A Unifying Formal Basis for the Sensoria Languages

Ugo Montanari
Università di Pisa
Content

- Introduction
- Modeling service oriented architectures
- Modeling service oriented calculi
- A general view
Graph-based Modeling

- SENSORIA Architecture level
  - High-level specification languages and frameworks
    - UML4SOA
    - SRML
    - Service Modes
  - Guaranteed reconfigurations
- SENSORIA Service-Oriented Calculi level
  - Process algebras & Process calculi approaches
    - SAGAS
    - CaSPiS
  - Sound and Complete graphical encoding
Graphs Are Everywhere

- Use of diagrams / graphs is pervasive to Computer Science
ADR formulas:
- ADR = Designs + Term Rewriting
- Designs = Typed Hierarchical Graphs (with Interfaces)

ADR ingredients:
- Sorts: Vocabulary, Types (edge and node labels)
- Values: Designs (hierarchical graphs with interfaces)
- Operations: Graph-grammar-like rules
- Terms: proofs of construction
- Terms (with variables): partial Designs, partial proofs
- Axioms: properties of operations
- Membership predicates: additional style rules
- Rewrite rules: behaviour, reconfigurations
- Rewrite strategies: style conformance, style analysis, etc.
Hierarchical Graph Models

■ Why graph models?
  ▪ More natural for distributed systems
  ▪ Built-in structural axioms (e.g. name handling, AC axioms)
  ▪ Uniform treatment of most ordinary process algebras (e.g. via the SHR approach)

■ Why hierarchical?
  ▪ Nested structures: ambients, block structure, sessions, transactions, etc.
  ▪ Interaction between siblings, without referring to the closest common ancestor

■ Yet another graph model?
    ▪ place graph for localities and link graph for connectivity
    ▪ Semantics via reduction rules and minimal contexts
  ▪ Gs-monoidal graphs (gs-graphs) by Ferrari and Montanari (1997)
    ▪ Based on gs-monoidal categories by Corradini and Gadducci
  ▪ Top view & side view

■ ADR metamodel and Design Algebra:
  ▪ algebras of hierarchical graphs with graphical representations

■ Applications to Sensoria
  ▪ ADR modeling of UML4SOA, SRML, Modes
  ▪ DA modeling of process calculi: transaction workflows (Sagas), service sessions (CaSPiS)
  ▪ Reconcile Top view & Side view
Content

- Introduction
- Modeling service oriented architectures
- Modeling service oriented calculi
- A general view
Inductively defined, style-preserving reconfigurations
ADR for SRML

Inductively defined, well-formed, static & dynamic binding

Formal perational semantics via SHR rules (Synchronized Hyperedge Replacement)

Ugo Montanari, University of Pisa, A Unifying Formal Basis for the Sensoria Languages
ADR for Modes

Inductively defined, mode-driven, style-preserving reconfigurations
Content

- Introduction
- Modeling service oriented architectures
- Modeling service oriented calculi
- A general view
The Goal: Sound and Complete Encoding Also for Hierarchical Graphs

flow1

| a
| | b
| | [ c % d ]

graph1

flow2

| b
| | [ c % d ]
| | a

graph2

congruent

isomorphic
Design Algebra

$$D ::= L_\overline{x}[G]$$

$$G ::= 0 \mid x \mid l(\overline{x}) \mid G \mid G \mid (\nu x)G \mid D(\overline{x})$$

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(DA1)</td>
<td>$\text{fn}(0) = \emptyset$</td>
</tr>
<tr>
<td>(DA2)</td>
<td>$\text{fn}(l(\overline{x})) = [\overline{x}]$</td>
</tr>
<tr>
<td>(DA3)</td>
<td>$\text{fn}((\nu x)G) = \text{fn}(G) \setminus {x}$</td>
</tr>
<tr>
<td>(DA4)</td>
<td>$\text{fn}(L_\overline{x}[G]) = \text{fn}(G) \setminus [\overline{x}]$</td>
</tr>
<tr>
<td>(DA5)</td>
<td>$\text{fn}(x) = x$</td>
</tr>
<tr>
<td>(DA6)</td>
<td>$\text{fn}(G \mid H) = \text{fn}(G) \cup \text{fn}(H)$</td>
</tr>
<tr>
<td>(DA7)</td>
<td>$\text{fn}(D(\overline{x})) = \text{fn}(D) \cup [\overline{x}]$</td>
</tr>
</tbody>
</table>

Theorem: Sound and Complete axioms for hierarchical graphs
(also surjective map from terms to hierarchical graphs)

Theorem: Sound and Complete axioms for gs-graphs
CaSPiS encoding over the Design Algebra, I

\[
\begin{align*}
\llbracket A.P \rrbracket & \quad \text{def} \quad A.P \llbracket \llbracket \text{fn}(A.P) \rrbracket \rrbracket \\
\llbracket r \triangleright P \rrbracket & \quad \text{def} \quad \text{SES} \llbracket \llbracket P \rrbracket \rrbracket \llbracket r \rrbracket
\end{align*}
\]
CaSPiS encoding over the Design Algebra, II

\[ [P > (\exists x)Q] \stackrel{\text{def}}{=} x > Q[P] \left( \text{fn}(Q) \setminus \{x\} \right) \]
Transformation Rules for CaSPiS Reduction Semantics, III

\[ x > R \]

\[ \langle u \rangle . P''' \]

\[ \Rightarrow \]

\[ x > R \]

\[ [R\{u/x}\}]] \]

\[ P \equiv C[ (P'\langle u\rangle . P''') > (?x).R ] \]

\[ Q \equiv C[ R\{u/x}\} | ((P'\langle P'''\rangle) > (?x).R ) | \]

(PipelineSync)
Content

- Introduction
- Modeling service oriented architectures
- Modeling service oriented calculi
- A general view
A General View from a Personal Perspective

nominal calculi
coaalgebras on presheaves
ext. to resources, constrains

graph modeling
isom. vs. struct. axioms
DPO, SHR
SO hierarchical graphs

algebraic specifications
typing, refinement
(conditional) rewriting
SOS, De Simone

LTS, coalgebras
bisimilarity, final object
adequate modal logic

quantitative LTS
probabilistic/stochastic LTS
Markov chains, processes
non idempotency
coalgebras on metric spaces

behavioral types
sessions
transactions
Negotiate Commit Execute

process calculi
bialgebras
bisimilar. as congruence
process algebras
THANKS FOR THE ATTENTION