Non-deterministic Interaction Nets

Article · January 1999

Vladimir Alexiev
Ontotext AD

Non-deterministic Interaction Nets

PhD Thesis by

Vladimir Alexiev
University of Alberta

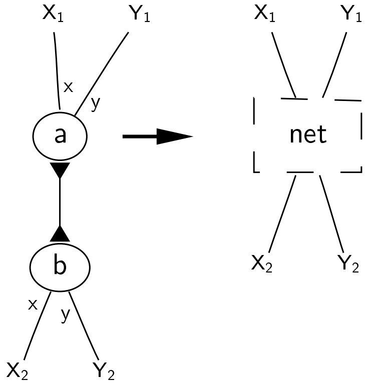
June 28, 1999

1 Overview

- Interaction Nets
    - IN Example: List Processing
  - Linearity: δ and ϵ Nodes
    - IN Example: Unary Arithmetics
  - Application: SK Combinators
- Non-determinism in IN
  - IN with Multiple Reduction Rules (INMR)
  - IN with Multiple Principal Ports (INMPP)
    - INMPP Example: Queue Merger
  - IN with MultiplePorts (INMP)
    - INMP Example: Variable (Reference)
  - IN with Multiple Connections (INMC)
    - INMC Example: Process Graphs

- Inter-representation of Non-Deterministic Models
  - INMPP as INMC: Port Diamonds
  - INMR as INMP: Self-Commitment
  - INMPP as INMP: Marker Nodes
  - INMC as INMP: Explicit Connectors
- Representing the π-calculus in MIN=INMP+INMPP
  - The π-calculus
    - Reduction Rules
  - MIN$_\pi$ Nodes, the Translation
  - MIN$_\pi$ Rules
    - Send/Receive
    - Input/Output
    - Blocking and Unblocking
  - Completeness and Soundness

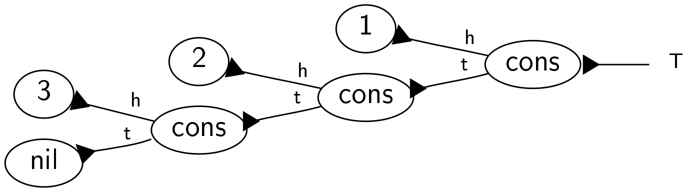
2 Interaction Nets



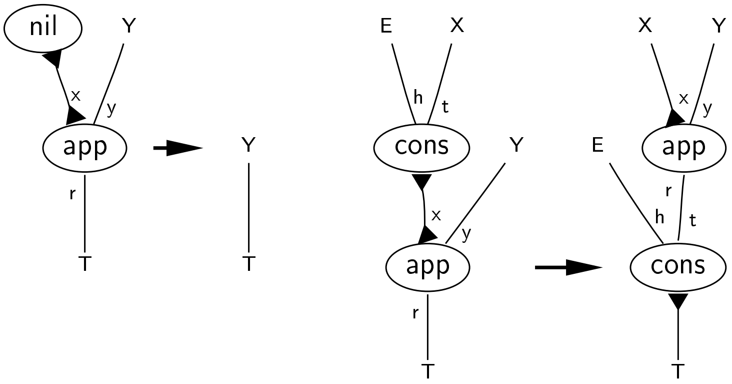
Lafont (1990), inspired by Proof Nets of Linear Logic.

Principal ports, linearity, binary local interaction, preserves interface.

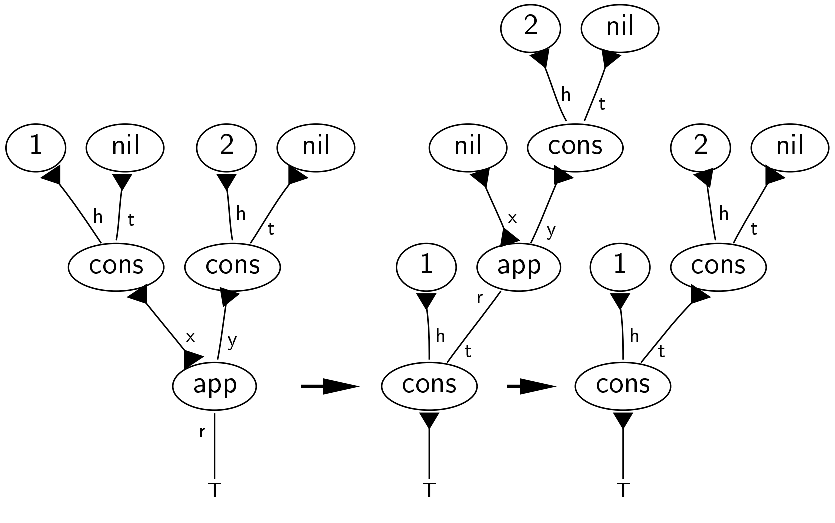
2.1 IN Example: List Processing



List is represented as a chain of cons nodes

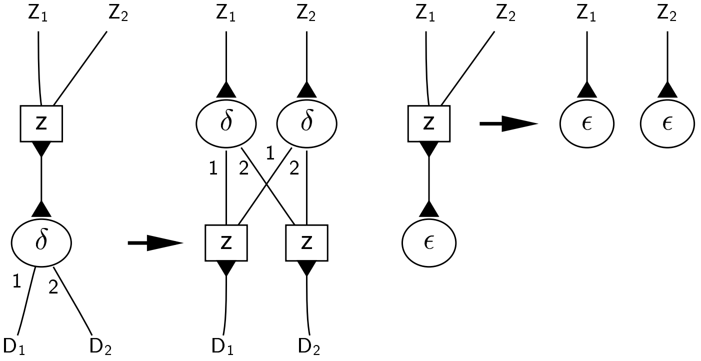


Rules for the append operation



Example: append lists (1) and (2)

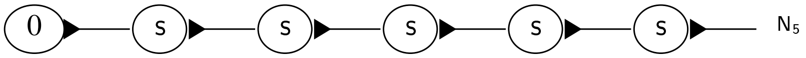
2.2 Linearity: δ and ϵ Nodes



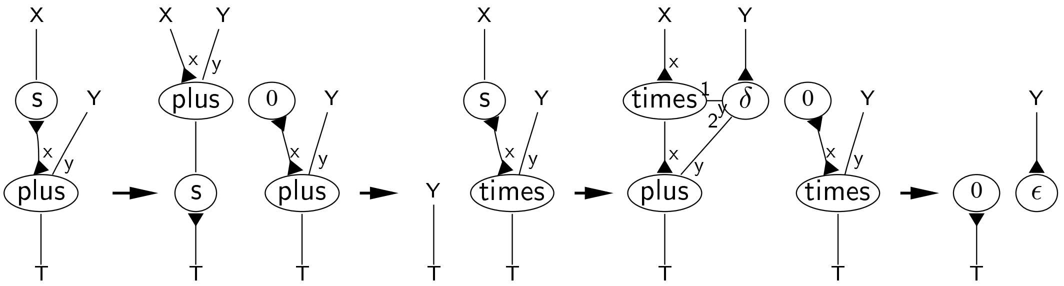
(Here z is any binary node.)

To duplicate a net, use δ.
To erase a net, use ϵ.

2.2.1 IN Example: Unary Arithmetics



The number 5 in unary notation



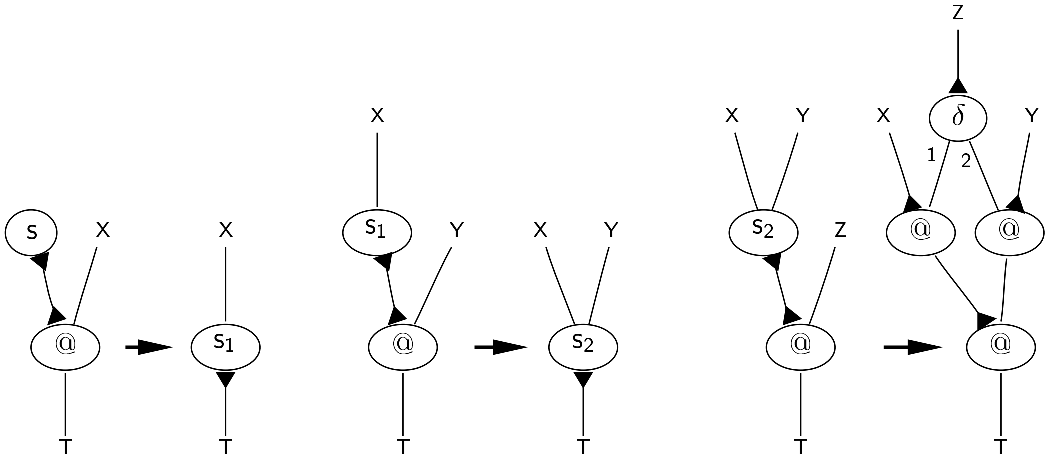
Arithmetic rules, corresponding to

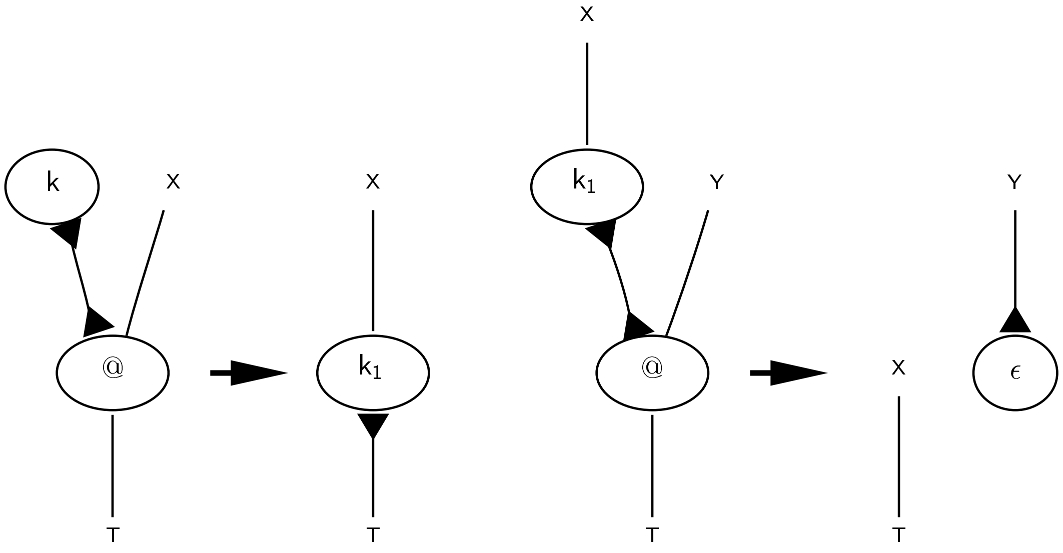
$$sx + y = s(x + y) \quad 0 + y = y \quad sx \times y = x \times y + y \quad 0 \times y = 0.$$

2.3 Application: SK Combinators

SK Combinators: $Sxyz = xz(yz)$, $Kxy = x$

Make application explicit: $@(@(@(\mathsf{s},\mathsf{X}),\mathsf{Y}),\mathsf{Z}) = @(@(\mathsf{X},\mathsf{Z}),@(\mathsf{Y},\mathsf{Z}))$





$@(@(\mathsf{k},\mathsf{X}),\mathsf{Y}) = \mathsf{X}$

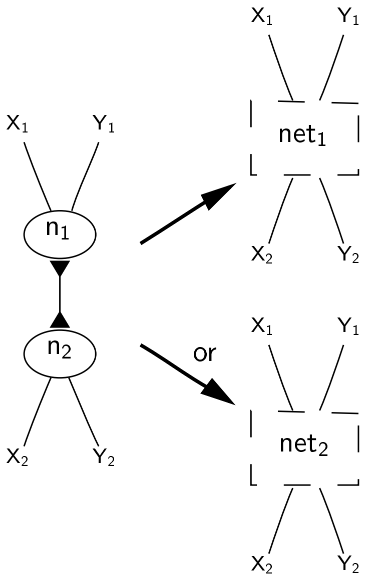
3 Non-determinism in IN

Traditional IN are *confluent*, therefore are limited to deterministic, functional programming. To represent agents, objects, processes, non-determinism, we need to break the confluence.

Several possible ways to do that:

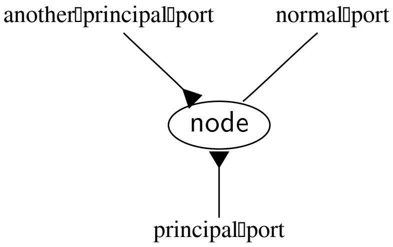
- IN with Multiple Reduction Rules (INMR)
- IN with Multiple Principal Ports (INMPP)
- IN with MultiplePorts (INMP)
- IN with Multiple Connections (INMC)

3.1 IN with Multiple Reduction Rules (INMR)

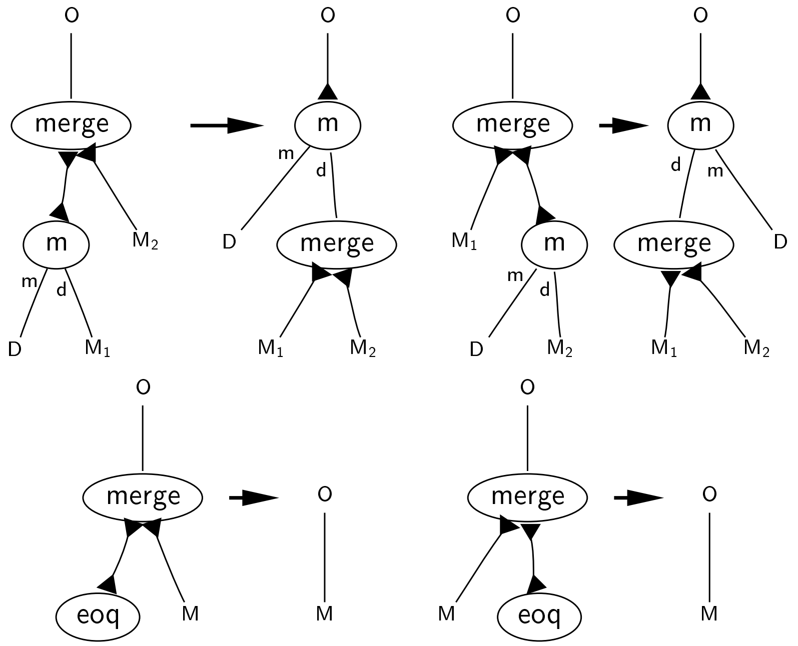


Turns out that we can limit ourselves to reflexive asymmetric rules only: $\mathsf{n} \bowtie \mathsf{n} \to \mathsf{net}$, where “net” is not symmetric.

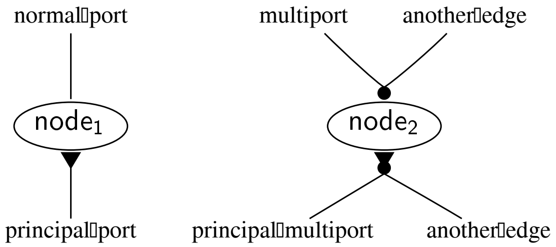
3.2 IN with Multiple Principal Ports (INMPP)



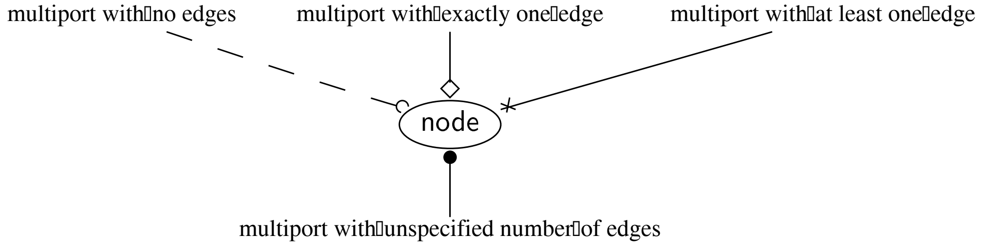
3.2.1 INMPP Example: Queue Merger



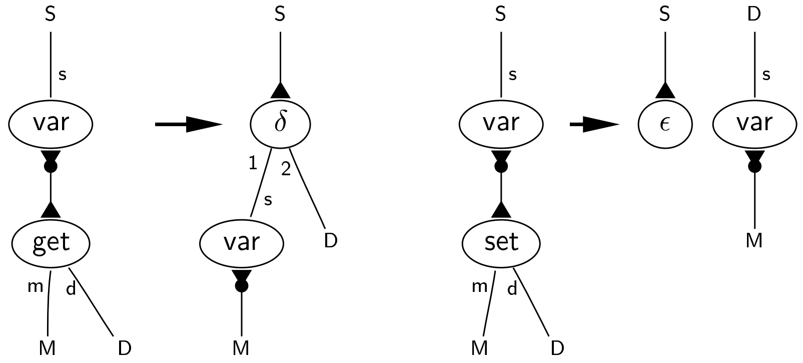
3.3 IN with MultiplePorts (INMP)



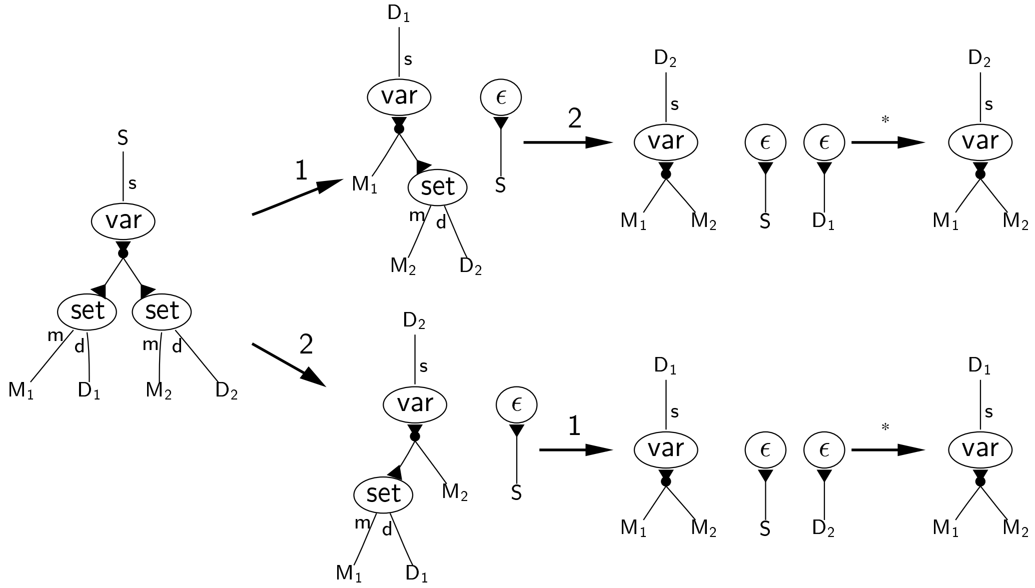
We also often need to specify the applicability of rules with *arity constraints*


multiport with no edges
multiport with exactly one edge
multiport with at least one edge
node
multiport with unspecified number of edges

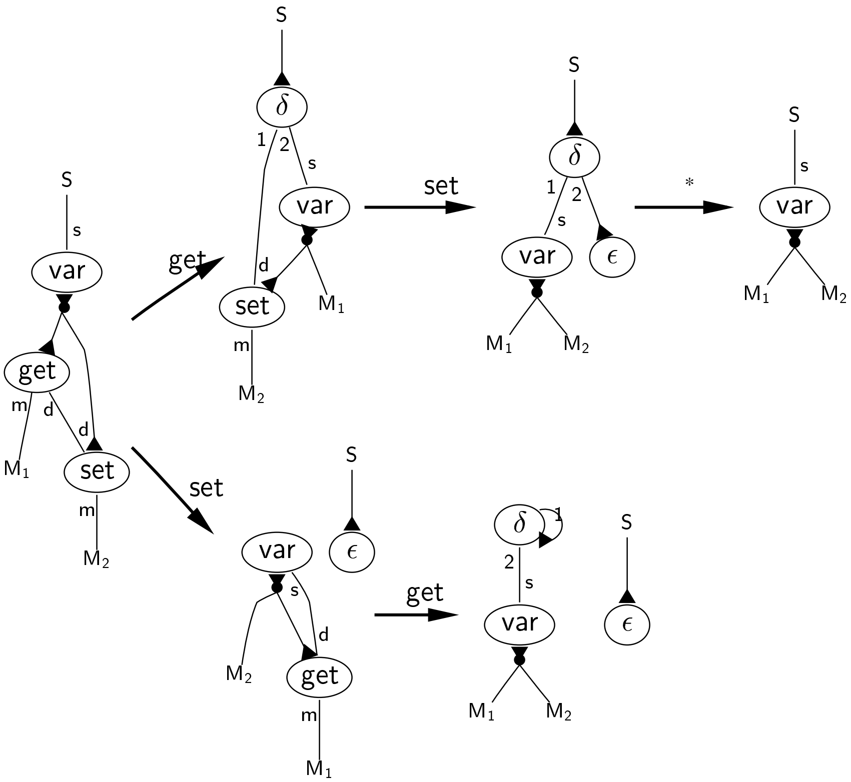
3.3.1 INMP Example: Variable (Reference)



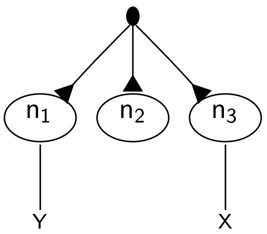
A variable is an "object" that handles get and set requests



Outcome depends on the order of interaction (non-determinism)

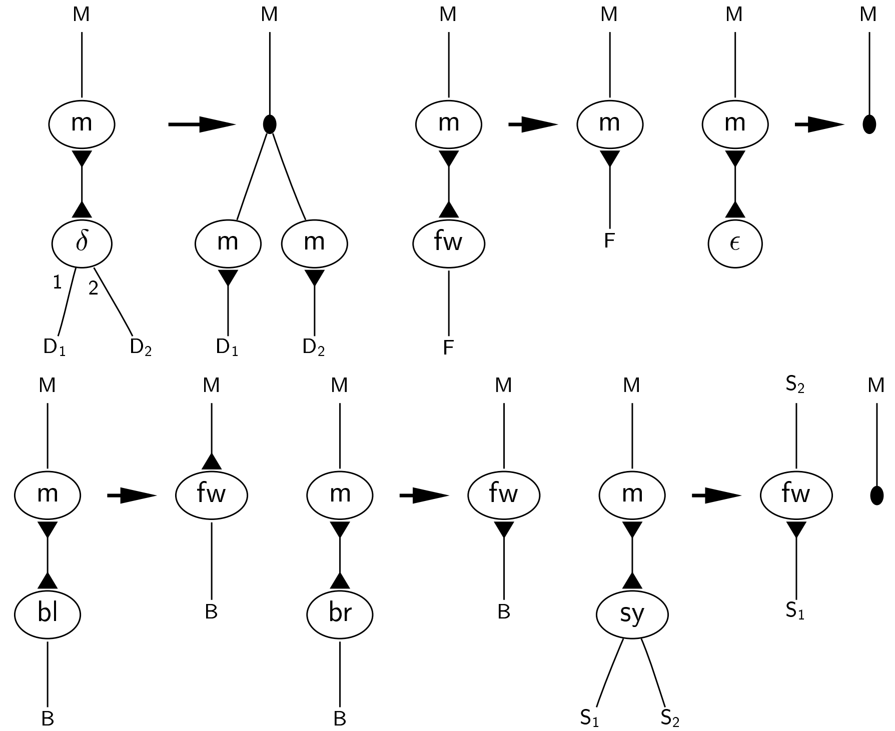


3.4 IN with Multiple Connections (INMC)



Allow hyper-edges (edges connecting more than two ports). We denote such with a connector point (bold dot)

3.4.1 INMC Example: Process Graphs

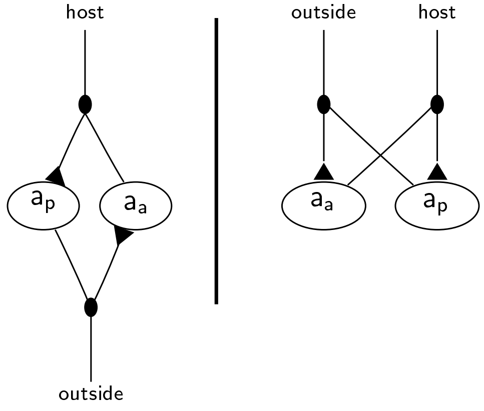


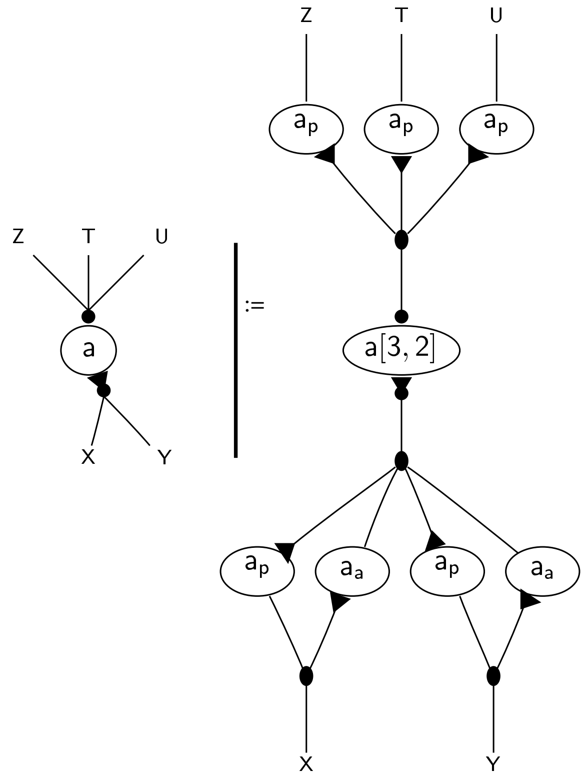
4 Inter-representation of Non-Deterministic Models

Which models can represent which others, and at what price?

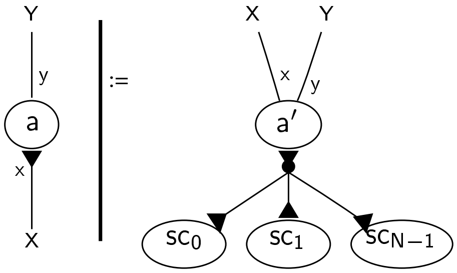
- Complexity of translation
- Complexity of reduction
- Atomicity properties
- Commitment properties

4.1 INMP as INMC: Port Diamonds

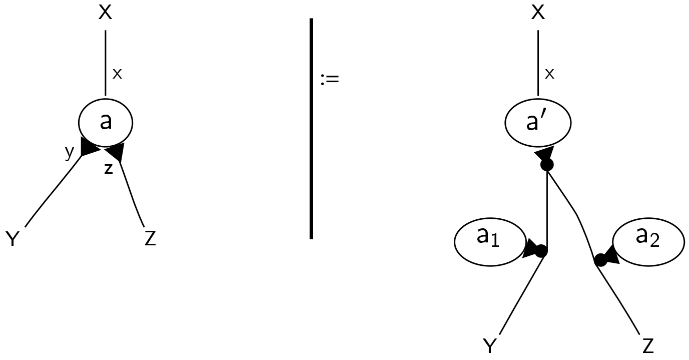




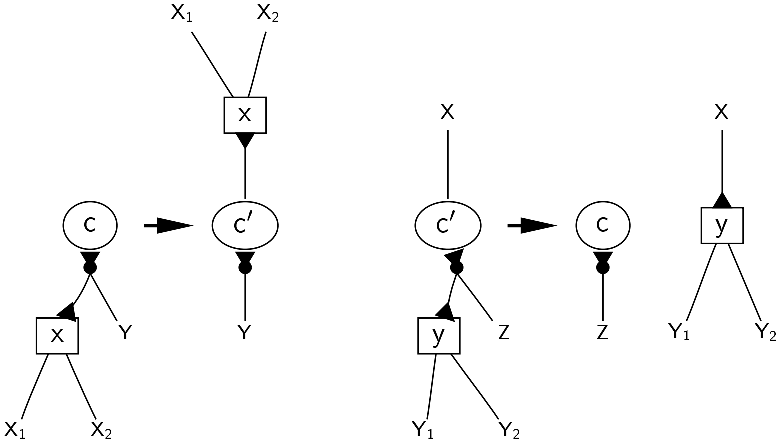
4.2 INMR as INMP: Self-Commitment



4.3 INMPP as INMP: Marker Nodes



4.4 INMC as INMP: Explicit Connectors



5 Representing the π-calculus in MIN=INMP+INMPP

Concurrency = Interaction + Non-determinism

5.1 The π-calculus

Zero $\boxed{0}$ is the empty process.

Parallel Composition $\boxed{P, Q}$.

Output Prefix $\boxed{c!v.P}$ sends value v along channel c, then does P.

Input Prefix $\boxed{c?x.P}$ receives value v from channel c, then does $P[v/x]$.

Hiding/Restriction $\boxed{(c)P}$ can't interact on channel c.

5.1.1 π-calculus Example: Email and Phone

$$t?x.x!m, t!e, e?y$$

A person talks on the telephone t to another person and receives an email address e. Then s/he sends a message m to that address e, which is received by a third person

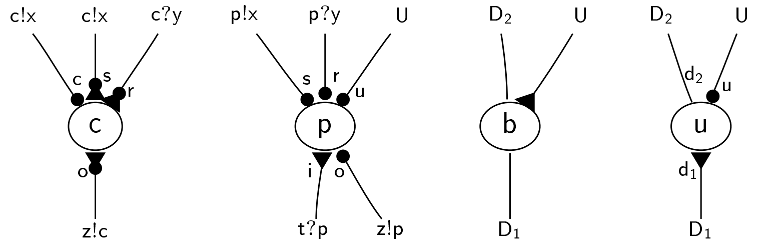
5.1.2 Reduction Rules

Comm $a?x.P, a!c.Q \to P[c/x], Q$: analogous to β-reduction

Par If $P \to Q$ then $P, R \to Q, R$

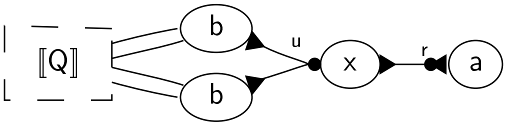
Res If $P \to Q$ then $(x)P \to (x)Q$: restriction does not restrict the internal process.

5.2 MIN$_\pi$ Nodes

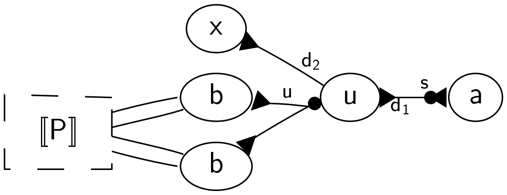


Channel, placeholder, blocker, unblocker

5.3 The Translation

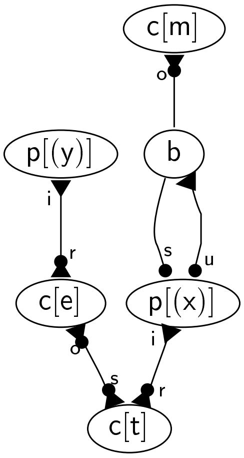


$[\![a?x.Q]\!]$



$[\![a!x.P]\!]$

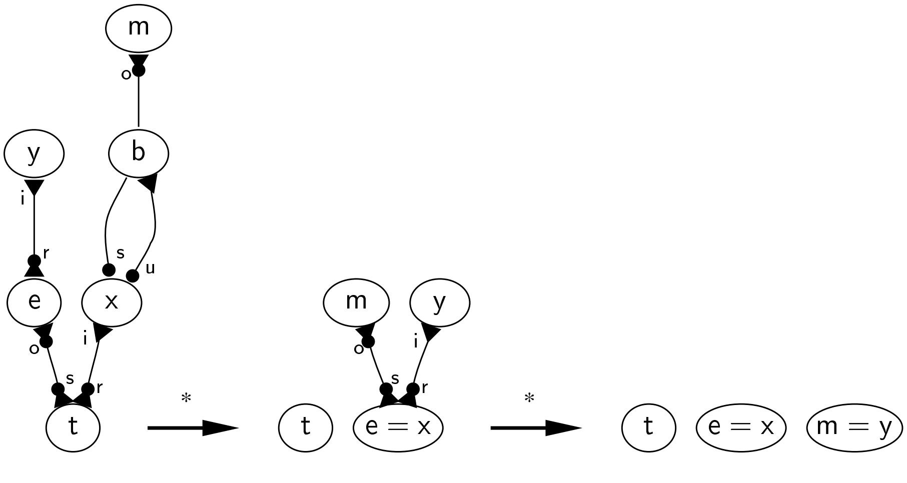
5.3.1 Example: Translation of Email-Phone



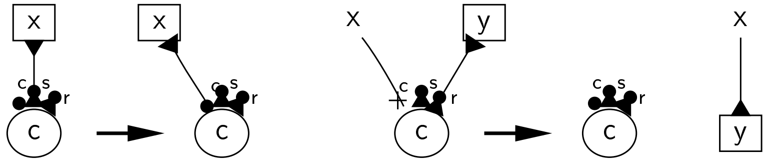
$t?x.x!m, t!e, e?y$

5.4 MIN_π Reduction Rules

Example of reduction: Email-Phone

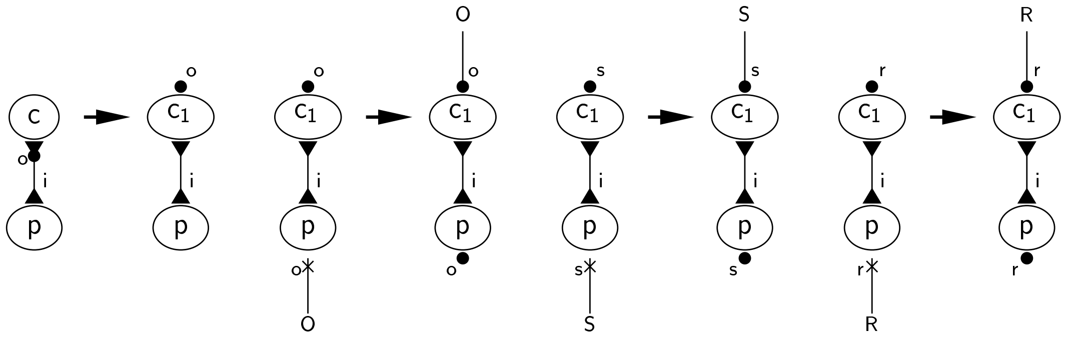


5.4.1 Send/Receive

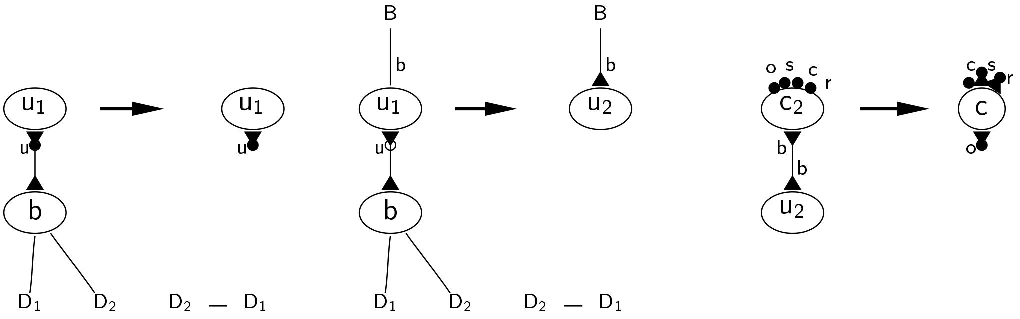


5.4.2 Input/Output

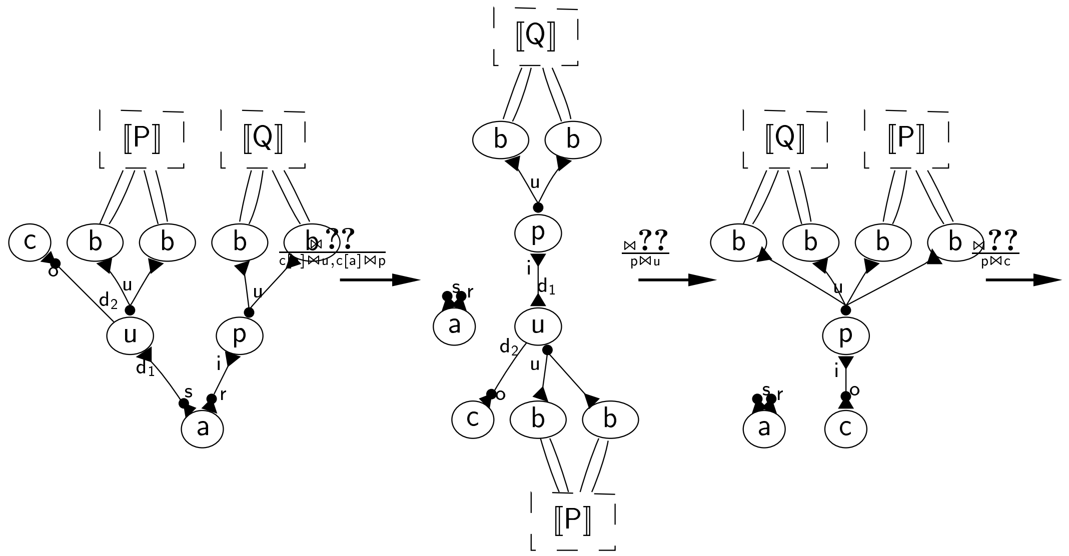
(Link Migration)

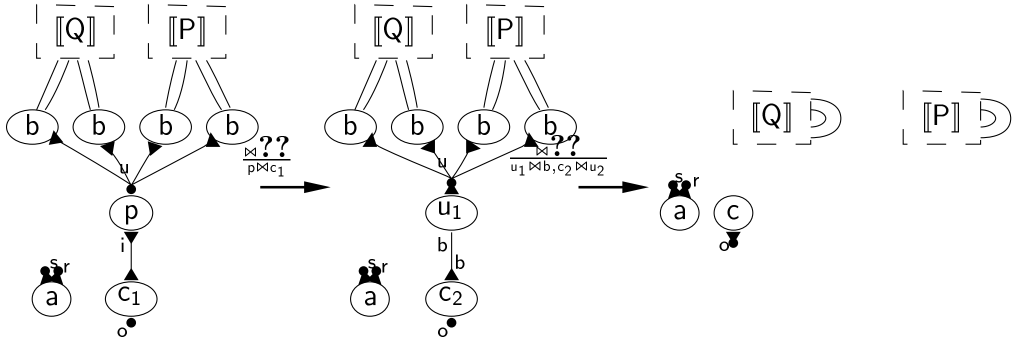


5.4.3 Blocking and Unblocking



5.4.4 Example of Reduction





5.5 Completeness and Soundness

Completeness:

$$\begin{tikzcd}
\forall P \arrow[r] \arrow[d, mapsto] & \forall P' \arrow[d, mapsto] \\
\forall \llbracket P \rrbracket \arrow[r, Rightarrow] & \exists \llbracket P' \rrbracket
\end{tikzcd}$$

Soundness: more problematic

$$\begin{tikzcd}
\forall P \arrow[rr, Rightarrow] \arrow[d, mapsto] & & \exists P' \arrow[d, mapsto] \\
\forall \llbracket P \rrbracket \arrow[r, Rightarrow] & \forall N \arrow[r, Rightarrow, "\prec"] & \exists \llbracket P' \rrbracket
\end{tikzcd}$$