Graph Representation of Sessions and Pipelines for Structured Service Programming

Liang Zhao[1,2]

with Roberto Bruni[1] and Zhiming Liu[2]

[1]University of Pisa, Italy
[2]UNU-IIST, Macao SAR, China

FACS 2010

Introduction

Traditional process calculi

- successful in modeling concurrent systems, mobile systems
- modeling service systems?
- Problem: low level communication primitives, complexity of analysis

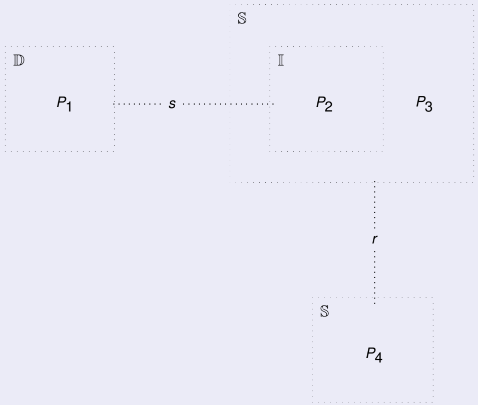
Calculus of Sessions and Pipelines (CaSPiS)

- Aspects: service autonomy, client-service interaction, orchestration
- Key notions: session (define interactions between two sides), pipeline (orchestrate the flow of data)
- Sessions and pipelines can be nested.
- Operational semantics: transition rules, silent transitions (reductions)

Introduction

Motivation

- hierarchical service system



- *vs.* textual expression: $s.P_1 | r \triangleright (\bar{s}.P_2 | P_3) | r \triangleright P_4$

Outline

1. **The calculus CaSPiS**
   - Syntax
   - Operational semantics (reduction)
2. **Algebra of hierarchical graphs**
   - Grammar and semantic model
   - Graph transformation by DPO
3. **Graph representation of CaSPiS**
   - Processes as designs
   - Graph transformation rules
   - Soundness and completeness

Outline

1. **The calculus CaSPiS**
   - Syntax
   - Operational semantics (reduction)

2. **Algebra of hierarchical graphs**
   - Grammar and semantic model
   - Graph transformation by DPO

3. **Graph representation of CaSPiS**
   - Processes as designs
   - Graph transformation rules
   - Soundness and completeness

Syntax

Syntax of CaSPiS

$$
\begin{array}{llll}
\text{Process} & P, Q & ::= & M \mid P|Q \mid s.P \mid \bar{s}.P \mid r \rhd P \mid (\nu n)P \mid P > Q \\
\text{Sum} & M & ::= & \mathbf{0} \mid (?x)P \mid \langle V \rangle P \mid \langle V \rangle^{\uparrow} P \mid M + M \\
\text{Value} & V & ::= & x \mid c
\end{array}
$$

- Example: $P_0 = time.\langle T \rangle | \overline{time}.(?x)\langle x \rangle^{\uparrow}$

Structural Congruence

$$
\begin{array}{rcll}
(P_1|P_2)|P_3 & \equiv_c & P_1|(P_2|P_3) & \\
M_1 + M_2 & \equiv_c & M_2 + M_1 & \\
(\nu n)\mathbf{0} & \equiv_c & \mathbf{0} & \\
r \rhd (\nu n)P & \equiv_c & (\nu n)(r \rhd P) & \text{if } n \neq r \\
 & \ldots & &
\end{array}
$$

Semantics

Contexts

- Dynamic operators: $(?x)[\cdot]$, $[\cdot]+M$, $s.[\cdot]$, $P > [\cdot]$
- Static contexts: $[\cdot]|Q$, $r \rhd [\cdot]$, $(\nu x)[\cdot]$, $[\cdot]|[\cdot]$, …

Basic Reductions

- (Sync): $C[s.P,\ \overline{s}.Q] \rightarrow (\nu r)C[r \rhd P,\ r \rhd Q]$
- (S-Sync): $C[r \rhd (P_0|\langle y\rangle P),\ r \rhd (?x)Q] \rightarrow C[r \rhd (P_0|P),\ r \rhd Q[y/x]]$
- (P-Sync): $C[(P_0|\langle y\rangle P) > (?x)Q] \rightarrow C[Q[y/x]|((P_0|P) > (?x)Q)]$
- …

Example

$$\begin{array}{rcl} & P_0 & = & time.\langle T\rangle|\overline{time}.(?x)\langle x\rangle^{\uparrow} \\ \rightarrow & P_1 & = & (\nu r)(r \rhd \langle T\rangle | r \rhd (?x)\langle x\rangle^{\uparrow}) \\ \rightarrow & P_2 & = & (\nu r)(r \rhd \mathbf{0} | r \rhd \langle T\rangle^{\uparrow}) \end{array}$$

Outline

Graph Grammar

Terms

Graph	G	$::=$	$\mathbf{0} \mid x \mid l(\vec{x}) \mid G\vert G \mid (\nu x)G \mid D\langle\vec{x}\rangle$
Design	D	$::=$	$L_{\vec{y}}[G]$

Free Node

- Free node: not restricted or exposed
- Example: $L_y[(\nu x)l(x, y)]\langle z\rangle$

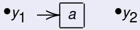
Type

- Fixed Type: each node x, edges label l, design label L
- Well-typedness: $l(\vec{x})$, $L_{\vec{y}}[G]\langle\vec{x}\rangle$

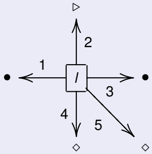
Semantic Model

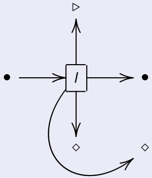
Interpretation of Terms

- $G_1 = a(y_1)|y_2$



Order of Tentacles of (Hyper-)edges

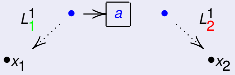




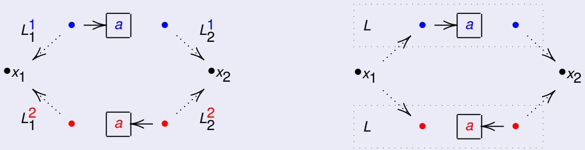
Semantic Model

Interpretation of Terms

- $G_2 = L_{(y_1,y_2)}[a(y_1)|y_2]\langle x_1, x_2\rangle$



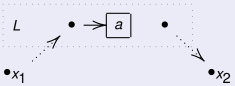
- $G_3 = L_{(y_1,y_2)}[a(y_1)|y_2]\langle x_1, x_2\rangle \mid L_{(y_1,y_2)}[y_1|a(y_2)]\langle x_1, x_2\rangle$

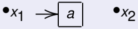


Semantic Model

Flat Design Edge

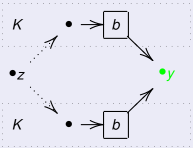
- $L_{(y_1,y_2)}[a(y_1)|y_2]\langle x_1, x_2\rangle$ *vs.* $F_{(y_1,y_2)}[a(y_1)|y_2]\langle x_1, x_2\rangle$

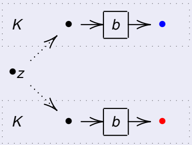




Node Sharing

- $K_x[b(x,y)]\langle z\rangle|K_x[b(x,y)]\langle z\rangle$ *vs.* $K_x[(\nu y)b(x,y)]\langle z\rangle|K_x[(\nu y)b(x,y)]\langle z\rangle$





Graph Transformation

Morphism and Pushout

- Morphism: a mapping ($m : G_1 \rightarrow G_2$) that preserves types of nodes, labels and tentacles of edges
- Pushout: a square of (four) morphisms that commute

Double Pushout (DPO) Rules

- $R : GL \overset{m_l}{\leftarrow} Gl \overset{m_r}{\rightarrow} GR$, or simply $GL|Gl|GR$

$$\begin{array}{ccccc} GL & \xleftarrow{m_l} & Gl & \xrightarrow{m_r} & GR \\ \downarrow{\scriptstyle m_1} & & \downarrow{\scriptstyle m_2} & & \downarrow{\scriptstyle m_3} \\ G & \xleftarrow{m'_l} & G'' & \xrightarrow{m'_r} & G' \end{array}$$

- Direct derivation: $G \Rightarrow_R G'$
- Derivation: a sequence of direct derivations, $G \Rightarrow^*_\Delta G'$

Outline

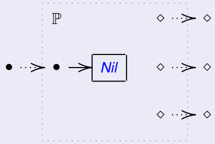
1. The calculus CaSPiS
   - Syntax
   - Operational semantics (reduction)

2. Algebra of hierarchical graphs
   - Grammar and semantic model
   - Graph transformation by DPO

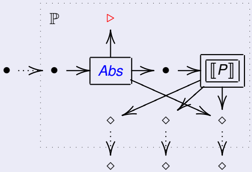
3. Graph representation of CaSPiS
   - Processes as designs
   - Graph transformation rules
   - Soundness and completeness

Graph Representation

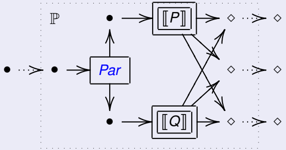
Nil, Abstraction and Parallel composition



$[\![\mathbf{0}]\!]$



$[\![(?x)P]\!]$



$[\![P|Q]\!]$

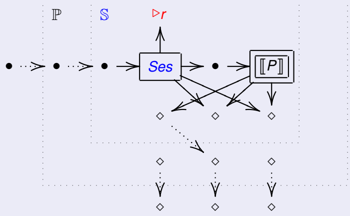
$$[\![\mathbf{0}]\!] \stackrel{\text{def}}{=} \mathbb{P}_{(p,i,o,t)}[i|o|t|Nil(p)]$$

$$[\![(?x)P]\!] \stackrel{\text{def}}{=} \mathbb{P}_{(p,i,o,t)}[(\nu\{p_1,x\})\big(Abs(p,x,p_1,i)|[\![P]\!]\langle p_1,i,o,t\rangle\big)]$$

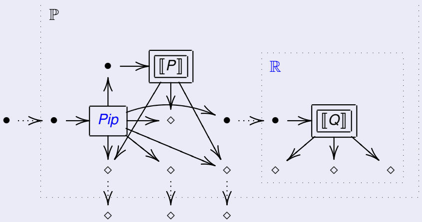
$$[\![P|Q]\!] \stackrel{\text{def}}{=} \mathbb{P}_{(p,i,o,t)}[(\nu\{p_1,p_2\}) \big(Par(p,p_1,p_2)|[\![P]\!]\langle p_1,i,o,t\rangle|[\![Q]\!]\langle p_2,i,o,t\rangle\big)]$$

Graph Representation

Session and Pipeline



$[\![r \rhd P]\!]$



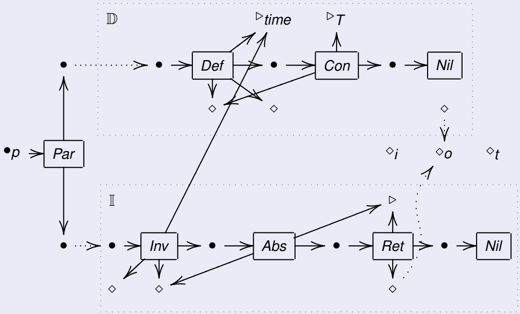
$[\![P > Q]\!]$

$$[\![r \rhd P]\!] \stackrel{\text{def}}{=} \mathbb{P}_{(p,i,o,t)}[i|t|\mathbb{S}_{(p,t)}[(\nu\{p_1,i_1,o_1\})\big(Ses(p,r,p_1,i_1,o_1)|[\![P]\!]\langle p_1,i_1,o_1,t\rangle\big)]\langle p,o\rangle]$$

$$[\![P > Q]\!] \stackrel{\text{def}}{=} \mathbb{P}_{(p,i,o,t)}[(\nu\{p_1,p_2,o_1\})\big(Pip(p,p_1,p_2,o_1,i,o,t)|[\![P]\!]\langle p_1,i,o_1,t\rangle|\mathbb{R}_p[(\nu\{i,o,t\})[\![Q]\!]\langle p,i,o,t\rangle]\langle p_2\rangle\big)]$$

Graph Representation

An Example

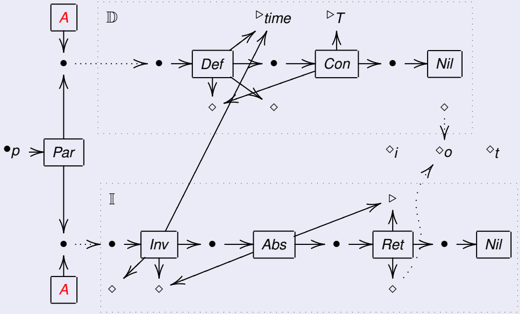


$[\![P_0]\!]\langle p, i, o, t\rangle$

- $P_0 = time.\langle T\rangle|\overline{time}.(?x)\langle x\rangle^{\uparrow}$

Graph Representation

Tagged Graph



$$[\![P_0]\!]^{\dagger}\langle p, i, o, t\rangle$$

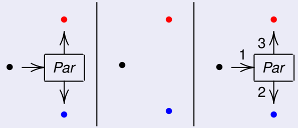
- $P_0 = time.\langle T\rangle|\overline{time}.(?x)\langle x\rangle^{\uparrow}$

Graph Transformation Rules

- Do congruence processes have the same graph representation? $[\![P_1|P_2]\!]^\dagger$ vs. $[\![P_2|P_1]\!]^\dagger$
- Δ_C: Rules for congruence

- What is the relation between $[\![P]\!]^\dagger$ and $[\![Q]\!]^\dagger$ with $P \longrightarrow Q$?
- Δ_R: Rules for reduction
- Auxiliary rules (tagging, garbage collection)

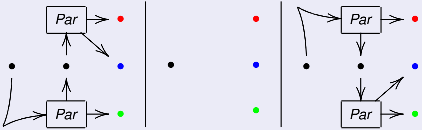
Rule for Congruence



- $[\![C[P_1|P_2]]\!]^\dagger \Rightarrow [\![C[P_2|P_1]]\!]^\dagger$

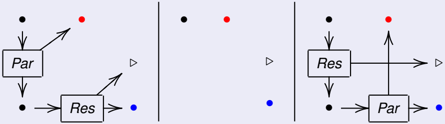
Graph Transformation Rules

Rule for Congruence (Cont.)



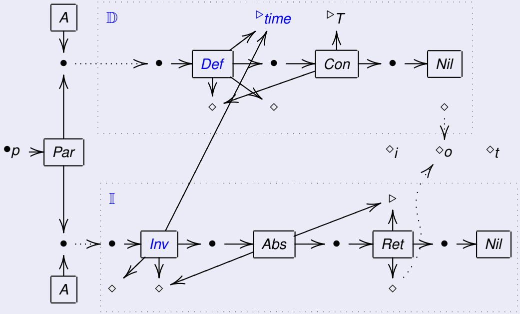
- $[\![C[(P_1|P_2)|P_3]]\!]^\dagger \Rightarrow [\![C[P_1|(P_2|P_3)]]\!]^\dagger$

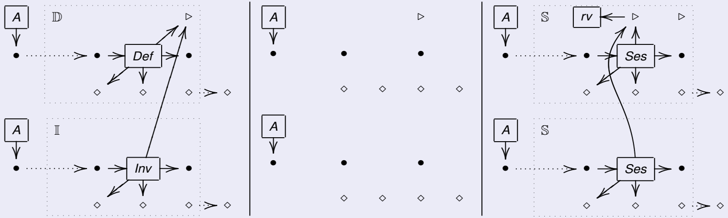
Rule for Congruence (Cont.)



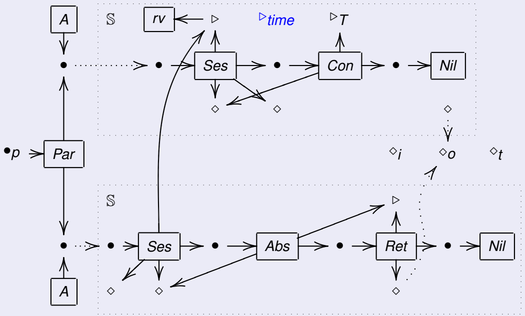
- $[\![C[P_1|(\nu n)P_2]]\!]^\dagger \Rightarrow [\![C[(\nu n)(P_1|P_2)]]\!]^\dagger$

Graph Transformation Rules

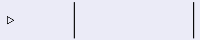




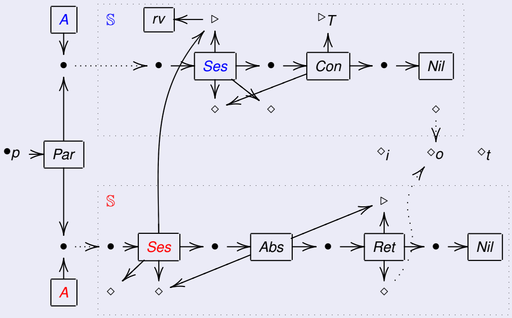
Graph Transformation Rules



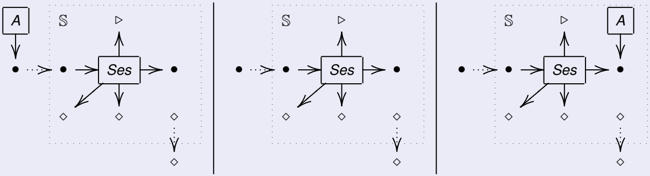
Garbage Collection Rule



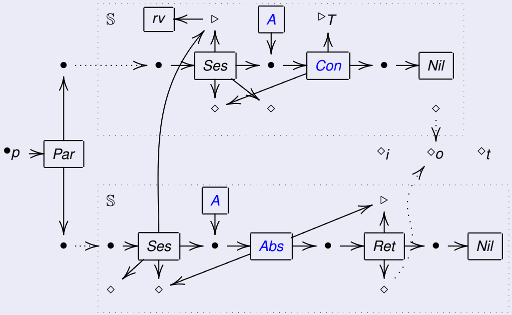
Graph Transformation Rules



Tagging Rule



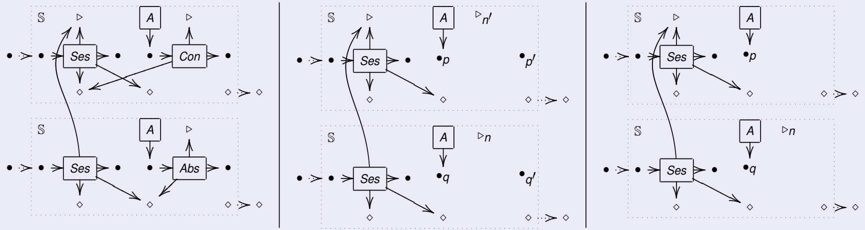
Graph Transformation Rules



- $[\![P_1]\!]^{\dagger}\langle p, i, o, t\rangle$
- $P_1 = (\nu r)(r \rhd \langle T\rangle | r \rhd (?x)\langle x\rangle^{\uparrow})$

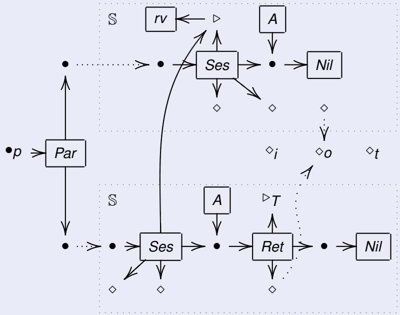
Graph Transformation Rules

Rule for Reduction



$$p/p' \to p,\ q/q' \to q; n/n' \to n$$

Graph Transformation Rules



- $[\![P_2]\!]^{\dagger}\langle p, i, o, t\rangle$
- $P_2 = (\nu r)(r \triangleright \mathbf{0} | r \triangleright \langle T\rangle^{\uparrow})$

Soundness and Completeness

Theorem (soundness w.r.t. congruence)

- $[\![P]\!]^\dagger \Rightarrow^*_{\Delta_C} G$ implies $G \equiv_d [\![Q]\!]^\dagger$ for some $Q \equiv_c P$

Theorem (completeness w.r.t. congruence)

- $P \equiv_c Q$ implies $[\![P]\!]^\dagger \Rightarrow^*_{\Delta_C} [\![Q']\!]^\dagger$ and $[\![Q]\!]^\dagger \Rightarrow^*_{\Delta_c} [\![Q']\!]^\dagger$ for some Q'

Conjecture (soundness w.r.t. reduction)

- $[\![P]\!]^\dagger \Rightarrow^*_{\Delta_A} [\![Q]\!]^\dagger$ implies $P \longrightarrow^* Q$
- difficulty: intermediate states

Conjecture (completeness w.r.t. reduction)

- $P \longrightarrow Q$ implies $[\![P]\!]^\dagger \Rightarrow^*_{\Delta_A} [\![Q']\!]^\dagger$ for some $Q' \equiv_c Q$
- difficulty: replications

Summary

What we have done

- a graph algebra: grammar, semantic model
- graph representation of CaSPiS processes (tagged graphs)
- graph transformation rules (DPO): congruence, reduction, auxiliary (tagging, garbage collection)
- soundness and completeness of DPO rules w.r.t. congruence

Future Work

- soundness and completeness of DPO rules w.r.t. reduction
- case study and implementation