A Declarative Approach to Wide-Area Network Programming with Service Level Agreement

Emilio Tuosto

Dipartimento di Informatica, Università di Pisa
via F. Buonarroti 2, 56127 Pisa, Italy. +39-50-2212799. +39-50-2212726.
etuosto@di.unipi.it – http://www.di.unipi.it/~etuosto

Padova, 6 February 2004
Plan of the talk

- What 'WAN programming' means (for us)
- Declarative programming model: Hypergraphs
  - Hint on its adequacy
    - Ambient calculus
    - Wireless communications
    - GRID
- Programming SLA: KAOS
  - SLA & Hypergraphs: reasoning on optimal routing
- SLAK
- Final considerations
Global Computing

WAN programming for building global systems. They are hard to be made robust because:

- Absence of centralised control
- Client-Server not enough: P2P
- Administrative domains (Security)
- Interoperability
  - different platforms
  - different devices
    (e.g. PDA, laptop, mobile phones...)
- “Mobility” (resources and computation)
- ...

- G2G1 L
- 1 54
- 2
- 3

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>Q'</th>
</tr>
</thead>
<tbody>
<tr>
<td>P'</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

κ

– p. 3/54
Global Computing

WAN programming for building global systems. They are hard to be made robust because:

- Network Awareness
  - Applications are location dependent
  - Locations have different features
  - and allow multiple access policies
  - Independently programmed in a distributed environment
  - Reasoning on space and time
  - ...

- ...
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”
Motivations

WAN programming is not just $\text{go}(P)$, $\bar{s}(x)$ or $s(y)$

- Lifting SLA issues to application level...
- ...in a WAN scenario, where programming is composition of WS
- SLA as a coordination mechanism
- A Formal Basis for Reasoning on Programmable SLA [DFM+03]
- Resource availability and access as a parameter for the SLA
- Proof techniques and tools
A Model for Declarative WAN Programming

In collaboration with

G. Ferrari (Pisa) and U. Montanari (Pisa)

Working Group:
Dan Hirsch (Pisa), Ivan Lanese (Pisa),
<table>
<thead>
<tr>
<th>Process Algebraic Foundations of WAN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$\pi$-calculus</strong> [MPW92]</td>
</tr>
<tr>
<td>Djoin [FG96, FGL+96]</td>
</tr>
<tr>
<td>$D_\pi$ [HR98, HR00]</td>
</tr>
<tr>
<td>Fusion [PV98]</td>
</tr>
<tr>
<td>Rich theory</td>
</tr>
<tr>
<td>basic wrt WAN</td>
</tr>
<tr>
<td>(only link mobility)</td>
</tr>
<tr>
<td><strong>Ambient</strong> [CG00]</td>
</tr>
<tr>
<td>Seal [VC98]</td>
</tr>
<tr>
<td>Boxed [BCC01]</td>
</tr>
<tr>
<td>Safe [LS00]</td>
</tr>
<tr>
<td>Hierarchical</td>
</tr>
<tr>
<td>not very natural</td>
</tr>
<tr>
<td><strong>Klaim</strong> [BBD+03]</td>
</tr>
<tr>
<td>Hierarchical [BLP02]</td>
</tr>
<tr>
<td>OKlaim [BBV03]</td>
</tr>
<tr>
<td>MetaKlaim [FMPar]</td>
</tr>
<tr>
<td>Very natural</td>
</tr>
<tr>
<td>Lack of observational semantics</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
Hypergraphs Programming model 1

- Edge replacement for graph rewritings [Fed71, Pav72]
- Graphs for distributed systems [CM83, 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]
- Extension to node fusions [FMT01]
- ...

---

Footnote: 1
Hypergraphs Programming model

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 synchronously rewrite hyperedges...
- ...according to a synchronisation policy
Replacement of Hyperedges

\[ L \rightarrow G \]
Replacement of Hyperedges

$L \rightarrow G$

![Diagram](image_url)
Replacement of Hyperedges

$L \rightarrow G$
Replacement of Hyperedges

$L \rightarrow G$

- Edge replacement: local
- Synchronisation as distributed constraint solving
- Multiple synchronisation
- New node creation
- Node fusion: model of mobility and communication

G1  G  G2'
Replacement of Hyperedges

$L \rightarrow G$

- Edge replacement: local
- Synchronisation as distributed constraint solving
- Multiple synchronisation
- New node creation
- Node fusion: model of mobility and communication

Benefits:
- Uniform framework for $\pi$, $\pi$-I, fusion
- LTS for Ambient ...
- ... for Klaim ...

Site $s$'site $s$
Replacement of Hyperedges

$L \rightarrow G$

- Edge replacement: local
- Synchronisation as distributed constraint solving
- Multiple synchronisation
- New node creation
- Node fusion: model of mobility and communication

Benefits:
- Uniform framework for $\pi$, $\pi$-l, fusion
- LTS for Ambient ...
- ... for Klaim ...
- ... and path reservation for KAOS
- Expressive for distributed coordination
- Wireless networks
A hyperedge generalises edges: It connects more than two nodes

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

\[ x \bullet - 3 - L - 2 - z \bullet \]

\[ y \bullet \]

\[ L(y, z, x) \]
A hyperedge generalises edges: It connects more than two nodes

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

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

\[ x \quad \quad L \quad \quad y \quad \quad z \]
A hyperedge generalises edges: It connects more than two nodes

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

\[
\begin{array}{c}
\text{Syntactic Judgement} \\
\Gamma \vdash G, \quad fn(G) \subseteq \Gamma
\end{array}
\]
A hyperedge generalises edges: It connects more than two nodes

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

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

**Syntactic Judgement**

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

An example:

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

\[ x, y \vdash \nu z.(L(y,z,x)|M(y,z)) \]
A hyperedge generalises edges: It connects more than two nodes

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

\[
\begin{array}{c}
\text{Syntactic Judgement} \\
\Gamma \vdash G, \quad \text{fn}(G) \subseteq \Gamma
\end{array}
\]

An example:

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

\[
x, y \vdash \nu z. (L(y, z, x) | M(y, z))
\]
Hypergraph Semantics: Productions

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

- \( \Lambda \subseteq X \times Act \times \mathcal{N}^* \) set of constraints
- \( \pi : X \to X \) 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: 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.

\[
\forall x_i, x_j \in X. \pi(x_i) = x_j \Rightarrow \pi(x_j) = x_j
\]

- \( \Gamma = \pi(X) \cup (n(\Lambda) \setminus X) \)
- \( \text{fn}(G) \subseteq \Gamma \)

**Graph Rewritings**

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

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

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

\[ \rho = \left[ \frac{\pi(x)}{\pi(y)} \right] \]

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

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

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

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

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

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

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

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

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

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

\[ \Gamma_1 \cup \Gamma_1' \vdash G_1 | G_1' \xrightarrow{\lambda \cup \lambda'}_{\pi \cup \pi'} \Gamma_2 \cup \Gamma_2' \vdash G_2 | G_2' \]
Hypergraph Adequacy: Ambient

In collaboration with
G. Ferrari (Pisa) and U. Montanari (Pisa)
From SHR to Ambient

Ambient $a[...] | open a \rightarrow \ldots$
From SHR to Ambient

Ambient

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

Components

\[ a[\cdots]: \quad x \xrightarrow{a} y, \]

\[ \text{open } a: \quad \boxed{L_{\text{open } a}} \rightarrow z \]
From SHR to Ambient

Ambient

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

Components

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

\[ \text{open } a : \quad \xrightarrow{L_{\text{open } a}} \quad z \]

Productions

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

\[ \xrightarrow{L_{\text{open } a}} \quad \xrightarrow{\text{open } a} \quad z \]
Node Fusion

\[ G \rightarrow a \leftarrow L_{\text{open } a} \]
Node Fusion

$G$

$a$

$\text{open } a$

$L_{\text{open } a}$

$\text{open } a$

$L_{\text{open } a}$
\[
[\text{nil}]_x = x \vdash \text{nil}
\]
\[
[n[P]]_x = x \vdash \nu y. (G \mid n(y, x)), \text{ if } y \neq x \land [P]_y = y \vdash G
\]
\[
[M.P]_x = x \vdash L_{M.P}(x)
\]
\[
[P_1 | P_2]_x = x \vdash G_1 | G_2, \text{ if } [P_i]_x = x \vdash G_i \land i = 1, 2
\]
\[
[\text{rec } X. P]_x = [P[\text{rec } X. P / X]]_x
\]

\textbf{Theorem} \[\_\] is a bijection on ambient graphs
Coordination Productions for Ambient

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

(input1)

\[ x \xrightarrow{b} y \quad \text{in } a \quad \xrightarrow{\text{input } a, z} \quad x \xrightarrow{b} y \]

\[ x \xrightarrow{\text{input } a, z} y \]

(input2)

\[ x \xrightarrow{a} y \quad \text{in } a, x \quad \xrightarrow{\text{input } a, x} \quad x \xrightarrow{a} y \]

\[ x \xrightarrow{a} y \]
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)*

\[ x \]
\[ \bullet \]
\[ \text{in } a \]
\[ \xrightarrow{b} \]
\[ \bullet \]
\[ \text{input } a, z \]
\[ \Rightarrow \]
\[ x \]
\[ \bullet \]
\[ \leftarrow b \]
\[ \bullet \]
\[ \downarrow \]
\[ y \]
\[ \bullet \]
\[ \Rightarrow \]

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

*(input2)*

\[ x \]
\[ \bullet \]
\[ \leftarrow a \]
\[ \bullet \]
\[ \text{input } a, x \]
\[ \Rightarrow \]
\[ x \]
\[ \bullet \]
\[ \leftarrow a \]
\[ \bullet \]
\[ \Rightarrow \]
\[ y \]
\[ \bullet \]
Theorem If $P \rightarrow Q$ then $\sem{P}_x \xrightarrow{\Lambda} \sem{Q}_x$ and

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

Semantic Correspondence
Semantic Correspondence

**Theorem** If $P \rightarrow Q$ then $\left[ P \right]_x \xrightarrow{\Lambda} \left[ Q \right]_x$ and

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

**Theorem** If $\left[ P \right]_x \xrightarrow{\Lambda} \Gamma \vdash G$ is a basic transition, then

- either $\left[ P \right]_x = \Gamma \vdash G$
- or $\exists Q \in Proc : P \rightarrow Q \land \Gamma \vdash G = \left[ Q \right]_x$
Hypergraph Adequacy: Wireless communications
Wireless Phenomena

- Wireless networks devices present peculiarities wrt wired ones
  - Dynamism of network topology
  - Energy constraints
  - Transmitting capacity
- Hence, nodes can asynchronously disappear
- Wireless networks typically are peer-to-peer
- the physical environment might cause interferences or interdict communications

For ad-hoc networks that share the same spectrum, new methods of cooperation are required to permit coexistence. Such methods are difficult to research without real-world channel models and simulation methodologies; there is still fundamental work to be done in this area [Mob98]
Traditional Models not Satisfactory

The communication infrastructure does not permit to individuate the position of components by their name

- Ambient: “vicinity” condition does not encompass any distance concept (| is commutative, therefore \( a[P|Q|R] = a[P|R|Q] \))
- KAOS can deal with distance between but...

... neither Ambient nor KAOS can easily model interference on wireless communications caused by *third party* movements.

This is difficult to capture in traditional frameworks because even if a link encompass the distance between nodes, it is under the control of the connected nodes and a third entity cannot “break” or modify it.
Tarzan is a SHR-based framework that captures

- Radio/Infrared Signal propagation
- Devices nomadism
- Physical environment characteristics
**Tarzan** is a SHR-based framework that captures

- Radio/Infrared Signal propagation
- Devices nomadism
- Physical environment characteristics

![Diagram of Tarzan framework](image)
Tarzan is a SHR-based framework that captures
- Radio/Infrared Signal propagation
- Devices nomadism
- Physical environment characteristics
Hypergraph Adequacy: GRID

In collaboration with
Marco Aldinucci (Pisa)
Semantics for GRID

Grid-aware applications are usually made of cooperating components with a graph topology. Successively, behaviors to adhere performance and fault-tolerance constraints are defined. These behaviors regard parallelism degree adaptivity matching both performance and fault-tolerance requirements. Point out patterns of behaviors for abstracting suitable primitives.
Semantics for GRID

- Grid-aware applications are usually made of cooperating components with a graph topology.
Semantics for GRID

- Grid-aware applications are usually made of cooperating components with a graph topology.
- Successively behaviors to adhere performance and fault-tolerance constraints are defined.
Grid-aware applications are usually made of cooperating components with a graph topology.

Successively behaviors to adhere performance and fault-tolerance constraints are defined.

These behaviors regard parallelism degree adaptivity matching both performance and fault-tolerance requirements.
Grid-aware applications are usually made of cooperating components with a graph topology.

Successively behaviors to adhere performance and fault-tolerance constraints are defined.

These behaviors regard parallelism degree adaptivity matching both performance and fault-tolerance requirements.

Point out patterns of behaviors for abstracting suitable primitives.
Our goal is to formally define a basic language containing primitives suitable for GRID.

Grid computing tries to enable the development of large applications. Grid-aware applications make use of computational power of distributed resources. Developing algorithms able to exploit GRID is difficult. Programmers must design highly concurrent WAN algorithms with few homogeneity hypothesis. Hence programmers have to face up classical problems of parallel computing as well as Grid-specific ones.
Our goal is to formally define a basic language containing primitives suitable for GRID.
Our goal is to formally define a basic language containing primitives suitable for GRID.

Grid computing tries to enable the development of large applications.
Our goal is to formally define a basic language containing primitives suitable for GRID.

Grid computing tries to enable the development of large applications.

Grid-aware applications make use of computational power of distributed resources.
Our goal is to formally define a basic language containing primitives suitable for GRID.

Grid computing tries to enable the development of large applications

Grid-aware applications make use of computational power of distributed resources

Developing algorithms able to exploit GRID is difficult
Our goal is to formally define a basic language containing primitives suitable for GRID.

Grid computing tries to enable the development of large applications.

Grid-aware applications make use of computational power of distributed resources.

Developing algorithms able to exploit GRID is difficult.

Programmers must design highly concurrent WAN algorithms with few homogeneity hypothesis.
Our goal is to formally define a basic language containing primitives suitable for GRID.

Grid computing tries to enable the development of large applications.

Grid-aware applications make use of computational power of distributed resources.

Developing algorithms able to exploit GRID is difficult.

Programmers must design highly concurrent WAN algorithms with few homogeneity hypothesis.

Hence programmers have to face up classical problems of parallel computing as well as Grid-specific ones.
SHR as a Semantics for GRID

- We provide a high-level programming model for GRID programming
- We describe a SHR semantics of the framework

For instance

- Migration
- Replication
- Kill

| Components as hyperedges |
| Coordination interface separated by its computational activity |
| SHR rewriting mechanism for coordinating components |
SHR as a Semantics for GRID
KAOS: Expressing and reasoning on Connection Properties

In collaboration with
R. De Nicola (Firenze), G. Ferrari, U. Montanari, R. Pugliese (Firenze)
Multiple tuple spaces
Localities: first class citizens
Process migration

P := nil
j : P
j P 1 := a
@ s
a := ...

// Klaim actions

Multiple tuple spaces
Multiple tuple spaces

Localities: first class citizens
Multiple tuple spaces

Localities: first class citizens

Process migration
Multiple tuple spaces
Localities: first class citizens
Process migration

![Diagram](image-url)
Multiple tuple spaces
Localities: first class citizens
Process migration

\[ P', a(t)@s' \]

[DFP98]
Multiple tuple spaces

Localities: first class citizens

Process migration

$P ::= \text{nil}$

$| \alpha.P$

$| P_1 | P_2$

$\alpha ::= a@s$

$a ::= \ldots$ // Klaim actions

$| \varepsilon(P)$
KAOS: Gateways

Coordinators (super processes)
KAOS: Gateways

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

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

[BLP02]
KAOS: Gateways

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

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

new(s',|P')

site s

site s'
KAOS: Gateways

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

[BLP02]
Coordinators (super processes)

Dynamic creation of sites

Gateway connection management

\[
P ::= \gamma . P \mid P_1 \mid P_2
\]

\[
\gamma ::= \alpha
\]

\[
\mid \text{new}(s, P)
\]

\[
\mid \text{link}(s_\kappa)
\]

\[
\mid \text{accept}(s_\kappa)
\]

\[
\mid \text{logout}(s)
\]

\[
\mid \text{disc}(s)
\]

[BLP02]
Connection costs

κ abstracts characteristics of connections (distance, access rights, price ...
abstracts characteristics of connections (distance, access rights, price ...)

Algebra on costs: c-semiring \[\text{[BMR95, BMR97]}\]

\[\langle A, +, *, 0, 1 \rangle \text{ where} \]

- \( A \) is a set
- \( 0, 1 \in A \)
- \( + : A \times A \rightarrow A \)
- \( \ast : A \times A \rightarrow A \)

\[
\begin{align*}
  x + y &= y + x & (x + y) + z &= x + (y + z) \\
  x + x &= x & x + 0 &= x & x + 1 &= 1 \\
  x \ast y &= y \ast x & (x \ast y) \ast z &= x \ast (y \ast z) \\
  x \ast 1 &= x & x \ast 0 &= 0 \\
  (x + y) \ast z &= (x \ast z) + (y \ast z) 
\end{align*}
\]
Connection costs

κ abstracts characteristics of connections (distance, access rights, price ...)
Algebra on costs: c-semiring \([\text{BMR95, BMR97}]\)
\(<A, +, *, 0, 1>\) where

- \(A\) is a set
- \(0, 1 \in A\)
- \(+ : A \times A \to A\)
- \(* : A \times A \to A\)

\[
\begin{align*}
x + y &= y + x & (x + y) + z &= x + (y + z) \\
x + x &= x & x + 0 &= x & x + 1 &= 1 \\
x \times y &= y \times x & (x \times y) \times z &= x \times (y \times z) \\
x \times 1 &= x & x \times 0 &= 0 & (x + y) \times z &= (x \times z) + (y \times z)
\end{align*}
\]

\[a \leq b \iff \exists c : a + c = b\]

\(a \leq b\) means that \(a\) is more constrained than \(b\).
Connection costs: examples

- $\{T, F\}, \lor, \land, F, T$ truth values
- $\langle N, \text{min}, +, +\infty, 0 \rangle$, the c-semiring of natural numbers $N$
- $\langle \wp(\{A\}), \cup, \cap, A, A \rangle$, the powerset semiring

Cartesian product of c-semirings is a c-semiring. For instance

$$\langle c_1, \pi_1 \rangle \oplus \langle c_2, \pi_2 \rangle = \langle c_1 \text{ min } c_2, \pi_1 \cup \pi_2 \rangle$$

$$\langle c_1, \pi_1 \rangle \otimes \langle c_2, \pi_2 \rangle = \langle c_1 + c_2, \pi_1 \cap \pi_2 \rangle$$
Syntax of KAOS

\[ N ::= \text{Nets} \]

\[ s ::=^L P \quad \text{Single node} \]
\[ (\nu s)N \quad \text{Node restriction} \]
\[ N_1 \parallel N_2 \quad \text{Net composition} \]

\[ l ::= \text{Links} \]

\[ \langle s, \kappa \rangle \quad \text{Incoming link} \]
\[ \langle \kappa, s \rangle \quad \text{Outgoing link} \]

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

\[ \gamma ::= \text{Actions} \]

\[ (s) \quad \text{Input} \]
\[ \text{new}(s_\kappa) \quad \text{Node creation} \]
\[ \text{link}(s_\kappa) \quad \text{Login} \]
\[ \text{accept}(s_\kappa) \quad \text{Accept} \]
\[ \delta l \quad \text{Disconnect} \]

\[ \gamma . P \quad \text{Action prefixing} \]
\[ \epsilon(P)@s \quad \text{Remote spawning} \]
\[ P_1 \parallel P_2 \quad \text{Parallel} \]
\[ X \quad \text{Process vars} \]

\[ \text{nil} \quad \text{Null process} \]
Semantics of KAOS

(OUT) \[ s ::L \text{out}(t) \xrightarrow{s \triangleright t} s ::L \text{nil} \]

(IN) \[ s ::L (x).P \xrightarrow{s \downarrow t} s ::L P[t/x] \]

(LEVAL) \[ s ::L \langle s, \kappa \rangle \varepsilon(P)@s \xrightarrow{\tau} s ::L \langle s, \kappa \rangle P, \quad \text{if } \kappa \models T(P) \]

(EVAL) \[ s ::L \varepsilon(P)@t \xrightarrow{s(\emptyset, P)@t} s ::L \text{nil}, \quad \text{if } s \neq t \]

\[ s ::L (\text{new}(x_\kappa).P) | Q \xrightarrow{\tau} (\nu x)(s ::L \langle \kappa, x \rangle P | Q | x ::\langle s, \kappa \rangle \text{nil}), \quad \text{if } x \notin n(L) \cup \{s\} \cup \text{fn}(Q) \]

(LLOGIN) \[ s ::L \text{link}(s_\kappa).P \xrightarrow{\tau} s ::L \{\langle s, \kappa \rangle, \langle \kappa, s \rangle \} P \]

(LOGIN) \[ s ::L \text{link}(t_\kappa).P \xrightarrow{s \rightarrow t} s ::L \langle \kappa, t \rangle P, \quad \text{if } s \neq t \]

(ACCEPT) \[ s ::L \text{accept}(t_{\kappa'}).P \xrightarrow{t \rightarrow s} s ::L \langle t, \kappa \rangle P, \quad \text{if } \kappa \leq \kappa' \]

(LDISC) \[ s ::L \delta(s, \kappa).P \xrightarrow{\tau} s ::L \langle s \rangle \langle \kappa, s \rangle P \]

(IDISC) \[ s ::L \delta(t, \kappa).P \xrightarrow{\delta(s, \langle t, \kappa \rangle)} s ::L \langle t, \kappa \rangle P, \quad \text{if } t \neq s \]

(ODISC) \[ s ::L \delta(\kappa, t).P \xrightarrow{\delta(s, \langle \kappa, t \rangle)} s ::L \langle \kappa, t \rangle P, \quad \text{if } t \neq s \]

(NODE) \[ s ::L \cup \langle r, \kappa \rangle P \xrightarrow{X@s} s ::L \cup \langle r, \kappa \rangle P | X, \quad \text{if } X \text{ fresh} \]
\[
\begin{align*}
\llbracket s :: \llbracket P \rrbracket \rrbracket = \Gamma \vdash (\nu \vec{x}, p)(\llbracket P \rrbracket_p | \mathcal{G}_m^n (\vec{u}, \vec{x}, p) | \prod_{j=1}^{n} G_{t_j}^{\kappa_j}(x_j, v_j))
\end{align*}
\]
\[
\begin{align*}
\mathbf{ [ s ::^L P ] } & = \Gamma \vdash (\nu \vec{x}, p) (\mathbf{ [ P ] }_p | \mathcal{S}_{m,n}^s((\vec{u}, \vec{x}, p) | \prod_{j=1}^n G_{i,j}^{r_j}(x_j, v_j)) \\
[ \text{nil} ]_p & = \text{nil} \\
[ \text{out}(t) ]_p & = L_{\text{out}(t)}(p) \\
[ \gamma.P ]_p & = L_{\gamma.P}(p) \\
[ \varepsilon(P)@s ]_p & = (\nu u)(\varepsilon^{T(P)}(u, p) | S_P(u)) \\
[ P_1 \mid P_2 ]_p & = [ P_1 ]_p \mid [ P_2 ]_p \\
[ \text{rec X. } P ]_p & = [ P[^{\text{rec X. } P / X}] ]_p.
\end{align*}
\]
KAOS’s Graph semantics: pros & cons

Many productions (recently reduced)

Theorem

\[ G \neq 0 \Rightarrow \]

1. \( u \) and \( v \) are link-connected by a path of cost \( g \);
2. for any \( o \) there is a \( G \)

\[ Q_{hi} = 1 \]

(\( Q \) is the c-semiring multiplication)
KAOS’s Graph semantics: pros & cons

- Many productions (recently reduced :-)

\[ G \Rightarrow \text{path} \Rightarrow 0 \]

If \[ G \Rightarrow \text{path} \Rightarrow 0 \]

1. \[ \text{u} \text{ and } \text{v} \text{ are link-connected by a path of cost} \]

2. \[ \text{for any } o \circ \circ \text{u} \circ \circ \circ \text{s} \circ \circ \circ \text{m} \circ \circ \circ \text{n} \circ \circ \circ \text{G} \circ \circ \circ \text{1} \circ \circ \circ \text{s} \circ \circ \circ \text{1} \]

\[ \text{h} \circ \circ \circ \text{v} \circ \circ \circ \text{1} \]

\[ \text{there is a } G \Rightarrow \text{path} \Rightarrow 0 \text{ s.t.} \]

\[ Q \circ \circ \text{h} \circ \circ \circ 1 \]

\[ Q \text{is the c-semiring multiplication} \]

Path reservation

Optimal path routing (e.g., Floyd-Warshall)
KAOS’s Graph semantics: pros & cons

- Many productions (recently reduced :-)
- Determines the “optimal” path (also KAOS)

**Theorem** If \( \Gamma \vdash G \xrightarrow{\Lambda \cup \{u, v, \kappa, \langle u \rangle\}} \Gamma' \vdash G' \) then

1. \( u \) and \( v \) are link-connected by a path of cost \( \kappa \);
2. for any

\[
\begin{array}{c}
\bullet \quad G_{m_1,n_1}^s \quad \cdots \quad G_{s_1}^{\kappa_1} \quad \bullet \quad \cdots \quad \bullet \quad G_{m_h,n_h}^s \quad \cdots \quad G_t^{\kappa_h} \quad \bullet
\end{array}
\]

there is a \( \Gamma \vdash G \xrightarrow{\Lambda \cup \{u, v, \kappa, \langle u \rangle\}} \Gamma' \vdash G' \) s.t. \( \kappa \leq \prod_{i=1}^{h} \kappa_i \).

(\( \prod \) is the c-semiring multiplication)
KAOS’s Graph semantics: pros & cons

- Many productions (recently reduced :-)

- Determines the “optimal” path (also KAOS)

**Theorem** If 
\[ \Gamma \vdash G \xrightarrow{\Lambda \cup \{(u, v, \kappa, \langle u \rangle)\}} \Gamma' \vdash G' \]

then

1. \( u \) and \( v \) are link-connected by a path of cost \( \kappa \);
2. for any

\[
\begin{align*}
&\begin{array}{c}
\bullet \quad G_{m_1,n_1}^s \\
\circ \quad \circ \quad \circ
\end{array} \\
&\begin{array}{c}
\bullet \quad G_{s_1}^{\kappa_1} \\
\circ \quad \circ \quad \circ
\end{array} \\
&\begin{array}{c}
\bullet \quad \vdash \quad \vdash \\
\bullet \quad \bullet \quad \bullet
\end{array} \\
&\begin{array}{c}
\bullet \quad G_{m_h,n_h}^s \\
\circ \quad \circ \quad \circ
\end{array} \\
&\begin{array}{c}
\bullet \quad \vdash \quad \vdash \\
\bullet \quad \bullet \quad \bullet
\end{array} \\
&\begin{array}{c}
\bullet \quad G_{t}^{\kappa_h} \\
\circ \quad \circ \quad \circ
\end{array} \\
&\begin{array}{c}
\bullet \quad \vdash \quad \vdash \\
\bullet \quad \bullet \quad \bullet
\end{array} \\
&\begin{array}{c}
\bullet \quad \bullet \quad \bullet
\end{array}
\end{align*}
\]

there is a \( \Gamma \vdash G \xrightarrow{\Lambda \cup \{(u, v, \kappa, \langle u \rangle)\}} \Gamma' \vdash G' \) s.t. \( \kappa \leq \prod_{i=1}^{h} \kappa_i \).

(\( \prod \) is the c-semiring multiplication)

+ Path reservation
KAOS’s Graph semantics: pros & cons

- Many productions (recently reduced :-)
- Determines the “optimal” path (also KAOS)

Theorem If \( \Gamma \vdash G \xrightarrow{\Lambda \cup \{u, v, \kappa, \langle u \rangle \}} \Gamma' \vdash G' \) then

1. \( u \) and \( v \) are link-connected by a path of cost \( \kappa \);
2. for any

\[
\begin{array}{c}
\bullet u \\
\Rightarrow G'_{s_1}^{\kappa_1} \\
\Rightarrow \cdots \\
\Rightarrow \bullet v_1 \\
\cdots
\end{array}
\]

\[
\begin{array}{c}
\Rightarrow \cdots \\
\Rightarrow \bullet \quad \cdots \\
\Rightarrow \bullet \\
\Rightarrow G_t^{\kappa_h}
\end{array}
\]

there is a \( \Gamma \vdash G \xrightarrow{\Lambda \cup \{u, v, \kappa, \langle u \rangle \}} \Gamma' \vdash G' \) s.t. \( \kappa \leq \prod_{i=1}^{h} \kappa_i \).

(\( \prod \) is the c-semiring multiplication)

+ Path reservation
+ Optimal path routing (e.g., Floyd-Warshall)
SLAK: Service Level Agreement in Klaim

In collaboration with
R. De Nicola
Focussing on SLA

- Modern WAN applications quest for SLA specification and programming
- Consider WS:
  - programmers could drive the search phase of the required services by declaring their SLA constraint, and
  - language support guarantees satisfaction of requirements
  - Synchronizing = signing a “contract”
- KAOS is an attempt in this direction, but it has many concepts (e.g., links, costs, coordinators,...)
- We aim at refining KAOS in a calculus that abstracts
  - mechanisms for WAN programming...
  - ...together with SLA constraints
  - in a “flexible” framework
SLAK vs KAOS

Similarities wrt KAOS
- costs are c-semiring \[BMR97\]
- local (anonymous) communications
- remote spawning of processes

Simplifications wrt KAOS
- Bidirectional Links
- Incremental definition
- A different (syntactic) concept of “site”
- \( \Rightarrow \) self-links are dealt with uniformly
\[ \begin{align*}
N & ::= \text{Nets} \\
\quad s & ::=^L P \quad \text{Located Process} \\
\quad | & \quad N_1 \parallel N_2 \quad \text{Net composition} \\
\gamma & ::= \text{Prefixes} \\
\quad (x) & \quad \text{Input} \\
\quad | & \quad \text{new}(s_\kappa) \quad \text{Node creation} \\
\quad | & \quad \varepsilon(P)@s \quad \text{Process spawning} \\
P & ::= \text{Processes} \\
\quad \text{nil} & \quad \text{Null process} \\
\quad | & \quad \gamma.P \quad \text{Action prefixing} \\
\quad | & \quad \text{out}(t) \quad \text{Output}
\end{align*} \]
\[ N ::= \]
\[ s ::= \]
\[ L P \quad \text{Located Process} \]
\[ N_1 \parallel N_2 \quad \text{Net composition} \]
\[ \gamma ::= \]
\[ (x) \quad \text{Input} \]
\[ \text{new}(s_{\kappa}) \quad \text{Node creation} \]
\[ \varepsilon(P)@s \quad \text{Process spawning} \]
\[ P ::= \]
\[ \text{nil} \quad \text{Null process} \]
\[ \gamma.P \quad \text{Action prefixing} \]
\[ \text{out}(t) \quad \text{Output} \]
Local Transitions

(OUT) \[ s ::^L \text{out}(r) \xrightarrow{\langle r \rangle} s ::^L \text{nil} \]

(IN) \[ s ::^L (x).P \xrightarrow{\langle r \rangle} s ::^L P[r/x], \quad r \in S \]

(NEW) \[ s ::^L \text{new}(u_\kappa).P \xrightarrow{\text{new}(r)} s ::^{L \cup r_\kappa} P \parallel r ::\{u_1, s_\kappa\} \text{nil}, \quad r \notin \text{dom}(L) \]

(EVAL) \[ s ::^L \varepsilon(P)@t.Q \xrightarrow{s[P]@t} s ::^L Q \]

(SITE) \[ s ::^{L \cup r_\kappa} P \xrightarrow{\text{from}(r)} s ::^{L \cup r_\kappa} P \]
Global Transitions$^1$

\[
N_1 \xrightarrow{\alpha} N'_1 \\
N_1 \parallel N_2 \xrightarrow{\alpha} N'_1 \parallel N_2
\]

\[\text{bn}(\alpha) \cap \text{fn}(N_2) = \emptyset\]

\[
N_1 \xrightarrow{\langle r \rangle} N'_1 \\
N_2 \xrightarrow{\langle r \rangle} N'_2
\]

\[
N_1 \parallel N_2 \xrightarrow{\tau} N'_1 \parallel N'_2
\]
Global Transitions\textsuperscript{2}

\[
\begin{align*}
N_1 & \xrightarrow{s[P]@t_{\mathcal{R},\kappa_1}} N'_1 & N_2 & \xrightarrow{\text{from}(r)_{\mathcal{R}',\kappa_2}} N'_2 & \kappa_2 & = T(P) \\
N_1 \parallel N_2 & \xrightarrow{s[P]@t_{\mathcal{R}',\kappa_1 \ast \kappa_2}} N'_1 \parallel N'_2 \\
N_1 & \xrightarrow{s[P]@t_{\mathcal{R},\kappa_1}} N'_1 & N_2 & \xrightarrow{\text{from}(r)_{t,\kappa_2}} N'_2 & \kappa_2 & = T(P) \\
N_1 \parallel N_2 & \xrightarrow{s[P]@t_{t,\kappa_1 \ast \kappa_2}} N'_1 \parallel N'_2 \parallel t :: \{t_{\kappa_1 \ast \kappa_2}\} P
\end{align*}
\]
Correctness & Completeness

**Definition** Let \( N \) be a net and let \( s \) and \( t \) be two sites in \( N \). We denote with \( \mathcal{P}_N(s, t) \) the set of costs of the paths from \( s \) to \( t \) in \( N \).

**Theorem** Let \( N \) be a net, then

\[
N \xrightarrow{s[P]@} t, \kappa \implies \kappa \in \mathcal{P}_N(s, t)
\]

**Theorem** Given a net \( N \) and a process \( P \)

\[
N \xrightarrow{s[P]@} t, 1 \implies N \xrightarrow{s[P]@} t, \kappa
\]

for all \( \kappa \in \mathcal{P}_N(s, t) \) s.t. \( \kappa \models T(P) \)
A slight variation: \( P ::= \ldots \mid \varepsilon_\kappa(P)@s \)

(EVAL')
\[
\begin{array}{c}
s ::=^L \varepsilon(P)@t \overset{s[P]@t}{\to} s ::=^L \text{nil}
\end{array}
\]

(ROUTE')
\[
\begin{array}{c}
N_1 \overset{s[P]@t}{\to}_{r, \kappa_1} N'_1 & N_2 \overset{\text{from}(r)}{\to}_{r', \kappa_2} N'_2 & \kappa_2 \models T(P) \land \kappa \geq \kappa_1 \cdot \kappa_2
\end{array}
\]

\( t \neq r' \)

(LANDING')
\[
\begin{array}{c}
N_1 \parallel N_2 \overset{s[P]@t}{\to}_{r', \kappa_1 \cdot \kappa_2} N'_1 \parallel N'_2
\end{array}
\]

\[
\begin{array}{c}
N_1 \overset{s[P]@t}{\to}_{r, \kappa_1} N'_1 & N_2 \overset{\text{from}(r)}{\to}_{t, \kappa_2} N'_2 & \kappa_2 \models T(P) \land \kappa \geq \kappa_1 \cdot \kappa_2
\end{array}
\]

\[
\begin{array}{c}
N_1 \parallel N_2 \overset{s[P]@t}{\to}_{t, \kappa_1 \cdot \kappa_2} N'_1 \parallel N'_2 \parallel t ::=^{\{t_{\kappa_1 \cdot \kappa_2}\}} P
\end{array}
\]
Theorem If $N \xrightarrow{s[P_\kappa]@ t, \kappa'} N'$ then $\hat{N} \xrightarrow{s[P]@ t, \kappa'} \hat{N}'$ where $\hat{N}$ and $\hat{N}'$ are obtained by removing costs from $\varepsilon$ prefixes in $N$ and $N'$, respectively.
Dynamic Links

\[ \gamma ::= \]

... 

\| \text{link}(s_\kappa) \quad \text{Link} 

\| \text{accept}(s_\kappa) \quad \text{Accept} 

\| \text{disc}(s) \quad \text{Disconnect} 

\text{(LINK)} \quad s :: L \text{link}(r_\kappa).P \xrightarrow{\text{link}(r_\kappa)} s :: L \uplus r_\kappa \kappa' P 

\text{(ACCEPT)} \quad s :: L \text{accept}(r_\kappa).P \xrightarrow{\text{accept}(r_\kappa)} s :: L \uplus r_\kappa \kappa' P 

\text{(ACCEPT')} \quad s :: L \uplus r_\kappa P \xrightarrow{\text{accept}(r_\kappa)} s :: L \uplus r_\kappa P 

\text{(DISC)} \quad s :: L \text{disc}(r).P \xrightarrow{\tau} s :: L \setminus r P
Summing up...
Conclusions

Declarative Model of WAN programming

- Captures aspects of Web Services metaphor
- Suitable for representing several WAN network issues e.g.,
  - different models (Ambient, Klaim,...)
  - wireless communications
  - routing
- SLA as a Coordination Mechanism
- Toward formal techniques for SLA programming
Simplifying the hypergraph framework
Recently a very simple semantics has been found:
- neither $\Gamma$ nor $\pi$,
- just one inference rule.
Decreased expressive power but still an interesting model of WAN

Modelling GRID via SHR [AT03]
Modelling wireless phenomena [Tuo04]

Tools
- SHE (Synchronized Hyperedge Environment [CTT04])
- Gredi (GRammar EDItor)
Future work

- Extending synchronizing policies
  - Synchronization algebra = c-semiring?
  - Is it possible to use c-semirings for semantic web?
  - Rule matching problem = semantic search/bind of services
- Applying SLA to wireless routing problem
- Extending SHR futures to match GRID’s data-oriented issues
- Analyzing SLAK expressivity (wrt KAOS)
- Extending SLAK
- Model Checking SLAK by exploiting [Lor02, DL02]
- and Gredi + SHE for verifying wireless networks
- SLAK’s observational semantics
References


[BC99] Boumediene Bal, Henri E. Belkhouche and Luca Cardelli, editors. Workshop on Internet Program-


Rocco De Nicola, Gianluigi Ferrari, and Rosario Pugliese. KLAIM: A kernel language for agents in-


[LS00] Francesca Levi and Davide Sangiorgi. Controlling interference in ambients. In Conference Record of POPL’00: The 27th ACM SIGPLAN-SIGACT Sym-


