Implementing Binding Bigraphs

Arne John Glenstrup
Programming, Logics and Semantics (PLS) Group
IT University of Copenhagen
Denmark

Talk at LIX, Ecole Polytechnique, Paris
10th May 2007

Joint work with Troels Christoffer Damgaard, Lars Birkedal and Robin Milner

Motivation
—creating a bigraph manipulation tool proven correct in great detail

- A bigraphical reactive system (BRS) is a system state and a set of reaction rules specifying how the state can change
- We would like to create a tool for experimenting with bigraphs
- The first step is building an engine for simulating BRS'es
- We would like it to be proven correct in as much detail as possible
- Note: in this work we consider only binding bigraphs

Outline

1. introduction
   - Motivation
   - Basics

2. Implementing BRS
   - Matching
   - Reaction process steps
   - Implementation

3. Perspective
   - Related work
   - Future outlook
   - Conclusion

Bigraphical reactive systems
Reaction rules are triples $R \xrightarrow{\varrho} R'$

- an agent representing system state
- a set of reaction rules specifying how the state can change

Redex $R : I \rightarrow J$ is a bigraph with local inner face

Instantiation $\varrho :: I \rightarrow I'$ is a mapping of each location with its names in $I'$ to a location with its names in $I$

Reactum $R' : I' \rightarrow J$ is a bigraph with local inner face

\[
\varrho = [1 & [x_0^0 \mapsto x_0, x_2^0 \mapsto x_2],
         1 & [x_1^1 \mapsto x_1, x_2^1 \mapsto x_2]]
\]
Reaction = matching—instantiation—composition

- agent \( a \) is decomposed into context \( C \), redex \( R \), and parameter \( d \)
- \( d \) is instantiated to fit into reactum \( R' \), yielding \( d' \)
- new agent \( a' \) is composed by \( a = C(d_Z \otimes R')d' \)

Place graph normal forms
—unique up to permutation under merge

- For simplicity, we initially consider just place graphs
- We represent place graphs as compositions and products of terms
- We can represent any place graph using these normal forms:

\[
M ::= KP \quad \text{molecule}
\]
\[
S ::= \text{id}_1 \mid M \quad \text{sing. top-level node}
\]
\[
P ::= \text{merge}_n \left( \bigotimes_i^n S_i \right) \quad \text{prime}
\]
\[
B ::= \left( \bigotimes_i^n P_i \right) \pi \quad \text{place bigraph}
\]

- They are unique, up to permutation of \( \bigotimes_i^n S_i \)
- If a place graph \( B \) is expressible with \( \pi = \text{id} \), it is regular

Matching sentence over place graphs
—a 4-tuple relation using a regular redex

- We define a matching sentence that intuitively infers a match

\[\]
Inference system for place graph matches

- We can represent any bigraph using the following normal forms

- They are unique, up to permutation of $\bigotimes_i S_i$, and renaming
- If a bigraph $B$ or $D$ is expressible with $\pi = id$, it is regular

General inference tree structure
—redex above, context below SWITCH

- PAR, MERGE and ION rules match agent and context structure
- SWITCH moves the redex into context position
- For a parameter of $n$ primes, the inference tree will obviously contain $n$ applications of PRIME-AXIOM
- For a redex of $n$ nontrivial primes, the inference tree will contain $n$ applications of SWITCH
- Between any leaf and the root, SWITCH is applied at most once

Matching sentence over binding bigraphs
—a 7-tuple separating global and local wiring

- A matching sentence for binding bigraphs is a 7-tuple relation

$$\omega_X, \omega_R, \omega_C \vdash a, R \leftrightarrow C, d$$

where $\omega_X, \omega_R, \omega_C$ are wirings, and $a, R, C, d$ are discrete with local inner faces, all regular except $C$.

- It is valid iff

$$\frac{(id \otimes \omega_a)\alpha = (id \otimes \omega_C)(id_{\omega_{2V}} \otimes C)(id_{\omega_2} \otimes (id \otimes \omega_R) R)d.}$$

- Global wiring is in $\omega_X$, local wiring in $a, R, C, d$
- Inference rules are simply augmented...
Inference system for binding bigraph matching
—augmenting inference rules with wiring constructs

\[
\begin{align*}
\text{PERM} &: \omega_b, \omega_R, \omega_C \vdash a, \bigotimes_i^m P_i \vdash_C (\pi \otimes \text{id})d \\
\text{MERGE} &: \omega_b, \omega_R, \omega_C \vdash a, R \leftrightarrow C, d \\
\text{PAR} &: \omega_b, \omega_R, \omega_C \vdash (\omega \bigotimes \text{id})p, R \leftarrow (\omega \bigotimes \text{id})C, d \\
\text{ION} &: \omega_b, \omega_R, \omega_C \vdash ((\nu) / (\tilde{X}) \otimes \text{id})p, R \leftarrow ((\nu) / (\tilde{Z}) \otimes \text{id})p, d \\
\text{SWITCH} &: \omega_b, \omega_R, \omega_C \vdash p, \text{id} \leftarrow p, d \\
\text{PRIME-AXIOM} &: \omega_b, \omega_R, \omega_C \vdash \sigma \leftarrow \text{id}, \text{id} \leftarrow \text{id}, \sigma \leftarrow \text{id}, \text{id} \\
\end{align*}
\]

Formal correctness of the inference system
—it infers exactly all possible matches

- The inference system has been proven sound and complete
- That is, we can infer \( \omega_b, \omega_R, \omega_C \vdash a, R \leftrightarrow C, d \) exactly when it is valid, i.e., \( (\text{id} \otimes \omega_b)a = (\text{id} \otimes \omega_C)(\text{id}_{Z_{\text{dV}}} \otimes C)(\text{id}_Z \otimes (\text{id} \otimes \omega_R)d \)
- We can thus implement matching \( a = C(Z_{\text{dV}} \otimes R)d \) by

1. decomposing \( a = (\text{id} \otimes \omega_b)a' \) and \( R = (\text{id} \otimes \omega_R)R' \)
2. inferring a match \( \omega_b, \omega_R, \omega_C \vdash a', R' \leftrightarrow C', d \)
3. computing \( C = (\text{id} \otimes \omega_C)(\text{id}_{Z_{\text{dV}}} \otimes C') \)

- Problem: \( a, R, C, d \) are bigraphs, not bigraph terms
- Thus, bigraph axiom rules are missing, e.g., \( a = a \otimes \text{id}_0 \), \( a \otimes (b \otimes c) = (a \otimes b) \otimes c \) and \( \text{merge}(a \otimes b) = \text{merge}(b \otimes a) \)
- To make these rules explicit, we reformulate rules over terms \( a, R, C, d \)

Inference system for binding bigraph matching
—adding rules for handling wiring constructs

- 3 rules are added, to handle
  - the base case for bigraphs that are just wirings
  - moving a local substitution to global wiring when removing an abstraction
  - internal edges (closed links)

\[
\begin{align*}
\text{WIRING-AXIOM} &: y, X, y / X \leftarrow \text{id}_x, \text{id}_y \leftarrow \text{id}_x, \text{id}_e \\
\text{ABSTR} &: \sigma_a \otimes \omega_b, \sigma_c \otimes \omega_c \leftarrow \text{id}, R \leftarrow R \leftarrow R \leftarrow \text{id}, \sigma_a : Z \leftarrow W, \sigma_c : U \leftarrow W, P : U \otimes X \\
\text{CLOSE} &: \sigma_a, \sigma_b, \text{id}_{Y_a} \otimes \text{id} \leftarrow a, R \leftarrow C, d \\
\end{align*}
\]

Reformulating PAR for bigraph terms
—separating associative grouping from subterm matching

\[
\begin{align*}
\text{PAR} : \omega_b, \omega_R, \omega_C \vdash (\omega \bigotimes \text{id})p, R \leftarrow (\omega \bigotimes \text{id})C, d \\
\text{PAR} : \omega_b, \omega_R, \omega_C \vdash a, R \leftrightarrow C, d, S \rightarrow D, e \\
\end{align*}
\]

- The binary PAR rule is replaced by
  - an iterative rule taking \( n \) agent and redex parts, and
  - an equivalence rule grouping redex primes to match agent primes

\[
\begin{align*}
\text{PAR} : \omega_b, \omega_R, \omega_C \vdash a, R \leftrightarrow C, d \\
\end{align*}
\]
Reformulating MERGE for bigraph terms

—making molecule matching explicit

\[ \text{MER} : \omega_a, \omega_b, \omega_c \vdash a, R \to C, d \quad \text{a global} \]

\[ \omega_a, \omega_b, \omega_c \vdash (\text{merge} \otimes \text{id}) a, R \to (\text{merge} \otimes \text{id}) C, d \]

- The MERGE rule is replaced by a rule that makes explicit how agent molecules should be partitioned into sets
- Each redex prime is matched within one set of molecules

\[ \rho \in \vec{\rho}(n, m) \]

\[ \text{MER} : \sigma^a, \sigma^b, \sigma^c \vdash (\bigotimes_{j \in \rho} m_j) P \leadsto (\bigotimes_{i \in \rho} n_i S_{pi}) \vec{P}, \vec{Q} \]

\[ \sigma^a, \sigma^b, \sigma^c \vdash (\text{id} \otimes \text{merge}) \bigotimes_{i \in \rho} m_i, P \leadsto (\text{id} \otimes \text{merge}) \bigotimes_{i \in \rho} n_i S_i, \vec{Q} \]

(\vec{\rho}(n, m) \text{ is the set of partitionings of } 0, \ldots, n - 1 \text{ into } m \text{ sets.})

Inference system for bigraph term matching

- We get the following inference system:

\[ \begin{align*}
\text{PAX} & : \sigma : W \to Z \vdash \alpha : V \to W \quad \tau : X \to V \quad g : (X \to Z) \\
\sigma(d_2 \otimes \alpha_\tau), \text{id}_\alpha, \tau & \vdash g, (d_2 \otimes \alpha_\tau)(X)g \\
\text{ABS} & : \sigma_R^a \otimes \sigma_R^b \vdash g, P \leadsto G, \vec{Q} \quad \sigma_R^a : Z \to W \quad \sigma_R^b : U \to W \\
\sigma_R^a, \sigma_R^b & \vdash (\text{id} \otimes \sigma_R^a)(Z)g, P \leadsto (\text{id} \otimes \sigma_R^b)(U)G, \vec{Q} \\
\text{ION} & : \sigma : W \to Z \vdash \alpha : V \to W \\
\sigma_R^a, \sigma_R^b, \sigma & \vdash (\text{id} \otimes \sigma_R^a)(\vec{Y})n, P \leadsto (\text{id} \otimes \sigma_R^b)(\vec{Z})n, \vec{Q} \\
\alpha & = \gamma / \bar{u} \quad \sigma : \gamma / \bar{u} \to \sigma_R^a, \sigma_R^b, \sigma & \vdash (\text{id} \otimes \sigma)(W)G \leadsto \vec{U}, \vec{Q} \\
\text{SWX} & : \sigma \vdash \gamma / \bar{u} \vdash \gamma / \bar{u} \to \sigma \vdash g, \bigotimes_{i \in \rho} \text{id} \leadsto G, \vec{Q} \\
\text{G} & : \to (W \to Z) \quad \sigma : W \to U \\
\sigma^a, \sigma^b, \sigma^R \vdash g, (\text{id} \otimes \sigma)(W)G \leadsto \vec{U}, \vec{Q} \\
\end{align*} \]

Normal inference grammar

—restricts the order of rules in inference trees

- We let a normal inference be defined by \( D_N \) in this grammar:
Where is the nondeterminism hidden?

- There are several sources of nondeterminism:
  - Given normal inference, choice of rule is limited to $\text{D}_\varrho$ and $\text{D}'_\varrho$
  - Grouping (parenthisation) of tensor product in $\text{PAR}_\varrho$ rule
    \[
    \varrho \in \bar{\varrho}(n, m) \quad \vdash \quad \bigotimes_i^m \text{id} \otimes \text{merge} \otimes \bigotimes_j^k \varrho_i \varrho_j \quad \vdash \quad \bigotimes_i^m \text{id} \otimes \bigotimes_j^k \varrho_i \varrho_j
    \]
  - Partitioning of molecules by $\varrho$ in $\text{MER}_\varrho$ rule
    \[
    \gamma \in \bar{\gamma}(n, m) \quad \vdash \quad \bigotimes_i^m \text{id} \otimes \text{merge} \otimes \bigotimes_j^k \varrho_i \varrho_j \quad \vdash \quad \bigotimes_i^m \text{id} \otimes \bigotimes_j^k \varrho_i \varrho_j
    \]
  - Permutation of redex primes in $\text{PER}_\varrho$ rule
    \[
    \varpi \in \bar{\varpi}(n, m) \quad \vdash \quad \bigotimes_i^m \text{id} \otimes \text{merge} \otimes \bigotimes_j^k \varrho_i \varrho_j \quad \vdash \quad \bigotimes_i^m \text{id} \otimes \bigotimes_j^k \varrho_i \varrho_j
    \]
- Choice of $\sigma, \alpha, \tau$ in $\text{PAX}^\prime$
  \[
  \sigma : W \otimes Z \rightarrow \alpha : V \rightarrow W \quad \tau : X \rightarrow V \quad \gamma : (X \otimes Z)
  \]

### Normalisation

—normalises composition by traversing the term syntax tree

- We define normalisation relation $B \downarrow^B B'$ for
  - elementary bigraphs $\text{merge}_n, Y, X, Y, X, K_{\text{id}_Y}$ and $\pi$
  - operations: abstraction $(X)B$, product $\bigotimes_i B_i$, and composition $B_1B_2$
- Generally, we recursively normalise subterms, then recombine
- For tensor product we have
  \[
  \omega = \bigotimes_i \omega_i \quad P = \bigotimes_i \bigotimes_i P_i \quad D = \bigotimes_i D_i \equiv \bigotimes_i (\bigotimes_i \pi_i) \bigotimes_i \bigotimes_i \alpha_i
  \]
- Problem: $\bigotimes_i P_i$ will in general lead to name clash
- Thus, before normalisation, renaming must introduce fresh names

### Reaction process steps

- Matching is defined over regular bigraphs on binding discrete normal form
- Thus, the reaction cycle details reveal several processes
  - composing
  - renaming
  - instantiating
  - normalising
  - matching
  - regularising
  - output: $B'$ is term equivalent to $B$, with internal names renamed

### Renaming

—introduces fresh internal names where possible

- Unfortunately, we cannot replace every name by a fresh: Outer and inner names must not change, or we would get a different bigraph!
- We cannot even require that all term internal names are unique: a normalised subterm can contain several instances of a name, as $\text{id}_Y$ is part of the normal form!
- We thus define a function $\text{linknames}$ recursively over bigraphs
  - When $\text{linknames}$ is defined, normalisation will not name clash
  - Further, we ensure normalisation of subterms preserves $\text{linknames}$
- and inductively renaming judgment $\ising U \vdash \alpha, B \downarrow^B B', \beta \dashv V$, where
  - input: $U$ is a set of used names, $\alpha$ renames $B'$'s outer names to those of $B'$
  - output: $\beta$ renames $B'$'s outer names to those of $B'$, $V$ extends $U$ with names used in $B'$
  - $B'$ is term equivalent to $B$, with internal names renamed
  - $\text{linknames}(B')$ is defined, so normalisation of $B'$ will succeed
Splitting permutations

—into major and minor components, based on a list of local name set lists

- Match inference is only defined for regular bigraph terms
- Based on $\vec{X}$, we thus define $\pi^X$ and $\pi_i^X$ that split $\pi$ into one major and some minor permutations such that $(\bigotimes \pi^X_i)(\vec{X} \cdot \pi^X) = \pi$

![Diagram showing permutation splitting](image)

$\pi^X = [\{\}, \{\}, \{\}, \ldots ]$

Regularisation

—recursively splits $\pi$ and pushes it into subterms

- Based on permutation splitting, we define regularisation inductively over term syntax
- The only nontrivial rule is the name-discrete prime case:

$S_i : \langle m_i, \vec{X}_i \rangle \rightarrow J_i \pi = \pi^X_i \pi^X_{i'} \mapsto S_{i'}((X)(\text{id}_Y \otimes \text{merge}_n) \bigotimes_{i \in n} S_i)\pi_{i'} \mapsto (X)(\text{id}_Y \otimes \text{merge}_n) \bigotimes_{i \in n} S_{i'}(\pi_i)\pi_{i'}$

- If $\pi^X_i$ or $\pi^X_{i'}$ are undefined, the bigraph term does not represent a regular bigraph

Command line tool

—running as an extension of SMLNJ interactive command line

- Based on the matcher engine, we have made a command line interface, running as an extension of, e.g. the SMLNJ interactive command line:

```sml
- val K = active0 "K";
val K = K : 1 -> 1 : bgval
- val L = active0 "L";
val L = L : 1 -> 1 : bgval
- print_mv(match_v{agent = K o merge(2) o (L o <-> * K o <->),
    redex = K}) handle e => explain e;
val it = () : unit
```

- `match_v` returns a lazy list of matches, `print_mv` prints this list

Online web demo


BPLweb
Related work

- Jean Krivine et al.: Kappa Calculus, implemented as SimpIx
  - Efficient implementation
  - Different model than bigraphs, e.g., no nesting of nodes
- Hildebrandt et al.: Distributed Reactive XML.
  - Modelling BRS'es in XML using XPath, XQuery etc.
  - Until now only for pure bigraphs
- And, of course, general graph pattern matching work
  - Often does not handle redex parameters
  - Often does not handle wide redexes
  - Does not exploit layered (tree+link) bigraph structure

Future outlook

- More efficient implementation
- Graph based algorithm (subtree isomorphism, ...)
  - but we lose formally proven correctness
- Pre-computing approximation of possible matching locations
- Tacticals for controlling reaction patterns
- Graphical front & back ends
- Adding stochastic rates
- Implementing local bigraphs

Conclusion

- We have implemented a BRS tool that has been proven correct in great detail
- Key techniques include
  - inferring matching sentences by inference rules
  - defining matching sentences over bigraph terms
  - defining a grammar for normal inferences
  - normalising
  - regularising
- We have made command line and web demo front ends
- Implementing helped detect errors in the theory!
- Matching paper:
The ION rule illustrated

\[
\begin{align*}
\text{ION} & : \omega_a, \omega_R, \omega_C \vdash (\langle \vec{y} \rangle/\langle \vec{X} \rangle \otimes \text{id}) p, R \hookrightarrow (\langle \vec{y} \rangle/\langle \vec{Z} \rangle \otimes \text{id}) p, d \quad \alpha = \vec{y}/\vec{u} \quad \sigma : \{\vec{y}\} \rightarrow \\
& \sigma \parallel \omega_a, \omega_R, \sigma \alpha \parallel \omega_C \vdash (K_{\vec{y}(\vec{u})} \otimes \text{id}) p, R \hookrightarrow (K_{\vec{y}(\vec{u})} \otimes \text{id}) p, d
\end{align*}
\]

The CLOSE rule illustrated

\[
\begin{align*}
\text{CLOSE} & : \sigma_a, \sigma_R, \text{id} y_a \otimes \sigma_C \vdash a, R \hookrightarrow C, d \quad \sigma_C : \rightarrow Y \uplus Y_C \\
& \sigma_C \vdash (\text{id} \otimes (Y_R \uplus Y_C)) \sigma_a, (\text{id} \otimes (Y_R \uplus Y_C)) \sigma_R, (\text{id} \otimes (Y_R \uplus Y_C)) \sigma_C \vdash a, R \hookrightarrow C, d
\end{align*}
\]