Non-Functional Aspects of Wide Area Network Programming

Emilio Tuosto

Ph.D. Thesis

Dipartimento di Informatica
Università di Pisa
Plan of the talk

WAN programming: A short overview

Declarative programming model: Hypergraphs

Hypergraphs and Ambient calculus

Programming QoS: Qlaim

QoS & Hypergraphs: reasoning on optimal routing

Hypergraphs and UML specifications

Cryptographic protocols: cIP and PL

The Mihda environment
Plan of the talk

- WAN programming: A short overview

- Declarative programming model: Hypergraphs
  - Hypergraphs and Ambient calculus
  - Programming QoS: Qlaim
  - QoS & Hypergraphs: reasoning on optimal routing

- Hypergraphs and UML specifications

- Cryptographic protocols: cIP and PL

- The Mihda environment
Plan of the talk

- WAN programming: A short overview
- Declarative programming model: Hypergraphs
  - Hypergraphs and Ambient calculus
- Programming QoS: Qlaim
- QoS & Hypergraphs: reasoning on optimal routing
- Hypergraphs and UML specifications
- Cryptographic protocols:
  - cIP
  - PL
- The Mihda environment
Plan of the talk

- WAN programming: A short overview
- Declarative programming model: Hypergraphs
  - Hypergraphs and Ambient calculus
- Programming QoS: Qlaim
  - QoS & Hypergraphs: reasoning on optimal routing
Plan of the talk

- WAN programming: A short overview
- Declarative programming model: Hypergraphs
  - Hypergraphs and Ambient calculus
- Programming QoS: Qlaim
  - QoS & Hypergraphs: reasoning on optimal routing
- Hypergraphs and UML specifications
Plan of the talk

- WAN programming: A short overview
- Declarative programming model: Hypergraphs
  - Hypergraphs and Ambient calculus
- Programming QoS: Qlaim
  - QoS & Hypergraphs: reasoning on optimal routing
- Hypergraphs and UML specifications
- Cryptographic protocols: clP and PL
Plan of the talk

- WAN programming: A short overview
- Declarative programming model: Hypergraphs
  - Hypergraphs and Ambient calculus
- Programming QoS: Qlaim
  - QoS & Hypergraphs: reasoning on optimal routing
- Hypergraphs and UML specifications
- Cryptographic protocols: cIP and PL
  - The Mihda environment
Wide Area Network Programming Issues

- Absence of centralised control
- Administrative domains
- Interoperability
- “Mobility” (of resources and computation)
- Network Awareness
- Service Level Agreement
- Security
- ...
Web Services: A programming metaphor

- Applications access *services* that must be
  - Published
  - Searched
  - Binded

- Services are
  - “Autonomous”
  - Independent (local choices, independently built)
  - Mobile/stationary
  - “Interconnected”

- Security issues: hostile environment
$\pi$-calculus [MPW92] (very basic wrt WAN)

- Klaim [DFP98, DFPV00, BLP02]
- Ambient [CG00]
- $D_\pi$ [HR98, HR00]
- Djoin [FG96, FGL+96]
- Seal [VC98]
- ...
A Model for Declarative WAN Programming
Hypergraphs Programming model

Client-Server metaphor is not enough:

- P2P
- Mobility and dynamic linking of components
- Adaptability to different devices (e.g. PDA, laptop, mobile phones...)
- Location awareness
- Applications are location dependent
- Locations have different features and allow multiple (security) policies
- Independently programmed in a distributed environment
- Reasoning on space and time
Client-Server metaphor is not enough: P2P
Hypergraphs Programming model

- Client-Server metaphor is not enough: P2P
- Mobility and dynamic linking of components
Hypergraphs Programming model

- Client-Server metaphor is not enough: P2P
- Mobility and dynamic linking of components
- Adaptability to different devices (e.g. PDA, laptop, mobile phones...)
Hypergraphs Programming model

- Client-Server metaphor is not enough: P2P
- Mobility and dynamic linking of components
- Adaptability to different devices (e.g. PDA, laptop, mobile phones...)
- Location awareness
  - Applications are location dependent
  - Locations have different features
  - and allow multiple (security) policies
Hypergraphs Programming model

- Client-Server metaphor is not enough: P2P
- Mobility and dynamic linking of components
- Adaptability to different devices (e.g. PDA, laptop, mobile phones...)
- Location awareness
  - Applications are location dependent
  - Locations have different features
  - and allow multiple (security) policies
- Independently programmed in a distributed environment
Hypergraphs Programming model

- Client-Server metaphor is not enough: P2P
- Mobility and dynamic linking of components
- Adaptability to different devices (e.g. PDA, laptop, mobile phones...)
- Location awareness
  - Applications are location dependent
  - Locations have different features
  - and allow multiple (security) policies
- Independently programmed in a distributed environment
- Reasoning on space and time
Hypergraphs Programming model

- Graphs for distributed systems [CM83]
- Edge replacement for graph rewritings [DM87]
- Edge replacement/distributed constraint solving problem [MR96]
- Graphs grammars for software architecture styles [HIM00]
- Synchronised Hyperedge Replacement (SHR) with mobility for name passing calculi [HM01]
We aim at tackling new *non-functional* computational phenomena of systems using SHR. The metaphor is

- “WAN systems *as* Hypergraphs”
- “WAN computations *as* SHR”

In other words:

- Components are represented by hyperedges
- Systems are *bunches* of (connected) hyperedges
- Computing means to rewrite hyperedge...
- ...according to a synchronisation policy
A hyperedge generalises edges: It connects more than two nodes

$L : 3, \quad L(y, z, x)$,
Hyperedges and Hypergraphs Syntax

A hyperedge generalises edges: It connects more than two nodes

\[ L : 3, \quad L(y, z, x), \]

\[ G ::= \text{nil} \mid \nu y. G \]
\[ \mid L(\bar{x}) \mid G|G \]
Hyperedges and Hypergraphs Syntax

A hyperedge generalises edges: It connects more than two nodes

\[ L : 3, \; L(y, z, x), \]

\[
\begin{array}{c}
  \bullet \quad y \\
  \quad \downarrow \quad 1 \\
  x \quad \bullet \quad 3 \quad L \quad 2 \quad \bullet \quad z \\
\end{array}
\]

\[
G ::= \text{nil} \mid \nu y.G \\
| \quad L(\bar{x}) \mid G|G
\]

Syntactic Judgement \( \Gamma \vdash G, \) \( fn(G) \subseteq \Gamma \)
Hyperedges and Hypergraphs Syntax

A hyperedge generalises edges: It connects more than two nodes

\[ L : 3, \quad L(y, z, x), \]

\[ G ::= \text{nil} \mid \nu y.G \mid L(\overline{x}) \mid G|G \]

**Syntactic Judgement**

\[ \Gamma \vdash G, \quad fn(G) \subseteq \Gamma \]

An example:

\[ L : 3, \quad M : 2 \]

\[ x, y \vdash \nu z.(L(y, z, x) | M(y, z)) \]
Hyperedges and Hypergraphs Syntax

A hyperedge generalises edges: It connects more than two nodes

\[ L : 3, \quad L(y, z, x), \]

Syntactic Judgement

\[ \Gamma \vdash G, \quad fn(G') \subseteq \Gamma \]

An example:

\[ L : 3, \quad M : 2 \]
\[ x, y \vdash \nu z.(L(y, z, x) | M(y, z)) \]
Replacement of Hyperedges

$L \rightarrow G$
Replacement of Hyperedges

$L \rightarrow G$

Diagram: A labeled graph with nodes and edges illustrating the replacement of hyperedges.
Replacement of Hyperedges

$L \rightarrow G$

\[
\begin{array}{c}
\text{G1} \\
1 \quad 2 \quad 3 \\
\text{G} \\
5 \\
\text{G2'} \\
4
\end{array}
\]
Replacement of Hyperedges

$L \rightarrow G$

- Edge replacement: local
- Synchronisation as distributed constraint solving
- New node creation
- Node fusion: mobility model
Replacement of Hyperedges

\[ L \rightarrow G \]

- Edge replacement: local
- Synchronisation as distributed constraint solving
- New node creation
- Node fusion: mobility model

Benefits:
- Powerful model of system composition (\( \pi, \pi\text{-l}, \text{fusion} \))
- LTS for Ambient ...
- ...and for Klaim
- and path reservation for Qlaim
Hypergraph Semantics: Productions

\[ x_1, \ldots, x_n \vdash L(x_1, \ldots, x_n) \xrightarrow{\Lambda} \Gamma \vdash G, \]

- \( \Lambda \subseteq X \times Act \times \mathcal{N}^* \) set of constraints
- \( \pi : X \rightarrow X \) fusion substitution, i.e.
  \[ \forall x_i, x_j \in X. \pi(x_i) = x_j \Rightarrow \pi(x_j) = x_j \]
- \( \Gamma = \pi(X) \cup (n(\Lambda) \setminus X) \)
- \( \text{fn}(G) \subseteq \Gamma \)
Hypergraph Semantics: Transitions

\[
\Gamma_1 \vdash G_1 \xrightarrow{\Lambda} \Gamma_2 \vdash G_2
\]
Hypergraph Semantics: Transitions

\[ \Gamma, y \vdash G \frac{\Lambda}{\pi} \Rightarrow \Gamma' \vdash G' \]

\[ \Lambda(y) \uparrow \quad x \simeq_{\pi} y \Rightarrow y \neq \pi(y) \]

\[ \rho = [\pi(x) / \pi(y)] \]

\[ \Gamma \vdash [x / y]G \frac{\rho \Lambda}{(\pi; \rho) - y} \Rightarrow n(\rho \Lambda) \cup (\pi; \rho) - y(\Gamma) \vdash \rho G' \]

\[ \Gamma, y \vdash G \frac{\Lambda \cup \{(x, a, \bar{v}), (y, \bar{a}, \bar{w})\}}{\pi} \Rightarrow \Gamma' \vdash G' \]

\[ x \simeq_{\pi} y \Rightarrow y \neq \pi(y) \quad \rho = \text{mgu}\{[x / y]\bar{v} / [x / y]\bar{v}, [\pi(x) / \pi(y)]\} \]

\[ \Gamma'' = n(\rho \Lambda) \cup (\pi; \rho) - y(\Gamma) \quad U = \rho(\Gamma') \setminus \Gamma'' \]

\[ \Gamma \vdash [x / y]G \frac{(\rho \Lambda \cup (x, \tau, \langle \rangle))}{(\pi; \rho) - y} \Rightarrow \Gamma'' \vdash \nu U. \rho G'' \]
Hypergraph Semantics: Transitions

\[ \Gamma, y \vdash G \xrightarrow{\Lambda} \Gamma' \vdash G' \]

\[ \Lambda(y) \uparrow \forall \Lambda(y) = (\tau, \langle \rangle) \quad x \simeq_\pi y \Rightarrow y \neq \pi(y) \]

\[ U = \Gamma' \setminus (n(\Lambda) \cup \pi_{-y}(\Gamma)) \]

\[ \Gamma \vdash \nu y. G \xrightarrow{\Lambda \setminus (y, \tau, \langle \rangle)} \pi_{-y} n(\Lambda) \cup \pi_{-y}(\Gamma) \vdash \nu U. G' \]

\[ \Gamma_1 \vdash G_1 \xrightarrow{\Lambda} \Gamma_2 \vdash G_2 \quad \Gamma_1' \vdash G_1' \xrightarrow{\Lambda'} \Gamma_2' \vdash G_2' \quad \Gamma_1 \cap \Gamma_1' = \emptyset \]

\[ \Gamma_1 \cup \Gamma_1' \vdash G_1 | G_1' \xrightarrow{\Lambda \cup \Lambda'} \pi \cup \pi' \Gamma_2 \cup \Gamma_2' \vdash G_2 | G_2' \]
Applying the Model

Ambient \[ a[...]|open a \rightarrow \ldots \]
Applying the Model

Ambient

\[ a[\ldots]|\text{open } a \rightarrow \ldots \]

Components

\[ a[\ldots] \]

\[ \text{open } a \]

\[ x \quad a \quad y, \]

\[ L_{\text{open } a} \rightarrow z \]
Applying the Model

Ambient

\[ a[\ldots]|open\ a \rightarrow \ldots \]

Components

\[ a[\ldots] \quad x \xrightarrow{a} y, \]

\[ open\ a \quad L_{open\ a} \rightarrow z \]

Productions

\[ x \quad \xrightarrow{a} y \quad \text{open}\ a \quad \xrightarrow{[y/x]} \quad y = x \]

\[ L_{open\ a} \quad \xrightarrow{\text{open}\ a} \quad z \quad \rightarrow \quad z \]
Applying the Model: Node Fusion

\[ G \xrightarrow{a} L_{\text{open } a} \]
Applying the Model: Node Fusion

\[ G \xrightarrow{a} \text{open } a \]

\[ L_{\text{open } a} \]
Applying the Model: Node Fusion

\[ G \]

\[ L_{\text{open } a} \]

\[ \text{Intruder Knowledge} \]

\[ \text{P}_1 \text{ P}_n \]

\[ u \]

\[ y = x \]

\[ v = u \]
Graphs and Ambient

\[
\begin{align*}
\llbracket \text{nil} \rrbracket_x &= x \vdash \text{nil} \\
\llbracket n[P] \rrbracket_x &= x \vdash \nu y.(G \mid n(y, x)), \quad \text{if } y \neq x \land \llbracket P \rrbracket_y = y \vdash G \\
\llbracket M.P \rrbracket_x &= x \vdash L_{M.P}(x) \\
\llbracket P_1|P_2 \rrbracket_x &= x \vdash G_1 \mid G_2, \quad \text{if } \llbracket P_i \rrbracket_x = x \vdash G_i \land i = 1, 2 \\
\llbracket \text{rec } X. P \rrbracket_x &= \llbracket P[\text{rec } X. P/X] \rrbracket_x
\end{align*}
\]

Theorem \llbracket \_ \rrbracket \_ is a bijection on ambient graphs

Ambient Graphs

\[
\begin{array}{c}
\begin{array}{c}
 n_1 \\
\vdots \\
 n_h
\end{array}
\end{array}
\begin{array}{c}
G_1 \\
\vdots \\
G_h
\end{array}
\begin{array}{c}
L_{M_1.P_1} \\
\vdots \\
L_{M_k.P_k}
\end{array}
\]
Coordination Productions for Ambient

\[ x, y \vdash b(x, y) \xrightarrow{\{(x, \text{in } a, \langle \rangle), (y, \text{input } a, \langle z \rangle)\}} x, y, z \vdash b(x, z) \]

(input1)

\[ x \quad \bullet \quad b \quad \rightarrow \quad y \quad \bullet \quad \text{in } a \quad \text{input } a, z \quad \Rightarrow \quad x \quad \bullet \quad b \quad \rightarrow \quad y \quad \bullet \quad z \quad \text{open } a \]

(input2)

\[ x, y \vdash a(x, y) \xrightarrow{\{(y, \text{input } a, \langle x \rangle)\}} x, y \vdash a(x, y) \]

\[ x \quad \bullet \quad a \quad \rightarrow \quad y \quad \bullet \quad \text{input } a, x \quad \Rightarrow \quad x \quad \bullet \quad a \quad \rightarrow \quad y \quad \bullet \]
Theorem If $P \rightarrow Q$ then $\llbracket P \rrbracket_x \xrightarrow{\Lambda}_{id} \llbracket Q \rrbracket_x$ and

- either $\Lambda = \emptyset$
- or $\Lambda = \{(x, \tau, \langle \rangle)\}$
Theorem If $P \rightarrow Q$ then $\llbracket P \rrbracket_x \xrightarrow{\Lambda} \llbracket Q \rrbracket_x$ and

- either $\Lambda = \emptyset$
- or $\Lambda = \{(x, \tau, \langle \rangle)\}$

Theorem If $\llbracket P \rrbracket_x \xrightarrow{\Lambda} \Gamma \vdash G$ is a basic transition, then

- either $\llbracket P \rrbracket_x = \Gamma \vdash G$
- or $\exists Q \in \text{Proc} : P \rightarrow Q \land \Gamma \vdash G = \llbracket Q \rrbracket_x$
Klaim [DFP98]

Multiple TS Localities: first class citizens

Process migration

\[ P := \text{nil} \]

\[ j : P \]

\[ P_1 \]

\[ P_2 \]

\[ a \]

\[ @ \]

\[ s \]

\text{Klaim actions}

\[ j \]

\text{eval}

\[ (P) \]
Multiple TS
Multiple TS
Localities: first class citizens
Multiple TS
Localities: first class citizens
Process migration
- Multiple TS
- Localities: first class citizens
- Process migration

[Diagram showing sites and processes]

\[ P \setminus Q \Rightarrow R \]

\[ P' \setminus Q' \Rightarrow R' \]

\[
\begin{array}{c}
\text{site } s \\
R \\
P \\
Q \\
\end{array}
\]

\[
\begin{array}{c}
\text{site } s' \\
R' \\
Q' \\
\end{array}
\]
Klaim [DFP98]

- Multiple TS
- Localities: first class citizens
- Process migration

\[ P' \]@\[ s' \]

\[ Q \]@\[ s \]

\[ Q' \]@\[ s' \]

\[ a(t)@s' \]

\[ R \]

\[ R' \]

\[ P \]

\[ \text{site } s \]

\[ \text{site } s' \]

Intruder Knowledge

\( \theta \)

\( \pi \)

\( \text{Klaim actions} \)

\( \text{eval} \)
Multiple TS
Localities: first class citizens
Process migration

\begin{align*}
P & ::= \text{nil} \\
    & \mid \alpha.P \\
    & \mid P_1 \mid P_2 \\
\alpha & ::= a@s \\
a & ::= ... \text{ // Klaim actions} \\
    & \mid \text{eval}(P)
\end{align*}
In [BLP02]
Qlaim: Gateways

In [BLP02]

- Coordinators (super processes)
Claim: Gateways

In [BLP02]

- Coordinators (super processes)
- Dynamic creation of sites
Claim: Gateways

In [BLP02]

- Coordinators (super processes)
- Dynamic creation of sites
- Gateway connection management
Qlaim: Gateways

In [BLP02]

- Coordinators (super processes)
- Dynamic creation of sites
- Gateway connection management

Diagram:

```
<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
</tr>
</thead>
</table>
<pre><code>|
</code></pre>
```

site s
Claim: Gateways

In [BLP02]

- Coordinators (super processes)
- Dynamic creation of sites
- Gateway connection management
Claim: Gateways

In [BLP02]

- Coordinators (super processes)
- Dynamic creation of sites
- Gateway connection management

\[ Q \]

\[ P \]

\[ P' \]

\[ R \]

\[ R' \]

site s

site s'
Claim: Gateways

In [BLP02]
- Coordinators (super processes)
- Dynamic creation of sites
- Gateway connection management

\[
\begin{align*}
P & ::= \gamma.P \mid P_1 \mid P_2 \\
\gamma & ::= \alpha \\
& \mid \text{new}(s, P) \\
& \mid \text{login}(s, \kappa) \\
& \mid \text{accept}(s, \kappa) \\
& \mid \text{logout}(s, \kappa) \\
& \mid \text{disconnect}(s, \kappa)
\end{align*}
\]
Cost $\kappa$ abstracts characteristics of connections (bandwidth, latency, distance, access rights ...)
Connection costs

Cost $\kappa$ abstracts characteristics of connections (bandwidth, latency, distance, access rights ...)

Algebra on costs: $c$-semiring. For instance

\[
\langle c_1, \pi_1 \rangle \oplus \langle c_2, \pi_2 \rangle = \langle c_1 + c_2, \pi_1 \cup \pi_2 \rangle
\]

\[
\langle c_1, \pi_1 \rangle \otimes \langle c_2, \pi_2 \rangle = \begin{cases} 
\langle c_1 + c_2, \pi_1 \cap \pi_2 \rangle & \text{if } c_2 < c_1 \text{ and } \pi_2 \subseteq \pi_1 \\
\bot & \text{otherwise}
\end{cases}
\]
\[ [ s ::^L, P ] = \Gamma \vdash (\nu \vec{x}, p)([ P ]_p \mid \mathcal{G}_{m,n}^s(\vec{u}, \vec{x}, p) \mid \prod_{j=1}^{n} G_{t_j}^{k_j}(x_j, v_j)) \]
\[
[ s ::^L P ] = \Gamma \vdash (\nu \vec{x}, p)([ P ]_p | \mathcal{S}_{m,n}^s(\vec{u}, \vec{x}, p) | \prod_{j=1}^n G_{t_j}^{k_j}(x_j, v_j))
\]

\[
\begin{align*}
[ \text{nil} ]_p &= \text{nil} \\
[ \text{out} t ]_p &= L_{\text{out} t}(p) \\
[ \gamma . P ]_p &= L_{\gamma . P}(p) \\
[ \text{eval}(P) @ s ]_p &= (\nu u)(\text{eval}^T_s(P)(u, p) | S_P(u)) \\
[ P_1 | P_2 ]_p &= [ P_1 ]_p | [ P_2 ]_p \\
\end{align*}
\]
Qlaim’s Graph semantics: pros & cons

- Many productions (recently reduced)
- Determines the “optimal” path (also Qlaim)
- Path reservation
- Path routing

Theorem: Qlaim remote actions are routed on paths with minimal cost (wrt the c-semiring operations)
Qlaim’s Graph semantics: pros & cons

- Many productions (recently reduced :-)

Theorem
Qlaim remote actions are routed on paths with minimal cost (wrt the c-semiring operations).
Qlaim’s Graph semantics: pros & cons

- Many productions (recently reduced :-)
+ Determines the “optimal” path (also Qlaim)
Qlaim’s Graph semantics: pros & cons

- Many productions (recently reduced :-)
  + Determines the “optimal” path (also Qlaim)
  + Path reservation
Qlaim’s Graph semantics: pros & cons

- Many productions (recently reduced :-)
  + Determines the “optimal” path (also Qlaim)
  + Path reservation
  + Path routing
Qlaim’s Graph semantics: pros & cons

- Many productions (recently reduced :-)
  + Determines the “optimal” path (also Qlaim)
  + Path reservation
  + Path routing

**Theorem** Qlaim remote actions are routed on paths with minimal cost (wrt the c-semiring operations)
In [KGKK02] graph transformation is used for modelling dynamic behaviour of UML specifications.

+ Formal semantics of computations
  - No local rewrites
  - Distribution is not considered

SHR has been applied as a further refinement step in the software design process.
Security
The Dolev-Yao Model

Intruder Knowledge

P1  Pn

Intruder
Knowledge

Perfect Encryption Hypothesis

Names n; m; :::; A; B; S; :::;

Keys k; k0; :::; A + A; A + A; :::;

Messages M ::= N K M; M f M g M
The Dolev-Yao Model

- Receive and store any transmitted message
- Hinder a message
- Decompose messages into parts
- Forge messages using known data
- Perfect Encryption Hypothesis
The Dolev-Yao Model

Receive and store any transmitted message
Hinder a message
Decompose messages into parts
Forge messages using known data
Perfect Encryption Hypothesis

Names \( n, m, \ldots, A, B, S, \ldots \)
Keys \( k, k', \ldots, A^+, A^-, \ldots \)
Messages \( M ::= N \mid K \mid M, M \mid \{ M \}_M \)
Intruder capabilities: ⌵

\[ \frac{m \in \kappa}{\kappa \uplus m} \quad (\in) \quad \frac{\kappa \uplus m, \kappa \uplus n}{\kappa \uplus m, n} \quad (, ) \quad \frac{\kappa \uplus m \kappa \uplus \lambda}{\kappa \uplus \{m\}_\lambda} \quad (\{\}) \]

\[ \frac{\kappa \uplus m, n}{\kappa \uplus m} \quad (+1) \quad \frac{\kappa \uplus m, n}{\kappa \uplus n} \quad (+2) \quad \frac{\kappa \uplus \{m\}_\lambda \kappa \uplus \lambda^-}{\kappa \uplus m} \quad (\{\}) \]

Generalising \[\text{[CJM98]}\] to asymmetric key cryptography

**Theorem** ⌵ is decidable
Some design choices:

- Cryptography & communication (pattern-matching)
- Key-sharing via “name fusion”
- Rôle based calculus
- Multi-session facilities
## Syntax of cIP

### Extension of IP [BBT01]

\[
E, F ::= \text{nil} \mid \alpha.E \mid E + E \mid E|E
\]

\[
\alpha, \beta ::= \text{in}(d) \mid \text{out}(d)
\]

\[
d ::= N \mid K \mid d, d \mid \{d\}_d \mid x \mid ?x
\]

1. \(A \rightarrow B : \{na, A\}_{B+}\)
2. \(B \rightarrow A : \{na, nb\}_{A+}\)
3. \(A \rightarrow B : \{nb\}_{B+}\)

\[
A \triangleq (y)[ \quad \text{out}(\{na, A\}_{y+}). \quad \text{in}(\{na, ?u\}_{A-}). \quad \text{out}(\{u\}_{y+})]
\]

\[
B \triangleq ()[ \quad \text{in}(\{?x, ?z\}_{B-}). \quad \text{out}(\{x, nb\}_{z+}). \quad \text{in}(\{nb\}_{B-})]
\]
cIP Semantics

\[
\begin{align*}
\alpha. E & \xrightarrow{\alpha} E \\
E & \xrightarrow{\alpha} E' \\
E + F & \xrightarrow{\alpha} E' \\
E & \xrightarrow{\alpha} E' \\
E \parallel F & \xrightarrow{\alpha} E' \parallel F \\
bn(\alpha) \cap \text{fn}(F) = \emptyset
\end{align*}
\]

\[
\begin{align*}
E_i \xrightarrow{in(d)} E_i' \\
\partial(\kappa) \triangleright m : \exists \sigma \text{ ground s.t. } d\sigma \sim m \\
\langle (\tilde{X}_i)[E_i] \cup C, \chi, \kappa \rangle & \leftrightarrow \langle (\tilde{X}_i)[E'_i\sigma] \cup C, \chi_\sigma, \kappa \rangle \\
E_i & \xrightarrow{out(m)} E'_i \\
\langle (\tilde{X}_i)[E_i] \cup C, \chi, \kappa \rangle & \leftrightarrow \langle (\tilde{X}_i)[E'_i] \cup C, \chi, \kappa \cup m \rangle \\
C' & = \text{join}(A_i, \gamma, C) \\
A \triangleq (\tilde{X})[E] & \text{ i new} \\
\langle C, \chi, \kappa \rangle & \leftrightarrow \langle C', \chi_\gamma, \kappa \cup \{A_i\} \rangle
\end{align*}
\]
\[ \phi, \psi \ ::= \delta \in \mathcal{K} \\
| \ \ \ \forall \alpha : A. \phi \\
| \ x@\alpha = \delta \\
| \alpha = \beta \\
| \neg \phi \ |
\]

\[ \delta \ ::= \ d \ |
\alpha \\
| \ x@\alpha
\]

\[ \kappa \models_\chi \phi \]

“If \( B \) completes a protocol session and thinks that he has been talking to \( A \), then \( A \) had started a protocol session thinking that she has been talking to \( B \)”

\[ \forall \beta : B. \exists \alpha : A. (z@\beta = \alpha \rightarrow y@\alpha = \beta) \]
Mihda: Co-Algebraic Minimisation of Automata
Minimizing History Dependent Automata:
- HD-automata for history dependent calculi
- Co-algebraic specification
- Partition Refinement Algorithm based on co-algebraic specification [FMP02]
- Mihda: Ocaml implementation

<table>
<thead>
<tr>
<th></th>
<th>Comp. Time</th>
<th>States</th>
<th>Trans.</th>
<th>Min. Time</th>
<th>States</th>
<th>Trans.</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSM small</td>
<td>0m 0.931s</td>
<td>211</td>
<td>398</td>
<td>0m 4.193s</td>
<td>105</td>
<td>197</td>
</tr>
<tr>
<td>GSM full</td>
<td>0m 8.186s</td>
<td>964</td>
<td>1778</td>
<td>0m 54.690s</td>
<td>137</td>
<td>253</td>
</tr>
</tbody>
</table>
Mihda Architecture

- Adherent to specs
- Highly modular
- Easily extendible
The main step
The main step

let bundle hd q =
List.sort compare
  (List.iter (fun h → (Arrow.source h) = q) (arrows hd))
The main step

List.map $h_n$ bundle
\[ h_{n+1} = \text{norm}(\text{states}, \{\langle \ell, \pi, h_n(q'), \sigma'; \sigma \rangle | q \xrightarrow{\ell \pi \sigma} q' \land \sigma' \in \Sigma_n(q')\}) \]

\text{let red bl = ......}

\text{let bl_in = List.filter covered_in bl}

\text{in list_diff bl bl_in}
let an = active_names_bundle (red bundle) in
let remove_in ar = match ar with
  | Arrow(_,_,In(_,_)) → not (List.mem (obj ar) an)
  | _ → false in
  list_diff bundle (List.filter remove_in bundle)
The main step

\[ \Sigma_{n+1}(q) = \left( \text{compute}\_\text{group} \ (\text{norm}\ \text{bundle}) \right) ; \ \theta_q^{-1} \]
The main step

\[ \Sigma_{n+1}(q) = (\text{compute\_group} \ (\text{norm bundle})) \ ; \ \theta_q^{-1} \]

**Theorem** At the end of each iteration \( i \) **blocks** corresponds to \( h_{H_i} \)
http://jordie.di.unipi.it:8080/pweb
Initial steps toward:

- Declarative approach to WAN programming
  - Foundational aspects
  - QoS at application level
  - Software Architectures (to be developed)
- Web Services
  - Secure composition of components
  - Coordination mechanism
- Tool development
  - Distributed infrastructure
  - Proof strategies as programmable coordinators
Published papers


References


[CJM98] Edmund M. Clarke, Somesh Jha, and Wilfredo R. Marrero. Using state space exploration and a natural deduction style message derivation engine to


School of Cognitive and Computing Sciences, University of Sussex.


