A Logical Framework for the spi-calculus

Emilio Tuosto
Ugo Montanari

Dipartimento di Informatica, Università di Pisa

Overview

spi:
- Syntax
- Example
- Semantics

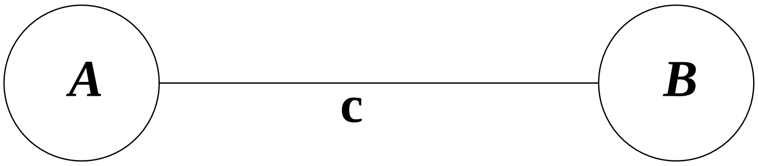
LF:
- LF introduction
- spi syntax in LF
- spi semantics in LF

Adequacy

Spi syntax

M, N	::=		terms
	n		channels
	k		keys (are names)
	$\{M\}_k$		encryption
p, q, r	::=		processes
	$\overline{n}\langle M\rangle.p$		output
	$n(x).p$		input
	$p \mid q$		composition
	$(\nu\, n)p$		restriction
	$!p$		bang
	$[M = N]p$		match
	0		nil
	case M of $\{x\}_k$ in p		decryption
a	::=		agents
	p		process
	$\lambda x.p$		abstraction
	$(\nu\, \vec{m})\langle M\rangle p$		concretion

A (trivial) protocol example



$$A(M) \stackrel{\triangle}{=} \overline{c}\langle \{M\}_k \rangle$$

$$B \stackrel{\triangle}{=} c(x).\mathsf{case}\ x\ \mathsf{of}\ \{y\}_k\ \mathsf{in}\ \overline{m}\langle y \rangle$$

$$\mathit{Inst}(M) \stackrel{\triangle}{=} (\nu\, k)(A(M) \mid B)$$

- LTS + structural laws + reduction relation

- LTS without structural laws & reduction relation

- Equivalence

Spi semantics 2

labels $\alpha ::= \tau \mid m \mid \overline{m}$

$$\begin{array}{rcll} (\nu\, m)\lambda x.p & \stackrel{\triangle}{=} & \lambda x.(\nu\, m)p, & x \neq m \\ r \mid \lambda x.p & \stackrel{\triangle}{=} & \lambda x.(r \mid p), & x \notin fn(r) \end{array}$$

Abstractions structural laws

$$\begin{array}{rcl} r \mid (\nu\, \vec{n})\langle M\rangle q & \stackrel{\triangle}{=} & (\nu\, \vec{n})\langle M\rangle(r \mid q), \quad \vec{n} \cap fn(r) = \emptyset \\ (\nu\, m)(\nu\, \vec{n})\langle M\rangle q & \stackrel{\triangle}{=} & \begin{cases} (\nu\, m, \vec{n})\langle M\rangle q, & m \in n(M) \setminus \{\vec{n}\} \\ (\nu\, \vec{n})\langle M\rangle(\nu\, m)q, & m \notin n(M) \cup \{\vec{n}\} \end{cases} \end{array}$$

Concretions structural laws

$$F \equiv \lambda x.p \qquad C \equiv (\nu\, \vec{m})\langle M\rangle q \qquad \{\vec{m}\} \cap fn(p)\emptyset$$

$$\begin{array}{rcl} F@C & \stackrel{\text{def}}{=} & (\nu\, \vec{m})(p[M/x] \mid q) \\ C@F & \stackrel{\text{def}}{=} & (\nu\, \vec{m})(q \mid p[M/x]) \end{array}$$

Agent interactions

$$\begin{array}{ccc} !p & \succ & p \mid !p \\ [M = M]p & \succ & p \\ \text{case } \{M\}_k \text{ of } \{x\}_k \text{ in } p & \succ & p[M/x] \end{array}$$

Reduction relation

Spi semantics 2.1

(IN) $$\frac{}{m(x).p \xrightarrow{m} \lambda x.p} \qquad \frac{}{\overline{m}\langle M\rangle.p \xrightarrow{\overline{m}} \langle M\rangle p}$$ (OUT)

(INTER1) $$\frac{p \xrightarrow{m} F \qquad q \xrightarrow{\overline{m}} C}{p \mid q \xrightarrow{\tau} F@C} \qquad \frac{p \xrightarrow{\overline{m}} C \qquad q \xrightarrow{m} F}{p \mid q \xrightarrow{\tau} C@F}$$ (INTER2)

(PAR1) $$\frac{p \xrightarrow{\alpha} a}{p \mid q \xrightarrow{\alpha} a \mid q} \qquad \frac{q \xrightarrow{\alpha} a}{p \mid q \xrightarrow{\alpha} p \mid a}$$ (PAR2)

(RES) $$\frac{p \xrightarrow{\alpha} a \qquad \alpha \notin \{m, \overline{m}\}}{(\nu\, m)p \xrightarrow{\alpha} (\nu\, m)a} \qquad \frac{p \succ q \qquad q \xrightarrow{\alpha} a}{p \xrightarrow{\alpha} a}$$ (RED)

IN
$$\frac{}{m(x).p \xrightarrow{m} \lambda x.p}$$

OUT
$$\frac{}{\overline{m}\langle M\rangle.p \xrightarrow{\overline{m}} \langle M\rangle p}$$

COM^l
$$\frac{p \xrightarrow{m} \lambda x.p' \qquad q \xrightarrow{\overline{m}} \langle M\rangle p}{p \mid q \xrightarrow{\tau} p'[M/x] \mid q'}$$

OPEN
$$\frac{p \xrightarrow{\overline{m}} \langle M\rangle q \qquad m \notin \vec{n} \qquad \vec{n} \cap fn(M) \neq \emptyset}{(\nu\,\vec{n})p \xrightarrow{\overline{m}} (\nu\,\vec{n})(\langle M\rangle q)}$$

RES$_1$
$$\frac{p \xrightarrow{m} \lambda x.r \qquad x, m \notin \vec{m}}{(\nu\,\vec{m})p \xrightarrow{m} \lambda x.(\nu\,\vec{m})r}$$

RES$_3$
$$\frac{p \xmapsto{\overline{m}} \langle M\rangle q \qquad \vec{m} \cap (fn(M) \cup \{m\}) = \emptyset}{(\nu\,\vec{m})p \xrightarrow{\overline{m}} \langle M\rangle(\nu\,\vec{m})q}$$

RES$_2$
$$\frac{p \xrightarrow{\tau} a}{(\nu\,\vec{m})p \xrightarrow{\tau} (\nu\,\vec{m})a}$$

CLOSE
$$\frac{p \xrightarrow{m} \lambda x.p' \qquad q \xrightarrow{\overline{m}} (\nu\,\vec{n})(\langle M\rangle q') \qquad \vec{n} \cap fn(p) = \emptyset}{p \mid q \xrightarrow{\tau} (\nu\,\vec{n})(p'[M/x] \mid q)}$$

$$\text{MATCH}\quad \frac{p \xrightarrow{\alpha} a}{[M = M]p \xrightarrow{\alpha} a} \qquad\qquad \frac{p \mid !p \xrightarrow{\alpha} a}{!p \xrightarrow{\alpha} a}\quad \text{BANG}$$

$$\text{PAR}^{l}_{1}\quad \frac{p \xrightarrow{\overline{m}} \langle M \rangle r}{p \mid q \xrightarrow{\overline{m}} \langle M \rangle (r \mid q)} \qquad \frac{p \xrightarrow{\overline{m}} (\nu\, \vec{n})(\langle M \rangle r) \quad \vec{n} \cap fn(q) = \emptyset \quad \vec{n} \cap fn(M) \neq \emptyset}{p \mid q \xrightarrow{\overline{m}} (\nu\, \vec{n})(\langle M \rangle (r \mid q))}\quad \text{PAR}^{l}_{2}$$

$$\text{PAR}^{l}_{3}\quad \frac{p \xrightarrow{m} \lambda x.r \qquad x \notin fn(q)}{p \mid q \xrightarrow{m} \lambda x.(r \mid q)} \qquad\qquad \frac{p \xrightarrow{\tau} r}{p \mid q \xrightarrow{\tau} r \mid q}\quad \text{PAR}^{l}_{4}$$

$$\text{DECR}\quad \frac{p[M/x] \xrightarrow{\alpha} a}{\text{case } \{M\}_k \text{ of } \{x\}_k \text{ in } p \xrightarrow{\alpha} a}$$

Lemma 1 $p \xrightarrow[\mathsf{ag}]{m} a \Rightarrow \exists r : a \stackrel{\triangle}{=} \lambda x.r$

Lemma 2 $p \xrightarrow[\mathsf{ag}]{\overline{m}} a \Rightarrow \exists \vec{n}, M, q : a \stackrel{\triangle}{=} (\nu\, \vec{n})\langle M \rangle q$

$$\Downarrow$$

Proposition $p \xrightarrow[\mathsf{ag}]{\alpha} a \Leftrightarrow p \xrightarrow{\alpha} a'$ for an agent $a' \stackrel{\triangle}{=} a$

$$(\xrightarrow[\mathsf{ag}]{\bullet} \;=\; \xrightarrow{\bullet} \stackrel{\triangle}{=})$$

Principles

- Formulas - as - types
- judgments - as - types
- proofs - as - terms

Tools

- Proof Editors
- Proof Checkers

Benefits

- no α-conversion
- no side-conditions
- all binders $\leadsto$ λ-abstraction
- communication $\leadsto$ β-reduction
- proof assistant

Syntax

Objects	M	$::= x \mid MN \mid \lambda x : A.M$
Types	A	$::= X \mid AM \mid \prod_{x:A}.B \mid \lambda x : A.B$
Kinds	K	$::=$ Type $\mid \prod_{x:A}.K$

Environments $\Gamma \quad ::= \quad \langle\rangle \mid \Gamma, x : A \mid \Gamma, X : K$

Assertions

$\vdash \Gamma \quad \equiv$ "Γ is a valid environment"

$\Gamma \vdash K \quad \equiv$ "K is a kind in Γ"

$\Gamma \vdash A : K \quad \equiv$ "A has kind K in Γ"

$\Gamma \vdash M : A \quad \equiv$ "M has type A in Γ"

Environments

$$\frac{}{\vdash \langle\rangle} \qquad \frac{\vdash \Gamma \qquad \Gamma \vdash K \qquad X \notin \mathit{dom}(\Gamma)}{\vdash \Gamma, X : K}$$

$$\frac{\vdash \Gamma \qquad \Gamma \vdash A : \mathsf{Type} \qquad x \notin \mathit{dom}(\Gamma)}{\vdash \Gamma, x : A}$$

Kinds

$$\frac{\vdash \Gamma}{\Gamma \vdash \mathsf{Type}} \qquad \frac{\Gamma, x : A \vdash K}{\Gamma \vdash \prod_{x:A}.K}$$

Types

$$\frac{\vdash \Gamma \qquad X : K \in \Gamma}{\Gamma \vdash X : K} \qquad \frac{\Gamma, x : A \vdash B : \mathsf{Type}}{\Gamma \vdash \prod_{x:A}.B : \mathsf{Type}}$$

$$\frac{\Gamma, x : A \vdash B : K}{\Gamma \vdash \lambda x : A.B : \prod_{x:A}.K} \qquad \frac{\Gamma \vdash A : \prod_{x:B} K \qquad \Gamma \vdash M : B}{\Gamma \vdash AM : K[M/x]}$$

$$\frac{\Gamma \vdash A : K \qquad \Gamma \vdash K' \qquad \Gamma \vdash K \equiv K'}{\Gamma \vdash A : K'}$$

Objects

$$\frac{\vdash \Gamma \qquad x : A \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : \prod_{x:A}.B}$$

$$\frac{\Gamma \vdash M : \prod_{x:A}.B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]} \qquad \frac{\Gamma \vdash M : \prod_{x:A}.B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]}$$

Translation scheme

- Syntactic categories $\rightsquigarrow$ Types
- Syntactic constructors $\rightsquigarrow$ types constructor with functional types
- Object language variables $\rightsquigarrow$ LF variables
- Transitions $\rightsquigarrow$ judgments
  - (new) types for transitions
  - LTS rules $\rightsquigarrow$ constructor for transition types

Translating the example

$$[\![A(M)]\!] = \textcolor{blue}{\mathrm{out}}(\delta_e, \delta_e, [\![c]\!], [\![\{M\}_k]\!])$$

$$[\![B]\!] = \textcolor{blue}{\mathrm{in}}(\delta_e, \delta_e, [\![c]\!], \lambda x : \mathcal{E}.\textcolor{blue}{\mathrm{decr}}(\delta_n, x, [\![k]\!], \lambda y : \mathcal{N}_n.\textcolor{blue}{\mathrm{out}}(\delta_n, \delta_n, [\![m]\!], y, \textcolor{blue}{\mathrm{nil}})))$$

$$[\![\textit{Inst}(M)]\!] = \nu(\lambda k' : \mathcal{N}_n.\textcolor{blue}{\mathrm{proc}}([\![A(M)]\!] \mid [\![B]\!][k'/[\![k]\!]]))$$

A LF for spi syntax: Types

$\mathcal{L}$	:	Type	$\mathcal{P}$	:	Type
$\mathcal{N}_n$	:	Type	$\mathcal{A}$	:	Type
$\mathcal{N}_e$	:	Type	$\mathcal{E}$	:	Type
D	:	Type	$\aleph$	:	Type

$$\mathcal{N}_l^{(z)} : \text{Type}$$

$$\mathcal{N}(d) = \begin{cases} \mathcal{N}_n, & \text{if } d = \delta_n \\ \mathcal{N}_e, & \text{if } d = \delta_e \\ \prod_{z:\aleph}.\mathcal{N}_l{}^{(z)}, & \text{if } d = \delta_l \end{cases}$$

$$\mathcal{T}(d, d') = \begin{cases} \mathcal{E}, & \text{if } d = \delta_e \\ \mathcal{N}(d'), & \text{if } d = \delta_n \text{ and } d' \neq \delta_l \\ \prod_{z:\aleph}.\mathcal{N}_l{}^{(z)}, & \text{if } d = \delta_l \end{cases}$$

A LF for spi syntax: Syntactic Constructors

tau	:	$\mathcal{L}$	
get	:	$\prod_{d:\mathbf{D}}.\mathcal{N}(d) \longrightarrow \mathcal{L}$	*Labels*
put	:	$\prod_{d:\mathbf{D}}.\mathcal{N}(d) \longrightarrow \mathcal{L}$	
out	:	$\prod_{d,d':\mathbf{D}}.\mathcal{N}(d) \longrightarrow \mathcal{T}(d,d') \longrightarrow \mathcal{P} \longrightarrow \mathcal{P}$	
in	:	$\prod_{d,d':\mathbf{D}}.\mathcal{N}(d) \longrightarrow (\mathcal{T}(d,d') \longrightarrow \mathcal{P}) \longrightarrow \mathcal{P}$	
$\mid$	:	$\mathcal{P} \longrightarrow \mathcal{P} \longrightarrow \mathcal{P}$	
ν	:	$(\mathcal{N}_l \longrightarrow \mathcal{A}) \longrightarrow \mathcal{P}$	
!	:	$\mathcal{P} \longrightarrow \mathcal{P}$	*Processes*
$[\_ = \_]$	:	$\prod_{d,d':\mathbf{D}}.\mathcal{T}(d,d') \longrightarrow \mathcal{T}(d,d') \longrightarrow \mathcal{P} \longrightarrow \mathcal{P}$	
nil	:	$\mathcal{P}$	
decr	:	$\prod_{d:\mathbf{D}}.\mathcal{E} \longrightarrow \mathcal{N}_n \longrightarrow (\mathcal{T}(d,\delta_n) \longrightarrow \mathcal{P}) \longrightarrow \mathcal{P}$	

$$
\begin{array}{rcl}
[\![n(x).p]\!] & = & \mathrm{in}(\hat{n}, \hat{x}, [\![n]\!], \lambda x : \mathcal{T}(\hat{n}, \hat{x}).[\![p]\!][x/[\![x]\!]]) \\
[\![\overline{n}\langle M\rangle.p]\!] & = & \mathrm{out}(\hat{n}, \widehat{M}, [\![n]\!], [\![M]\!], [\![p]\!]) \quad \text{if } \hat{n} = \widehat{M} \\
[\![0]\!] & = & \mathrm{nil} \\
[\![p \mid q]\!] & = & [\![p]\!] \mid [\![q]\!] \\
[\![!p]\!] & = & ![\![p]\!] \\
[\![(\nu\, n_1)\ldots(\nu\, n_k)p]\!] & = & \nu\,(\lambda l : \mathcal{N}_l{}^{(k)}.\mathrm{proc}([\![p]\!]\sigma)) \\
 & & \text{where } \sigma = [\mathrm{hd}_{\hat{n}_i}^{k-i+1}(\mathrm{tl}^{k-i+1}(l_i))/[\![n_i]\!]]_{i=1,\ldots,k} \\
 & & \qquad l_1 = l \quad l_{i+1} = \mathrm{tl}^{i+1}(l_i) \quad i = 1,\ldots,k-1 \\
[\![[M = N]p]\!] & = & \begin{cases} [\,[\![M]\!] = [\![N]\!]\,]^{\widehat{M},\widehat{M}}[\![p]\!] & \text{if } \widehat{M} = \delta_n \\ [\,[\![M]\!] = [\![N]\!]\,]^{\delta_e,\delta_e}[\![p]\!] & \text{if } \widehat{M} = \delta_e \end{cases} \\
[\![\text{case } M \text{ of } \{x\}_k \text{ in } p]\!] & = & \mathrm{decr}(\hat{x},[\![M]\!],[\![k]\!],\lambda x : \mathcal{T}(\hat{x}, \delta_n).[\![p]\!][x/[\![x]\!]])
\end{array}
$$

$$\longmapsto \quad : \quad \mathcal{P} \longrightarrow \mathcal{L} \longrightarrow \mathcal{A} \longrightarrow \mathsf{Type}$$

$$\longmapsto\!\!\!\!\rightarrow \quad : \quad \prod_{d,d':\mathbf{D}}.\mathcal{P} \longrightarrow \mathcal{L} \longrightarrow (\mathcal{T}(d,d') \longrightarrow \mathcal{A}) \longrightarrow \mathsf{Type}$$

A LF for spi semantics 2

IN	:	$\prod_{d,d':\mathsf{D}}.\ \prod_{m:\mathcal{N}(d)}.\ \prod_{p:\mathcal{T}(d,d')\longrightarrow\mathcal{P}}.\ \mathrm{in}(d,d',m,p) \overset{m}{\underset{d,\,d'}{\longmapsto\!\!\!\!\rightarrow}} p$
OUT	:	$\prod_{d,d':\mathsf{D}}.\ \prod_{m:\mathcal{N}(d)}.\ \prod_{t:\mathcal{T}(d,d')}.\ \prod_{p:\mathcal{P}}.$ $\mathrm{out}(d,d',m,t,p) \overset{\overline{m}}{\longmapsto} \mathrm{conc}(d,d',t,p)$
COM^l	:	$\prod_{d,d':\mathsf{D}}.\ \prod_{m:\mathcal{N}(d)}.\ \prod_{p,q,q':\mathcal{P}}.\ \prod_{p':\mathcal{T}(d,d')\longrightarrow\mathcal{P}}.$ $\prod_{t:\mathcal{T}(d,d')}.\ p \overset{m}{\underset{d,\,d'}{\longmapsto\!\!\!\!\rightarrow}} \lambda t' : \mathcal{T}(d,d').\mathrm{proc}(p't') \longrightarrow$ $q \overset{\overline{m}}{\longmapsto} \mathrm{conc}(d,d',t,q') \longrightarrow (p \mid q) \overset{\tau}{\longmapsto} \mathrm{proc}(p't \mid q')$
....	:	....

Let p a spi-calculus process

$$p \xrightarrow{\alpha} a$$

iff OP$_{\alpha}$(...$[\![p]\!]$, $[\![\alpha]\!]$, $[\![a]\!]$...)

is inhabited

Proposition

- if $(\widehat{m} = \delta_n \quad \wedge \quad \breve{m} = \widehat{x})$ or $\widehat{m} = \widehat{x} = \delta_e$ then $p \xrightarrow{m} \lambda x.q$

  $$\Longleftrightarrow$$

  $[\![p]\!] \underset{\widehat{m},\, \widehat{x}}{\overset{m}{\longmapsto\!\!\!\!\rightarrow}} \lambda x : \mathcal{T}(\widehat{n}, \widehat{x}).\mathrm{proc}([\![q]\!][x/\mathrm{Tch}(\widehat{x}, \breve{x}, x)])$ is inhabited;

- if $\widehat{m} = \widehat{M}$ then $p \xrightarrow{\overline{m}} \langle M \rangle q$

  $$\Longleftrightarrow$$

  $[\![p]\!] \overset{\overline{m}}{\longmapsto} \mathrm{conc}(\widehat{m}, \widehat{M}, [\![M]\!], [\![q]\!])$ is inhabited;

- $p \xrightarrow{\overline{m}} (\nu\, n_1) \ldots (\nu\, n_k) \langle M \rangle q$

  $$\Longleftrightarrow$$

  $[\![p]\!] \underset{\delta_l,\, \delta_l}{\overset{\overline{m}}{\longmapsto\!\!\!\!\rightarrow}} \lambda l : \mathcal{N}_l{}^{(k)}.\mathrm{conc}(\delta_l, \delta_l, [\![M]\!], [\![q]\!])\sigma$ is inhabited

  where $\quad l_1 = l \qquad l_{i+1} = \mathrm{tl}^{k-i+1}(l_i) \qquad i = 1, \ldots, k-1$

  $$\sigma = [\mathrm{hd}_{\widehat{n}_i}^{k-i+1}(\mathrm{tl}^{k-i+1}(l_i))/\mathrm{Tch}(\widehat{n}_i, \breve{n}_i, n_i)]_{i=1,\ldots,k}$$

- if $p \xrightarrow{\tau} q \quad \Longleftrightarrow \quad [\![p]\!] \overset{\tau}{\longmapsto} [\![q]\!]$ is inhabited.

- LF is a metalanguage
  - Uniformity in treating α-renaming & binders
  - Communications $\rightsquigarrow$ β-reduction
  - Tools independent from the object language
- Limitations
  - No subsorting allowed (for example, processes $\subseteq$ agents...)
  - Casting necessary
- Future Work: LF and security
  - Security property as reachability
  - Formalization of reachability property in LF
  - Tests with LF