An Algebra of Hierarchical Graphs

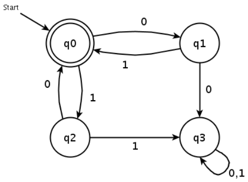
Roberto Bruni
(joint-work with Fabio Gadducci and Alberto Lluch)

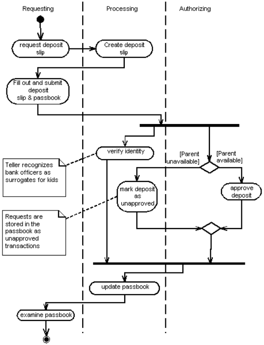
Department of Computer Science, University of Pisa
Software Engineering for Service-Oriented Overlay Computers

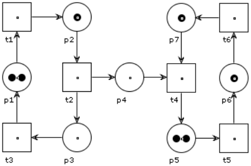
TGC 2010
5th Symposium on Trustworthy Global Computing
Munich (Germany), February 24-26, 2010.

Outline

Graphs are pervasive to Computer Science







Some advantages of graphs (up to isomorphism):

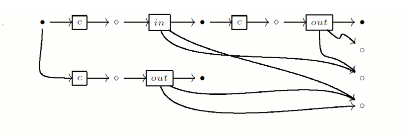
- names are helpful but inessential;
- element placement is helpful but inessential;
- connections between elements are essential

Algebras vs Graphs in distributed systems

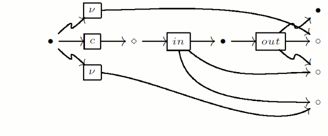
Goal

Flexible Graph-based representation of Service oriented systems

Mobile systems (names in π-calculus vs nodes in graphs)



$x(z).\overline{z}w.0 \mid \overline{x}x.0$



$(\nu x)(\nu y)x(z).\overline{z}y.0$

Service oriented systems

sessions, transactions, ambients: which graphs for containment?

Calculi vs Graphs

Algebraic

- Terms
  a | b

elements

Graph-based

- Graphs (diagrams)
  flat, hierarchical, etc.

Calculi vs Graphs

Algebraic

- Terms
  a | b
- Operations
  $\cdot|\cdot : \mathtt{W} \times \mathtt{W} \to \mathtt{W}$

elements

vocabulary

Graph-based

- Graphs (diagrams)
  flat, hierarchical, etc.
- Graph compositions
  Union, tensor, etc.

Calculi vs Graphs

Algebraic

- Terms
  $\mathtt{a} \mid \mathtt{b}$
- Operations
  $\cdot|\cdot : \mathtt{W} \times \mathtt{W} \to \mathtt{W}$
- Axioms
  $\mathtt{x} \mid \mathtt{y} \equiv \mathtt{y} \mid \mathtt{x}$

elements

vocabulary

equivalence

Graph-based

- Graphs (diagrams)
  flat, hierarchical, etc.
- Graph compositions
  Union, tensor, etc.
- Homomorphisms
  isomorphism, etc.

Calculi vs Graphs

Algebraic

- Terms
  $\mathtt{a} \mid \mathtt{b}$
- Operations
  $\cdot | \cdot : \mathtt{W} \times \mathtt{W} \to \mathtt{W}$
- Axioms
  $\mathtt{x} \mid \mathtt{y} \equiv \mathtt{y} \mid \mathtt{x}$
- Rewrite rules
  $\mathtt{a} \longrightarrow \mathtt{b}$

elements

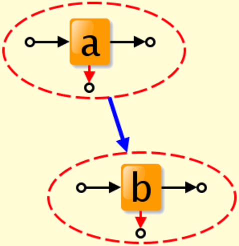
vocabulary

equivalence

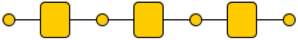
dynamics

Graph-based

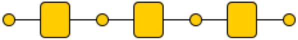
- Graphs (diagrams)
  flat, hierarchical, etc.
- Graph compositions
  Union, tensor, etc.
- Homomorphisms
  isomorphism, etc.
- Transformation rules

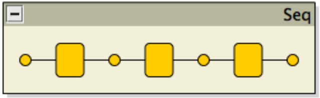


Which graphs?

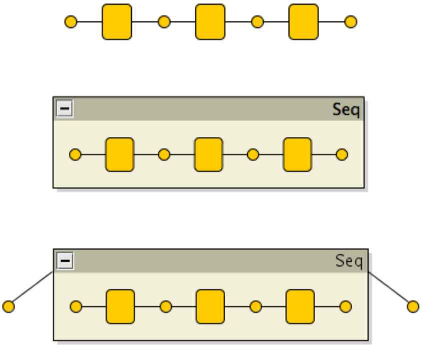


Which graphs?

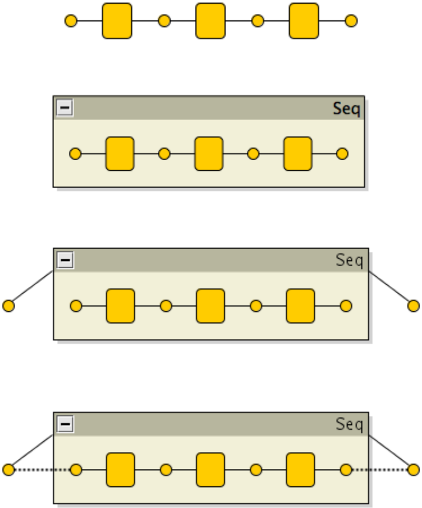




Which graphs?



Which graphs?



Main technical problem: representation distance

Definition 15 (processes). *Let $\mathcal{U}$ be a set of* names. *A* process *P is a term generated by the syntax*

$$P ::= \mathbf{0} \mid M \mid (\nu a)P \mid P \mid P$$
$$M ::= M + M \mid A.P$$

where $a, b \in$

Definition 15 (processes). *Let $\mathcal{U}$ be a set of* names. *A* process *P is a term generated by*

where $a, b \in$

grammar, structural congruence, etc.

Definition 22 (bigraph) ... $G = (V, ctrl, G^{\mathsf{T}}, G^{\mathsf{M}}) : I \to J$

where: $I = \langle m, X \rangle$ and ... ch combining a *width* (a finite

ordin

each

and 3

Definition 7 ... **rphisms).** *A hypergraph G is a triple $\langle E_G,$* ... *f edges, N_G is the set of nodes, and $t_G : E_G \to$*

Let G, H be hypergraphs. A (hypergraph) morphism $f : G \to H$ is a pair of functions $f_E : E_G \to E_H$, $f_N : N_G \to N_H$ preserving the tentacle function.

adjacency matrix, tuples, sets, morphisms

Main technical problem: representation distance

Definition 15 (processes). *Let* $\mathcal{U}$ *be a set of* names. *A* process P *is a term generated by the syntax*

$$P ::= \mathbf{0} \mid M \mid (\nu a)P \mid P \mid P$$
$$M ::= M + M \mid A.P$$

where $a, b \in$

Definition 1

generated by

where $a, b \in \mathcal{U}$

solution: graph algebras

$$[\![(\nu a)P]\!]_\Gamma = \begin{cases} [\![P]\!]_\Gamma & \textit{if } a \notin \mathtt{fn}(P) \\ (id_p \otimes \nu_c \otimes id_\Gamma) \circ [\![P\{^c/_a\}]\!]_{\{c\} \uplus \Gamma} & \textit{otherwise} \end{cases}$$

$$[\![P \mid Q]\!]_\Gamma = [\![P]\!]_\Gamma \otimes [\![Q]\!]_\Gamma \qquad [\![a(b).P]\!]_\Gamma = (in_{a,c} \otimes id_\Gamma) \circ [\![P\{^c/_b\}]\!]_{\{c\} \uplus \Gamma}$$

$$[\![0]\!]_\Gamma = 0_p \otimes 0_\Gamma \qquad [\![\overline{a}b.P]\!]_\Gamma = (out_{a,b} \otimes id_\Gamma) \circ [\![P]\!]_\Gamma$$

$$[\![\mathbf{0}]\!]_X = 1 \curlywedge X \qquad [\![P|Q]\!]_X = [\![P]\!]_X \curlywedge [\![Q]\!]_X \qquad [\![(x)P]\!]_X = \blacktriangle_x \circ [\![P]\!]_{X \uplus \{x\}}$$

$$[\![zx.P]\!]_X = \mathsf{get}^{x,z} \circ [\![P]\!]_X \qquad [\![\bar{z}x.P]\!]_X = \mathsf{send}^{x,z} \circ [\![P]\!]_X \qquad \text{where } x, z \in X$$

$$[(\nu a)P]^p_n = \mathtt{hide}_n([P\{\cdots/_a\}]_{n+1}) \qquad [\bar{i}j.P]^s_n = \mathtt{out}_{i,j,n}([P]^p_n)$$

$$[P \mid Q]^p_n = \mathtt{par}_n([P]^p_n, [Q]^p_n) \qquad [i(y).P]^s_n = \mathtt{in}_{i,n}([P\{^{n+1}/_y\}]^p_{n+1})$$

$$[0]^s_n = \mathtt{nil}_n \qquad [M + N]^s_n = \mathtt{choice}_n([M]^s_n, [N]^s_n)$$

Definition 22 (bigraph)
where: $I = \langle m, X \rangle$ and J
ordin
each
and 3

Definition 7 (
a triple $\langle E_G, N_G, t_G \rangle$ *such that* E_G *is the set of edges,* N_G *is the set of nodes, and* $t_G : E_G \to N_G^*$ *is the tentacle function.*

Let G*,* H *be hypergraphs. A (hypergraph) morphism* $f : G \to H$ *is a pair of functions* $f_E : E_G \to E_H$*,* $f_N : N_G \to N_H$ *preserving the tentacle function.*

Main result: a flexible, general intermediate language

- workflow language
- process calculus
- architecture description language
- etc.

- nested graphs
- gs-graphs
- bigraphs
- etc.

Main result: a flexible, general intermediate language

workflow language

process calculus

architecture description language

etc.

suitable graph algebra

nested graphs

gs-graphs

bigraphs

etc.

Outline

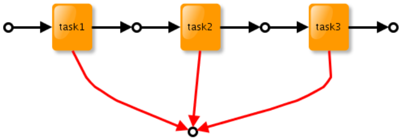
Running Example: Long running transactions

We shall consider a simple language for transactions with

- sequential composition;
- parallel (split-join) composition;
- compensating activity;
- scope of compensation.

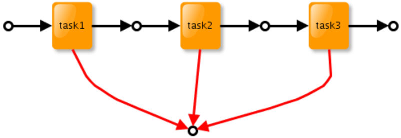
Analogous to the *Nested Sagas* of [BMM05].

Process terms and their graphical representations

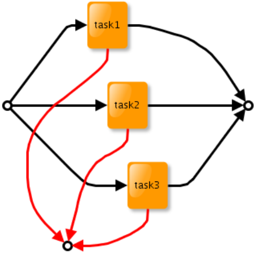


`task1 ; task2 ; task3`

Process terms and their graphical representations

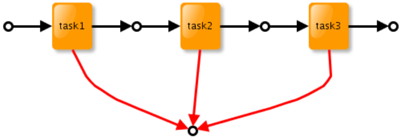


task1 ; task2 ; task3

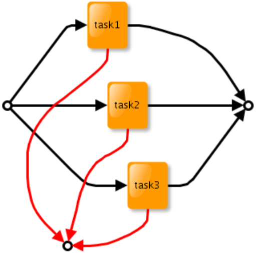


task1 | task2 | task3

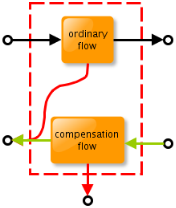
Process terms and their graphical representations



task1 ; task2 ; task3

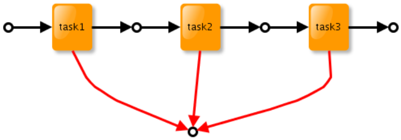


task1 | task2 | task3

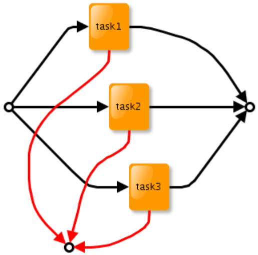


ordinary flow % compensation flow

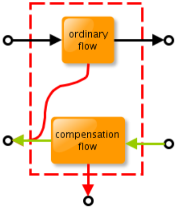
Process terms and their graphical representations



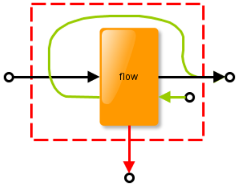
`task1 ; task2 ; task3`



`task1 | task2 | task3`



`ordinary flow % compensation flow`



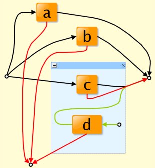
`[nested flow]`

Main technical goal: mapping coherent wrt. equivalence

flow1

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

graph1



Main technical goal: mapping coherent wrt. equivalence

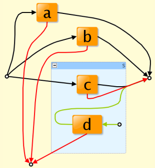
flow1

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

flow2

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

graph1



Main technical goal: mapping coherent wrt. equivalence

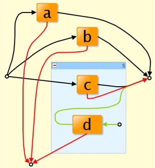
flow1

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

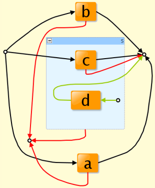
flow2

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

graph1



graph2



Outline

Graph layers

$\mathcal{N}$ universe of nodes
$\mathcal{A} = \mathcal{A}_{\mathcal{E}} \uplus \mathcal{A}_{\mathcal{D}}$ universe of edges

The set $\mathcal{L}$ of *graph layers* is the set of tuples $G = \langle N_G, E_G, t_G, F_G \rangle$ where

1. $E_G \subseteq \mathcal{A}$ is a (finite) set of edges,
2. $N_G \subseteq \mathcal{N}$ a (finite) set of nodes,
3. $t_G : E_G \to N_G^*$ a tentacle function, and
4. $F_G \subseteq N_G$ a set of free nodes.

The set $\mathcal{P} \subseteq \mathcal{L}$ of *plain graphs* contains those graph layers G such that $E_G \subseteq \mathcal{A}_{\mathcal{E}}$.
(standard notion of hypergraph plus a chosen set of *free* nodes)

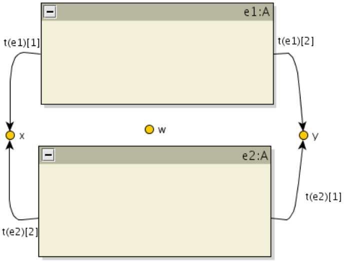
Hierarchical graphs

The set $\mathcal{H}$ of *hierarchical graphs* is the least set containing all the tuples $G = \langle N_G, E_G, t_G, i_G, x_G, r_G, F_G \rangle$ where

1. $\langle N_G, E_G, t_G, F_G \rangle$ is a graph layer;
2. $i_G : E_G \cap \mathcal{A}_\mathcal{D} \to \mathcal{H}$ (embedding function);
3. $x_G : E_G \cap \mathcal{A}_\mathcal{D} \to \mathcal{N}^*$ (exposure function), s.t. for all $e \in E_G \cap \mathcal{A}_\mathcal{D}$
   3.1 $\lfloor x_G(e) \rfloor \subseteq N_{i_G(e)} \setminus F_{i_G(e)}$, (free nodes of inner graphs not exposed)
   3.2 $|x_G(e)| = |t_G(e)|$, (same arity for exposure and tentacle func.)
   3.3 $\forall n, m \in \mathbb{N},\ \ x_G(e)[n] = x_G(e)[m]$ iff $t_G(e)[n] = t_G(e)[m]$;
4. $r_G : E_G \cap \mathcal{A}_\mathcal{D} \to (N_G \hookrightarrow \mathcal{N})$ is a renaming function, s.t. for all $e \in E_G \cap \mathcal{A}_\mathcal{D}$, $r_G(e)(N_G) = F_{i_G(e)}$.

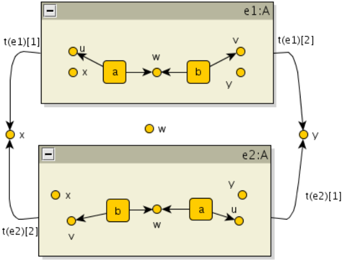
(for this talk, we can assume $r_G(e)$ is the ordinary inclusion)

A hierarchical graph and its simplified representation



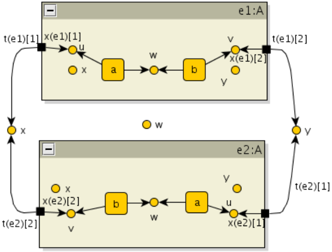
a graph layer (free nodes x and y)

A hierarchical graph and its simplified representation



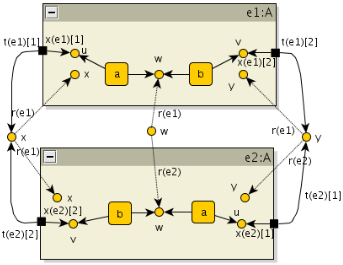
embedding function i_G

A hierarchical graph and its simplified representation



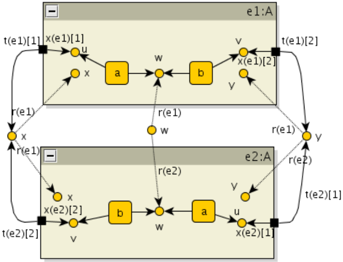
exposure function x_G

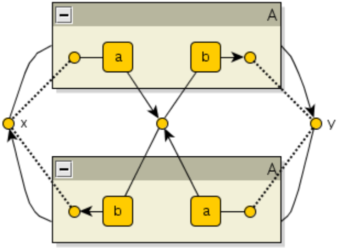
A hierarchical graph and its simplified representation



renaming function r_G

A hierarchical graph and its simplified representation

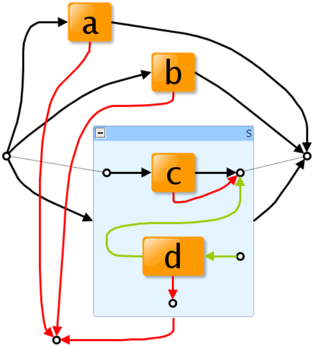


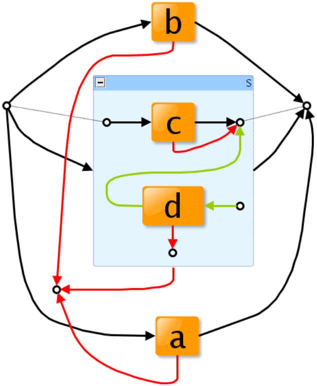


informal notation (free nodes x and y)

Hierarchical graph isomorphism

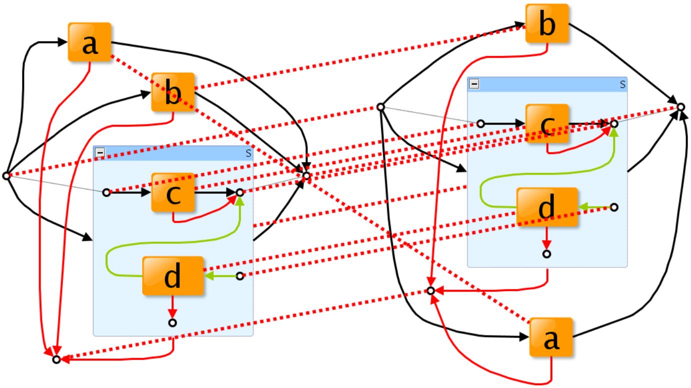
The actual model of hierarchical graphs has a suitable notion of isomorphism.





Hierarchical graph isomorphism

The actual model of hierarchical graphs has a suitable notion of isomorphism.



Hierarchical graph morphism (formally)

Let G, H be graphs such that $F_G \subseteq F_H$.
A *graph morphism* $\phi : G \to H$ is a tuple $\langle \phi_N, \phi_E, \phi_I \rangle$ where

1. $\phi_N : N_G \to N_H$ is a node morphism,
2. $\phi_E : E_G \to E_H$ an edge morphism, and
3. $\phi_I = \{\phi^e \mid e \in E_G \cap \mathcal{A}_\mathcal{D}\}$ a family of graph morphisms $\phi^e : i_G(e) \to i_H(\phi_E(e))$ such that
   - 3.1 $\forall e \in E_G$, $\phi_N(t_G(e)) = t_H(\phi_E(e))$, i.e. the tentacle function is respected;
   - 3.2 $\forall e \in E_G \cap \mathcal{A}_\mathcal{D}$, $\phi^e_N(x_G(e)) = x_H(\phi_E(e))$, i.e. the exposure function is respected;
   - 3.3 $\forall e \in E_G \cap \mathcal{A}_\mathcal{D}$, $\forall n \in N_G$, $\phi^e_N(r_G(e)(n)) = r_H(\phi_E(e))(\phi_N(n))$, i.e. the renaming function is respected;
   - 3.4 $\forall n \in F_G$, $\phi_N(n) = n$, i.e. the free nodes are preserved.

Outline

The syntax of the graph algebra

$$\mathbb{G}, \mathbb{H} \quad ::= \quad \mathbf{0}$$

the empty graph

The syntax of the graph algebra

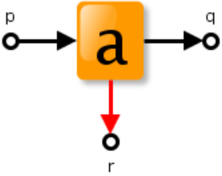
$$\mathbb{G}, \mathbb{H} \quad ::= \quad \mathbf{0} \mid x$$

a node called x

The syntax of the graph algebra

$$\mathbb{G}, \mathbb{H} \quad ::= \quad \mathbf{0} \mid x \mid l\langle \overline{x} \rangle$$

an (hyper)edge labelled with l attached to $\overline{x}$

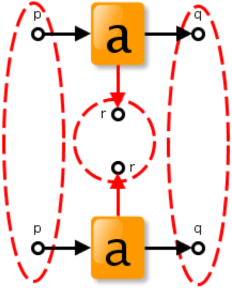


for instance, $\mathtt{a}\langle \mathtt{p,q,r} \rangle$

The syntax of the graph algebra

$$\mathbb{G}, \mathbb{H} \quad ::= \quad \mathbf{0} \mid x \mid l\langle \overline{x} \rangle \mid \mathbb{G}|\mathbb{H}$$

parallel composition: disjoint union up to common nodes

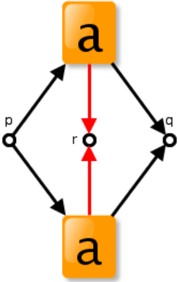


for instance, a⟨p,q,r⟩ | a⟨p,q,r⟩

The syntax of the graph algebra

$$\mathbb{G}, \mathbb{H} \quad ::= \quad \mathbf{0} \;\mid x \;\mid l\langle \overline{x} \rangle \;\mid \color{red}{\mathbb{G} | \mathbb{H}}$$

parallel composition: disjoint union up to common nodes

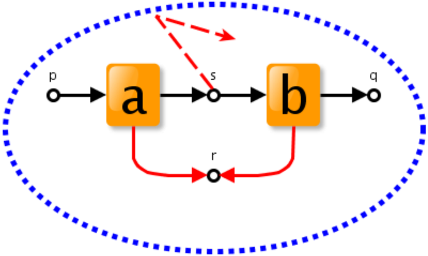


for instance, $\mathtt{a}\langle \mathtt{p},\mathtt{q},\mathtt{r} \rangle \;\; | \;\; \mathtt{a}\langle \mathtt{p},\mathtt{q},\mathtt{r} \rangle$

The syntax of the graph algebra

$$\mathbb{G}, \mathbb{H} \quad ::= \quad \mathbf{0} \ \mid x \ \mid l\langle \overline{x} \rangle \ \mid \mathbb{G}|\mathbb{H} \ \mid (\nu x)\mathbb{G}$$

declaration of a new node x

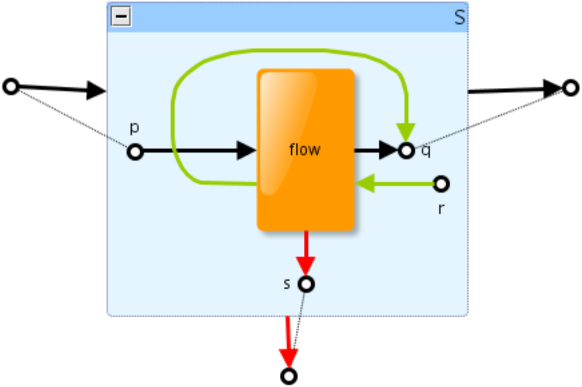


for instance, $(\nu \mathtt{s})\ (\mathtt{a}\langle \mathtt{p},\mathtt{s},\mathtt{r}\rangle \mid \mathtt{b}\langle \mathtt{s},\mathtt{q},\mathtt{r}\rangle)$

The syntax of the graph algebra

$$\mathbb{D} ::= L_{\overline{x}}[\mathbb{G}]$$
$$\mathbb{G}, \mathbb{H} ::= \mathbf{0} \mid x \mid l\langle\overline{x}\rangle \mid \mathbb{G}|\mathbb{H} \mid (\nu x)\mathbb{G}$$

graph G with interface of type L exposing $\overline{x}$

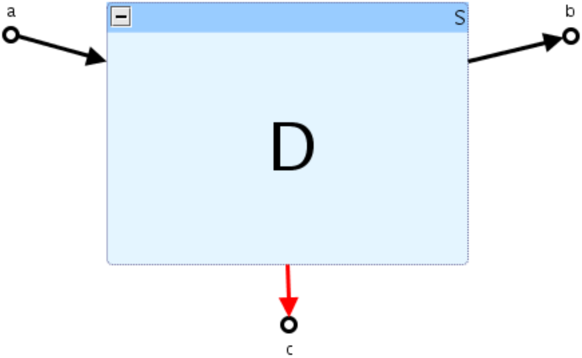


for instance, $\mathrm{S}_{\mathrm{p,q,s}}[(\nu \mathtt{r})\mathtt{flow}\langle \mathrm{p}, \mathrm{q}, \mathtt{r}, \mathrm{q}, \mathrm{s}\rangle]$

The syntax of the graph algebra

$$\mathbb{D} ::= L_{\overline{x}}[\mathbb{G}]$$
$$\mathbb{G}, \mathbb{H} ::= \mathbf{0} \mid x \mid l\langle\overline{x}\rangle \mid \mathbb{G}|\mathbb{H} \mid (\nu x)\mathbb{G} \mid \mathbb{D}\langle\overline{y}\rangle$$

a nested graph attached to $\overline{y}$

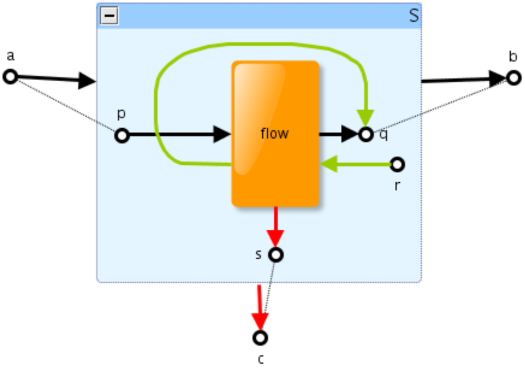


for instance, D⟨a,b,c⟩

The syntax of the graph algebra

$$\mathbb{D} ::= L_{\overline{x}}[\mathbb{G}]$$
$$\mathbb{G}, \mathbb{H} ::= \mathbf{0} \mid x \mid l\langle \overline{x} \rangle \mid \mathbb{G}|\mathbb{H} \mid (\nu x)\mathbb{G} \mid \mathbb{D}\langle \overline{y} \rangle$$

a nested graph attached to $\overline{y}$



for instance, $\mathtt{D}\langle \mathtt{a,b,c} \rangle$, with $\mathtt{D}{=}\mathtt{S}_{\mathtt{p,q,s}}[(\nu\mathtt{r})\mathtt{flow}\langle \mathtt{p,q,r,q,s} \rangle]$

Structural congruence axioms

Isomorphism is elegantly captured by structural axioms.

$$\mathbb{G} \mid \mathbb{H} \equiv \mathbb{H} \mid \mathbb{G} \tag{DA1}$$

$$\mathbb{G} \mid (\mathbb{H} \mid \mathbb{I}) \equiv (\mathbb{G} \mid \mathbb{H}) \mid \mathbb{I} \tag{DA2}$$

$$\mathbb{G} \mid \mathbf{0} \equiv \mathbb{G} \tag{DA3}$$

Structural congruence axioms

Isomorphism is elegantly captured by structural axioms.

$$
\begin{array}{rcll}
\mathbb{G} \mid \mathbb{H} & \equiv & \mathbb{H} \mid \mathbb{G} & \text{(DA1)} \\
\mathbb{G} \mid (\mathbb{H} \mid \mathbb{I}) & \equiv & (\mathbb{G} \mid \mathbb{H}) \mid \mathbb{I} & \text{(DA2)} \\
\mathbb{G} \mid \mathbf{0} & \equiv & \mathbb{G} & \text{(DA3)} \\
\\
(\nu x)(\nu y)\mathbb{G} & \equiv & (\nu y)(\nu x)\mathbb{G} & \text{(DA4)} \\
(\nu x)\mathbf{0} & \equiv & \mathbf{0} & \text{(DA5)} \\
\mathbb{G} \mid (\nu x)\mathbb{H} & \equiv & (\nu x)(\mathbb{G} \mid \mathbb{H}) \quad \text{if } x \notin fn(\mathbb{G}) & \text{(DA6)}
\end{array}
$$

Structural congruence axioms

Isomorphism is elegantly captured by structural axioms.

$$
\begin{array}{rcll}
\mathbb{G} \mid \mathbb{H} & \equiv & \mathbb{H} \mid \mathbb{G} & \text{(DA1)} \\
\mathbb{G} \mid (\mathbb{H} \mid \mathbb{I}) & \equiv & (\mathbb{G} \mid \mathbb{H}) \mid \mathbb{I} & \text{(DA2)} \\
\mathbb{G} \mid \mathbf{0} & \equiv & \mathbb{G} & \text{(DA3)} \\
\\
(\nu x)(\nu y)\mathbb{G} & \equiv & (\nu y)(\nu x)\mathbb{G} & \text{(DA4)} \\
(\nu x)\mathbf{0} & \equiv & \mathbf{0} & \text{(DA5)} \\
\mathbb{G} \mid (\nu x)\mathbb{H} & \equiv & (\nu x)(\mathbb{G} \mid \mathbb{H}) \quad \text{if } x \notin fn(\mathbb{G}) & \text{(DA6)} \\
\\
L_{\overline{x}}[\mathbb{G}] & \equiv & L_{\overline{y}}[\mathbb{G}\{^{\overline{y}}/_{\overline{x}}\}] \quad \text{if } \lfloor \overline{y} \rfloor \cap fn(\mathbb{G}) = \emptyset & \text{(DA7)} \\
(\nu x)\mathbb{G} & \equiv & (\nu y)\mathbb{G}\{^{y}/_{x}\} \quad \text{if } y \notin fn(\mathbb{G}) & \text{(DA8)}
\end{array}
$$

Structural congruence axioms

Isomorphism is elegantly captured by structural axioms.

$$
\begin{array}{rcllr}
\mathbb{G} \mid \mathbb{H} & \equiv & \mathbb{H} \mid \mathbb{G} & & \text{(DA1)} \\
\mathbb{G} \mid (\mathbb{H} \mid \mathbb{I}) & \equiv & (\mathbb{G} \mid \mathbb{H}) \mid \mathbb{I} & & \text{(DA2)} \\
\mathbb{G} \mid \mathbf{0} & \equiv & \mathbb{G} & & \text{(DA3)} \\
& & & & \\
(\nu x)(\nu y)\mathbb{G} & \equiv & (\nu y)(\nu x)\mathbb{G} & & \text{(DA4)} \\
(\nu x)\mathbf{0} & \equiv & \mathbf{0} & & \text{(DA5)} \\
\mathbb{G} \mid (\nu x)\mathbb{H} & \equiv & (\nu x)(\mathbb{G} \mid \mathbb{H}) & \text{if } x \notin fn(\mathbb{G}) & \text{(DA6)} \\
& & & & \\
L_{\overline{x}}[\mathbb{G}] & \equiv & L_{\overline{y}}[\mathbb{G}\{^{\overline{y}}/_{\overline{x}}\}] & \text{if } \lfloor \overline{y} \rfloor \cap fn(\mathbb{G}) = \emptyset & \text{(DA7)} \\
(\nu x)\mathbb{G} & \equiv & (\nu y)\mathbb{G}\{^{y}/_{x}\} & \text{if } y \notin fn(\mathbb{G}) & \text{(DA8)} \\
& & & & \\
x \mid \mathbb{G} & \equiv & \mathbb{G} & \text{if } x \in fn(\mathbb{G}) & \text{(DA9)} \\
L_{\overline{x}}[z \mid \mathbb{G}]\langle \overline{y} \rangle & \equiv & z \mid L_{\overline{x}}[\mathbb{G}]\langle \overline{y} \rangle & \text{if } z \notin \lfloor \overline{x} \rfloor & \text{(DA10)}
\end{array}
$$

Structural congruence axioms

Isomorphism is elegantly captured by structural axioms.

$$
\begin{array}{rcll r}
\mathbb{G} \mid \mathbb{H} & \equiv & \mathbb{H} \mid \mathbb{G} & & \text{(DA1)} \\
\mathbb{G} \mid (\mathbb{H} \mid \mathbb{I}) & \equiv & (\mathbb{G} \mid \mathbb{H}) \mid \mathbb{I} & & \text{(DA2)} \\
\mathbb{G} \mid \mathbf{0} & \equiv & \mathbb{G} & & \text{(DA3)} \\
& & & & \\
(\nu x)(\nu y)\mathbb{G} & \equiv & (\nu y)(\nu x)\mathbb{G} & & \text{(DA4)} \\
(\nu x)\mathbf{0} & \equiv & \mathbf{0} & & \text{(DA5)} \\
\mathbb{G} \mid (\nu x)\mathbb{H} & \equiv & (\nu x)(\mathbb{G} \mid \mathbb{H}) & \text{if } x \notin fn(\mathbb{G}) & \text{(DA6)} \\
& & & & \\
L_{\overline{x}}[\mathbb{G}] & \equiv & L_{\overline{y}}[\mathbb{G}\{^{\overline{y}}/_{\overline{x}}\}] & \text{if } \lfloor \overline{y} \rfloor \cap fn(\mathbb{G}) = \emptyset & \text{(DA7)} \\
(\nu x)\mathbb{G} & \equiv & (\nu y)\mathbb{G}\{^{y}/_{x}\} & \text{if } y \notin fn(\mathbb{G}) & \text{(DA8)} \\
& & & & \\
x \mid \mathbb{G} & \equiv & \mathbb{G} & \text{if } x \in fn(\mathbb{G}) & \text{(DA9)} \\
L_{\overline{x}}[z \mid \mathbb{G}]\langle \overline{y} \rangle & \equiv & z \mid L_{\overline{x}}[\mathbb{G}]\langle \overline{y} \rangle & \text{if } z \notin \lfloor \overline{x} \rfloor & \text{(DA10)}
\end{array}
$$

Axioms DA1–DA8 are rather *standard* and thus *intuitive* to those familiar with (nominal) process calculi.

Encoding

The encoding $[\![\cdot]\!]$, mapping (well-formed) terms into graphs, is the function inductively defined as (letting $[\![\mathbb{G}]\!] = \langle N_{\mathbb{G}}, E_{\mathbb{G}}, t_{\mathbb{G}}, i_{\mathbb{G}}, x_{\mathbb{G}}, r_{\mathbb{G}}, F_{\mathbb{G}}\rangle$)

$$
\begin{array}{rcl}
[\![\mathbf{0}]\!] & = & \langle \emptyset, \emptyset, \bot, \bot, \bot, \bot, \emptyset\rangle \\
[\![x]\!] & = & \langle \{x\}, \emptyset, \bot, \bot, \bot, \bot, \{x\}\rangle \\
[\![I\langle \overline{x}\rangle]\!] & = & \langle \lfloor \overline{x} \rfloor, \{e\}, e \mapsto \overline{x}, \bot, \bot, \bot, \lfloor \overline{x} \rfloor\rangle \\
[\![\mathbb{G} \mid \mathbb{H}]\!] & = & [\![\mathbb{G}]\!] \oplus [\![\mathbb{H}]\!] \\
[\![(\nu x)\mathbb{G}]\!] & = & \langle N_{\mathbb{G}}, E_{\mathbb{G}}, t_{\mathbb{G}}, i_{\mathbb{G}}, x_{\mathbb{G}}, r_{\mathbb{G}}, F_{\mathbb{G}} \setminus x\rangle \\
[\![L_{\overline{x}}[\mathbb{G}]\langle \overline{y}\rangle]\!] & = & \langle N_{\mathbb{G}}, \{e'\}, e' \mapsto \overline{y}, e' \mapsto [\![\mathbb{G}]\!] \oplus [\![\lfloor \overline{y} \rfloor]\!], e' \mapsto \overline{x}, \\
 & & e' \mapsto id_{N_{\mathbb{G}}}, (F_{\mathbb{G}} \setminus \lfloor \overline{x} \rfloor) \cup \lfloor \overline{y} \rfloor\rangle
\end{array}
$$

where $e \in \mathcal{A}_I$ and $e' \in \mathcal{A}_L$.

Main Result

It is worth to remark that the encoding is surjective, i.e. every graph can be denoted by a term of the algebra.

Theorem

Let G be a graph. Then, there exists a well-formed term $\mathbb{G}$ generated by the design algebra such that G is isomorphic to $[\![\mathbb{G}]\!]$.

Moreover, our encoding is sound and complete, meaning that equivalent terms are mapped to isomorphic graphs and vice versa.

Theorem

Let $\mathbb{G}_1$, $\mathbb{G}_2$ be well-formed terms generated by the design algebra. Then, $\mathbb{G}_1 \equiv \mathbb{G}_2$ if and only if $[\![\mathbb{G}_1]\!]$ is isomorphic to $[\![\mathbb{G}_2]\!]$.

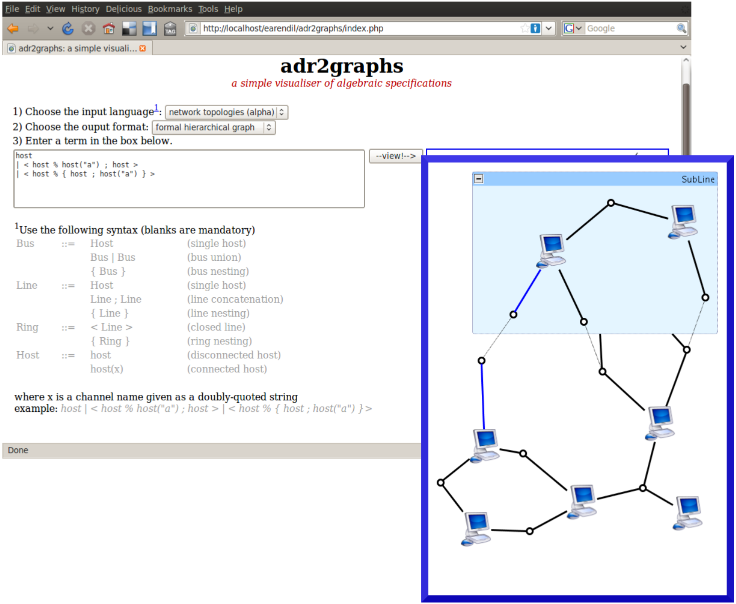
Outline

Concluding remarks

The approach...

- Grounds on widely-accepted models;
- Simplifies the graphical representation of complex systems;
- Hides the complexity of hierarchical graphs;
- Enables proofs by structural induction;
- Has been evaluated on various kinds of languages;
- Nesting and sharing features suitable for modelling SOC features such as transactions or sessions;
- Experimental implementation in RL/Maude (support for theorem proving, model checking, simulation, etc.);
- Offers a technique for complementing textual and visual notations in formal tools.

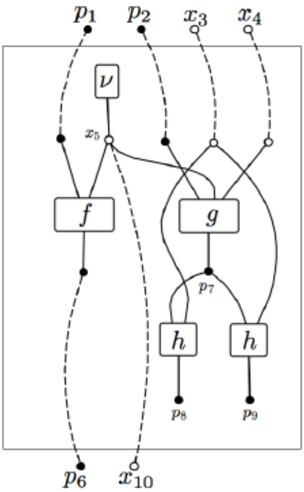
Visualizer: adr2graphs



Related work

GS-Graphs [CG99, FM00]

- syntactical structure, algebraic presentation
- flat (hierarchy-as-tree)



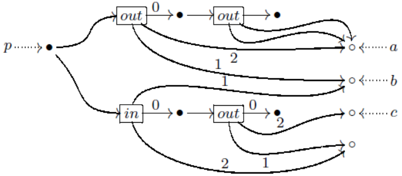
Related work

GS-Graphs [CG99, FM00]

- syntactical structure, algebraic presentation
- flat (hierarchy-as-tree)

Ranked Graphs [Gad03]

- node sharing, calculi encoding
- no composition interface, flat



Related work

GS-Graphs [CG99, FM00]

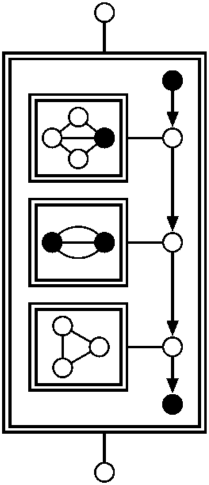
- syntactical structure, algebraic presentation
- flat (hierarchy-as-tree)

Ranked Graphs [Gad03]

- node sharing, calculi encoding
- no composition interface, flat

Hierarchical Graphs [DHP02]

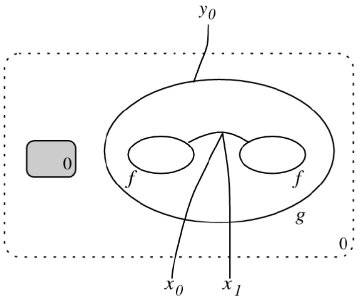
- basic model, composition interface
- no node sharing, no algebraic syntax



Related Work

Bigraphs [JM03]

- nesting + linking
- 2 overlapping structures, complex syntax, no composition interface, flat



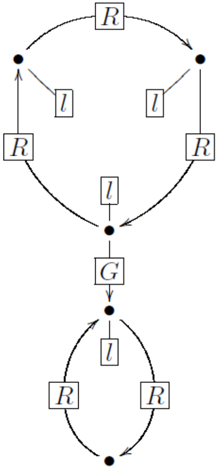
Related Work

Bigraphs [JM03]

- nesting + linking
- 2 overlapping structures, complex syntax, no composition interface, flat

Graph Algebra, SHR [CMR94]

- basic algebra
- flat, no composition interface



Credits and references I

[BMM05] Roberto Bruni, Hernán C. Melgratti, and Ugo Montanari.
Theoretical foundations for compensations in flow composition languages.
In Jens Palsberg and Martín Abadi, editors, *POPL*, pages 209–220. ACM, 2005.

[CG99] Andrea Corradini and Fabio Gadducci.
An algebraic presentation of term graphs, via gs-monoidal categories. applied categorical structures.
Applied Categorical Structures, 7:7–299, 1999.

[CMR94] Andrea Corradini, Ugo Montanari, and Francesca Rossi.
An abstract machine for concurrent modular systems: CHARM.
Theoretical Compututer Science, 122(1&2):165–200, 1994.

[DHP02] Frank Drewes, Berthold Hoffmann, and Detlef Plump.
Hierarchical graph transformation.
Journal on Computer and System Sciences, 64(2):249–283, 2002.

[FM00] G.L. Ferrari and U. Montanari.
Tile formats for located and mobile systems.
Inform. and Comput., 156(1-2):173–235, 2000.

[Gad03] Fabio Gadducci.
Term graph rewriting for the pi-calculus.
In Atsushi Ohori, editor, *Proceedings of the 1st Asian Symposium on Programming Languages and Systems*, volume 2895 of *Lecture Notes in Computer Science*, pages 37–54. Springer, 2003.

[JM03] O. H. Jensen and R. Milner.
Bigraphs and mobile processes.
Technical Report 570, Computer Laboratory, University of Cambridge, 2003.

Note: Some figures have been borrowed from the referred papers.