal{P}$ is a late hedged bisimulation if for all $(h, P, Q) \in \mathcal{R}$,

- $h$ is consistent
- if $P \xrightarrow{\alpha} P'$ and $\text{ch}(\alpha) \in \pi_1(h)$ then $Q \xrightarrow{\beta} Q'$ and
  1. if $\alpha = \tau$, then $\beta = \tau$ and $(h, P', Q') \in \mathcal{R}$
  2. if $\alpha = a(x)$, then $\beta = b(x)$ and $(a, b) \in S(h)$ for all $B \subseteq \mathcal{N} \times \mathcal{N}$ consistent (and minimal, and fresh) for all $(M, N) \in S(h \cup B)$, $(h \cup B, P'\{^M/x\}, Q'\{^N/x\}) \in \mathcal{R}$
  3. if $\alpha = (\nu \tilde{c}) \bar{a} M$, then $\beta = (\nu \tilde{d}) \bar{b} N$ and $(a, b) \in S(h)$ $(\mathcal{I}(h \cup \{(M, N)\}), P', Q') \in \mathcal{R}$
From late to open hedged

- An environment is now composed of:
  - A hedge $h$
  - A finite set of pairs of names $v$
  - A precedence relation $\preceq$ to indicate which part of $h$ was available (for each input)
  - Two sets of names $(\gamma_l, \gamma_r)$: type constraints for input names

Moreover, we define:
- The sets of pairs of respectful substitutions $(\sigma, \rho)$
- The consistency of an environment
- The updating of an environment
... and we finally define the bisimulation. The definition obtained is sound.
From late to open hedged

- An environment is now composed of
  - a hedge $h$: the emitted messages

Moreover, we define
- the sets of pair of respectful substitutions $(\sigma, \rho)$
- the consistency of an environment
- the updating of an environment...
... and we finally define the bisimulation.
The definition obtained is sound.
From late to open hedged

- An environment is now composed of
  - a hedge $h$: the emitted messages
  - a finite set of pair of names $\nu$: the input names
From late to open hedged

An environment is now composed of

- a hedge $h$: the emitted messages
- a finite set of pair of names $\nu$: the input names
- $\prec$: precedence relation to indicate which part of $h$ was available (for each input)
From late to open hedged

- An environment is now composed of
  - a hedge $h$: the emitted messages
  - a finite set of pairs of names $v$: the input names
  - $\prec$: precedence relation to indicate which part of $h$ was available (for each input)
  - two sets of names $(\gamma_l, \gamma_r)$: type constraints for input names

Moreover, we define
- the sets of pairs of respectful substitutions $(\sigma, \rho)$
- the consistency of an environment
- the updating of an environment
... and we finally define the bisimulation. The definition obtained is sound.
Environments

\[ P \overset{\text{def}}{=} (\nu k)(\nu m) \bar{a}\langle E_k(m)\rangle . a(x). (\bar{a}\langle k\rangle \parallel [x = k] \bar{a}\langle a\rangle) \]

<table>
<thead>
<tr>
<th>( h^{\frac{1}{2}} )</th>
<th>( \nu^{\frac{1}{2}} )</th>
<th>( \prec^{\frac{1}{2}} )</th>
<th>( \gamma l )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>( { a } )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
</tbody>
</table>
Environments

\[ P \overset{\text{def}}{=} (\nu k) (\nu m) \bar{a} E_k(m) . a(x) . (\bar{a} k \parallel [x = k] \bar{a} a) \]

<table>
<thead>
<tr>
<th>( h^1_2 )</th>
<th>( \nu^1_2 )</th>
<th>( \prec^1_2 )</th>
<th>( \gamma_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>( { a } )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( { E_k(m) } )</td>
<td>( { a } )</td>
<td>( \emptyset )</td>
<td>( { a } )</td>
</tr>
</tbody>
</table>
Environments

\[
P \overset{\text{def}}{=} (\nu k)(\nu m) \overline{a}\langle E_k(m)\rangle . a(x). (\overline{a}\langle k\rangle \parallel [x = k]\overline{a}\langle a\rangle )
\]

<table>
<thead>
<tr>
<th>( h^1 )</th>
<th>( v^1 )</th>
<th>( \prec^1 )</th>
<th>( \gamma_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>{a}</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>{E_k(m)}</td>
<td>{a}</td>
<td>( \emptyset )</td>
<td>{a}</td>
</tr>
<tr>
<td>{E_k(m)}</td>
<td>{a, x}</td>
<td>( E_k(m) \prec x )</td>
<td>{a}</td>
</tr>
</tbody>
</table>
Environments

\[ P \overset{\text{def}}{=} (\nu k)(\nu m) \bar{a}\langle E_k(m)\rangle . a(x). (\bar{a}\langle k \parallel [x = k]\rangle\langle a \rangle) \]

\[
\begin{array}{c|c|c|c}
  h^1_2 & \nu^1_2 & \prec^1_2 & \gamma^1_2 \\
  \emptyset & \{a\} & \emptyset & \emptyset \\
  \{E_k(m)\} & \{a\} & \emptyset & \{a\} \\
  \{E_k(m)\} & \{a, x\} & E_k(m) \prec x & \{a\} \\
\end{array}
\]
Environments

\[
P \overset{\text{def}}{=} (\nu k) (\nu m) \overline{a} \langle E_k(m) \rangle . a(x). (\overline{a} \langle k \rangle \| [x = k] \overline{a} \langle a \rangle )
\]

<table>
<thead>
<tr>
<th></th>
<th>(h^1)</th>
<th>(v^1)</th>
<th>(\prec^1)</th>
<th>(\gamma^1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>{a}</td>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
<td></td>
</tr>
<tr>
<td>{E_k(m)}</td>
<td>{a}</td>
<td>(\emptyset)</td>
<td>{a}</td>
<td></td>
</tr>
<tr>
<td>{E_k(m)}</td>
<td>{a, x}</td>
<td>E_k(m) \prec x</td>
<td>{a}</td>
<td></td>
</tr>
<tr>
<td>{E_k(m), k, m}</td>
<td>{a, x}</td>
<td>E_k(m) \prec x</td>
<td>{a}</td>
<td></td>
</tr>
</tbody>
</table>
Environments

\[ P \overset{\text{def}}{=} (\nu k) (\nu m) \overline{a} \langle E_k(m) \rangle . a(x). (\overline{a} \langle k \rangle \parallel [x = k] \overline{a} \langle a \rangle ) \]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_2^\frac{1}{2} )</td>
<td>( v_2^\frac{1}{2} )</td>
<td>( \prec_2^\frac{1}{2} )</td>
<td>( \gamma )</td>
<td></td>
</tr>
<tr>
<td>( \emptyset )</td>
<td>{a}</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td></td>
</tr>
<tr>
<td>{E_k(m)}</td>
<td>{a}</td>
<td>( \emptyset )</td>
<td>{a}</td>
<td></td>
</tr>
<tr>
<td>{E_k(m)}</td>
<td>{a, x}</td>
<td>( E_k(m) \prec x )</td>
<td>{a}</td>
<td></td>
</tr>
<tr>
<td>{E_k(m), k, m}</td>
<td>{a, x}</td>
<td>( E_k(m) \prec x )</td>
<td>{a}</td>
<td></td>
</tr>
</tbody>
</table>
From late to open hedged

- An environment is now composed of
  - a hedge $h$: the emitted messages messages
  - a finite set of pair of names $\nu$: the input names
  - $\prec$: precedence relation to indicate which part of $h$ was available (for each input)
  - two sets of names $(\gamma_l, \gamma_r)$: type constraints for input names

- Moreover, we define
  - the sets of pair of respectful substitutions $(\sigma, \rho)$
From late to open hedged

- An environment is now composed of
  - a hedge $h$: the emitted messages
  - a finite set of pair of names $\nu$: the input names
  - $\prec$: precedence relation to indicate which part of $h$ was available (for each input)
  - two sets of names $(\gamma_l, \gamma_r)$: type constraints for input names

- Moreover, we define
  - the sets of pair of respectful substitutions $(\sigma, \rho)$
  - the consistency of an environment
From late to open hedged

- An environment is now composed of
  - a hedge $h$: the emitted messages
  - a finite set of pair of names $\nu$: the input names
  - $\prec$: precedence relation to indicate which part of $h$ was available (for each input)
  - two sets of names $(\gamma_l, \gamma_r)$: type constraints for input names

- Moreover, we define
  - the sets of pair of respectful substitutions $(\sigma, \rho)$
  - the consistency of an environment
  - the updating of an environment
From late to open hedged

- An environment is now composed of
  - a hedge $h$: the emitted messages
  - a finite set of pair of names $\nu$: the input names
  - $\prec$: precedence relation to indicate which part of $h$ was available (for each input)
  - two sets of names $(\gamma_l, \gamma_r)$: type constraints for input names

- Moreover, we define
  - the sets of pair of respectful substitutions $(\sigma, \rho)$
  - the consistency of an environment
  - the updating of an environment

- ... and we finally define the bisimulation.
From late to open hedged

An environment is now composed of
- a hedge $h$: the emitted messages
- a finite set of pair of names $v$: the input names
- $\prec$: precedence relation to indicate which part of $h$ was available (for each input)
- two sets of names $(\gamma_l, \gamma_r)$: type constraints for input names

Moreover, we define
- the sets of pair of respectful substitutions $(\sigma, \rho)$
- the consistency of an environment
- the updating of an environment

... and we finally define the bisimulation.

The definition obtained is sound.
Conclusion
Conclusion

- Definition of K-open bisimulation
Conclusion

- Definition of K-open bisimulation
  - Coincides with open bisimulation
Conclusion

- Definition of K-open bisimulation
  - Coincides with open bisimulation
  - Defined of bigger set of contexts that preserves open
Conclusion

- Definition of K-open bisimulation
  - Coincides with open bisimulation
  - Defined of bigger set of contexts that preserves open
- Open hedged bisimulation
Conclusion and future work

Conclusion

- Definition of K-open bisimulation
  - Coincides with open bisimulation
  - Defined of bigger set of contexts that preserves open
- Open hedged bisimulation
  - Sound w.r.t. late hedged bisimulation
Future work
Future work

- Study open hedged bisimulation
Future work

- Study open hedged bisimulation
  - Link with symbolic bisimulation of [BBN04]
Future work

- Study open hedged bisimulation
  - Link with symbolic bisimulation of [BBN04]
  - Congruence properties?
Thank you!
Thank you!
Questions?
Appendix

Bibliography

- D. Sangiorgi
  *A Theory of Bisimulation for the $\pi$-calculus.*

- J. Borgström, S. Briais and U. Nestmann
  *Symbolic Bisimulations in the Spi Calculus*
e-respectful contexts

If \( e = (O, V, \prec) \), a context \( C[\cdot] \) respects \( e \) if it is generated by

\[
C_N[\cdot] ::= \begin{align*}
\& P \mid C_N[\cdot] \mid C_N[\cdot] \mid P \\
\& P + C_N[\cdot] \mid C_N[\cdot] + P \\
\& ! C_N[\cdot] \\
\& \phi C_N[\cdot] \\
(\nu x) C_{N\setminus\{x\}}[\cdot] \\
\overline{a}(z).C_N[\cdot] \\
a(x).C_N[\cdot] & \quad \text{if } x \not\in O \cup V \\
a(x).C_{N\cup N'}[\cdot] & \quad \text{if } x \in V \text{ and } N' = \{n \in O \mid \neg n \prec x\}
\end{align*}
\]

with \( N \subset O \) and \( C_\emptyset[\cdot] \) as start symbol.