

MagNets

Implementing Mobile Nets as Mobile Agents

Nadia Busi and Luca Padovani

{busi,lpadovan}@cs.unibo.it

Department of Computer Science

University of Bologna

Introduction

Goal: definition of formal semantics of Web service infrastructures

π -calculus	Petri Nets
+ dynamic configurations	– static configurations
– synchronization	+ synchronization
– conflicts	– conflicts

Introduction

Goal: definition of formal semantics of Web service infrastructures

π -calculus	Petri Nets
+ dynamic configurations	– static configurations
– synchronization	+ synchronization
– conflicts	– conflicts

MAGNETs

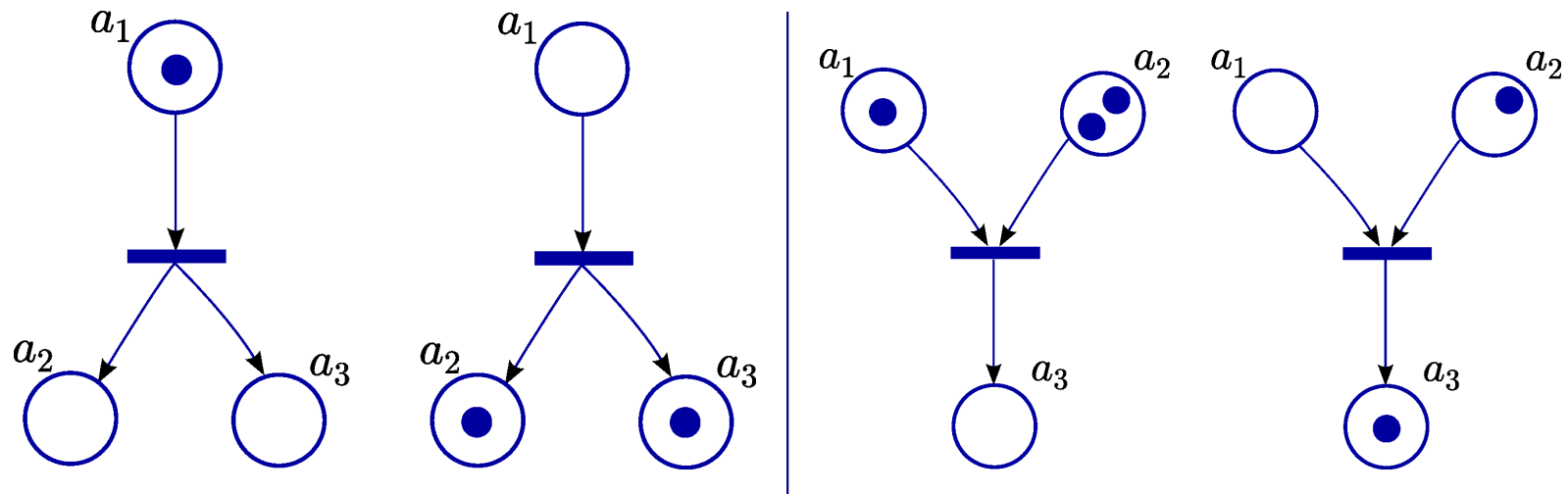
- try to combine the best-of-both worlds
- allow one to model highly dynamic scenarios
- are implementable

Outline

- ⊗ tutorial
- ⊗ syntax and semantics of MAGNETs
- ⊗ examples
- ⊗ implementation
- ⊗ concluding remarks

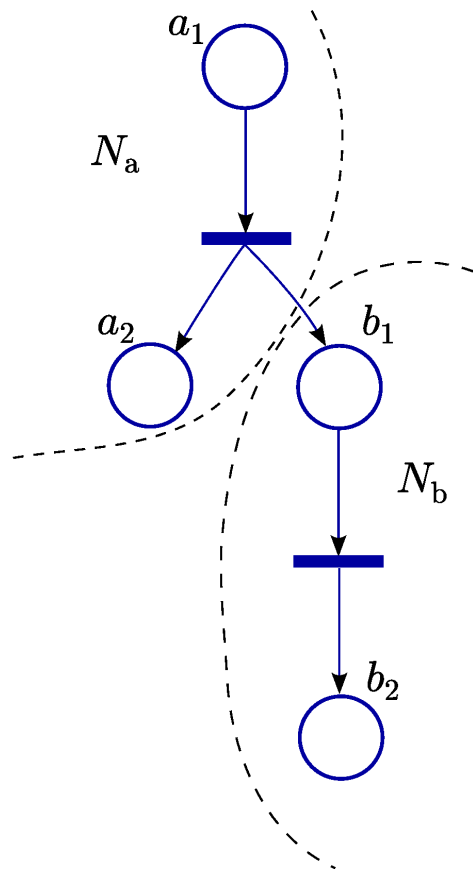
Tutorial: basic firing rules

Tokens, places, transitions...



Tokens are **colored**: the color is a possibly empty tuple of place names

Tutorial: local versus remote communication

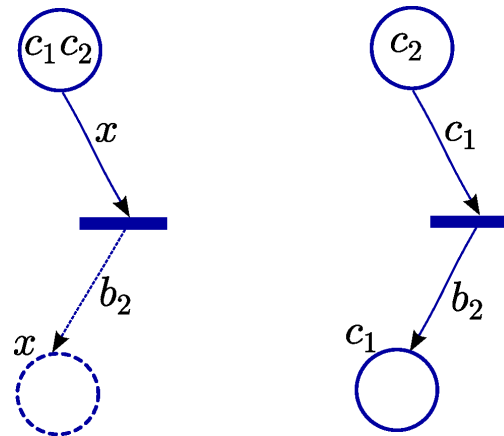


A **configuration** is made of different, possibly distributed net systems running concurrently (N_a and N_b)

Communication can be **local**
 ($a_1 \rightarrow a_2$ and $b_1 \rightarrow b_2$) or **remote**
 ($a_1 \rightarrow b_1$)

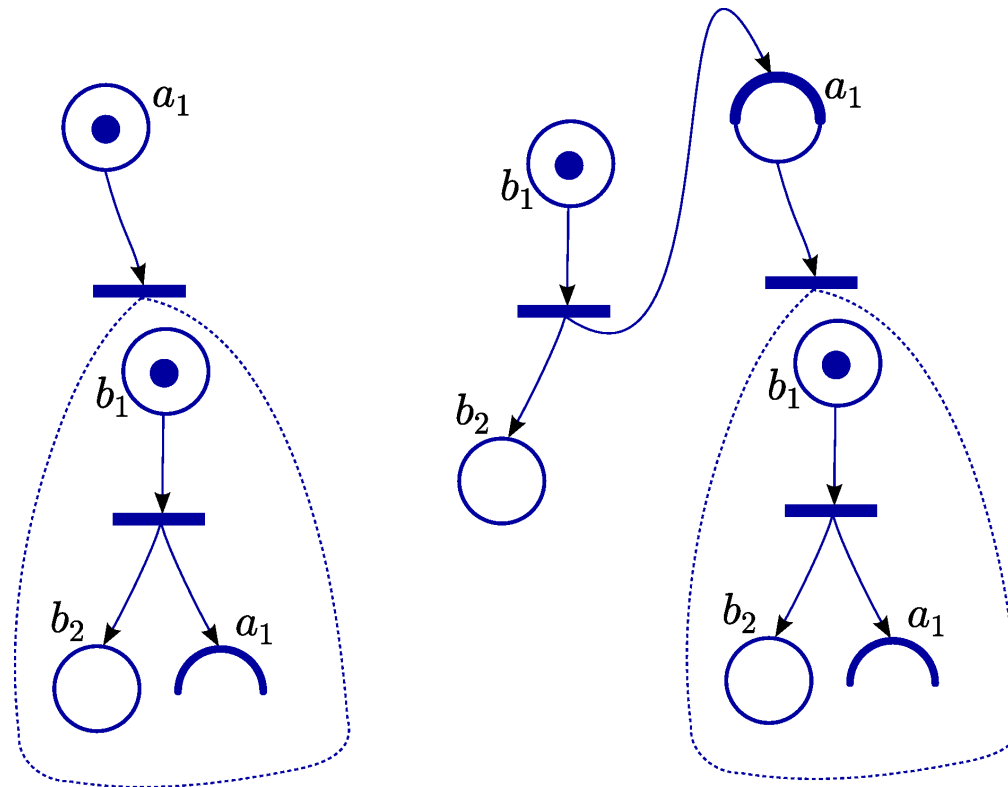
Tutorial: mobility

The color of the tokens consumed by a transition can be used to determine the place where the transition puts the produced tokens:



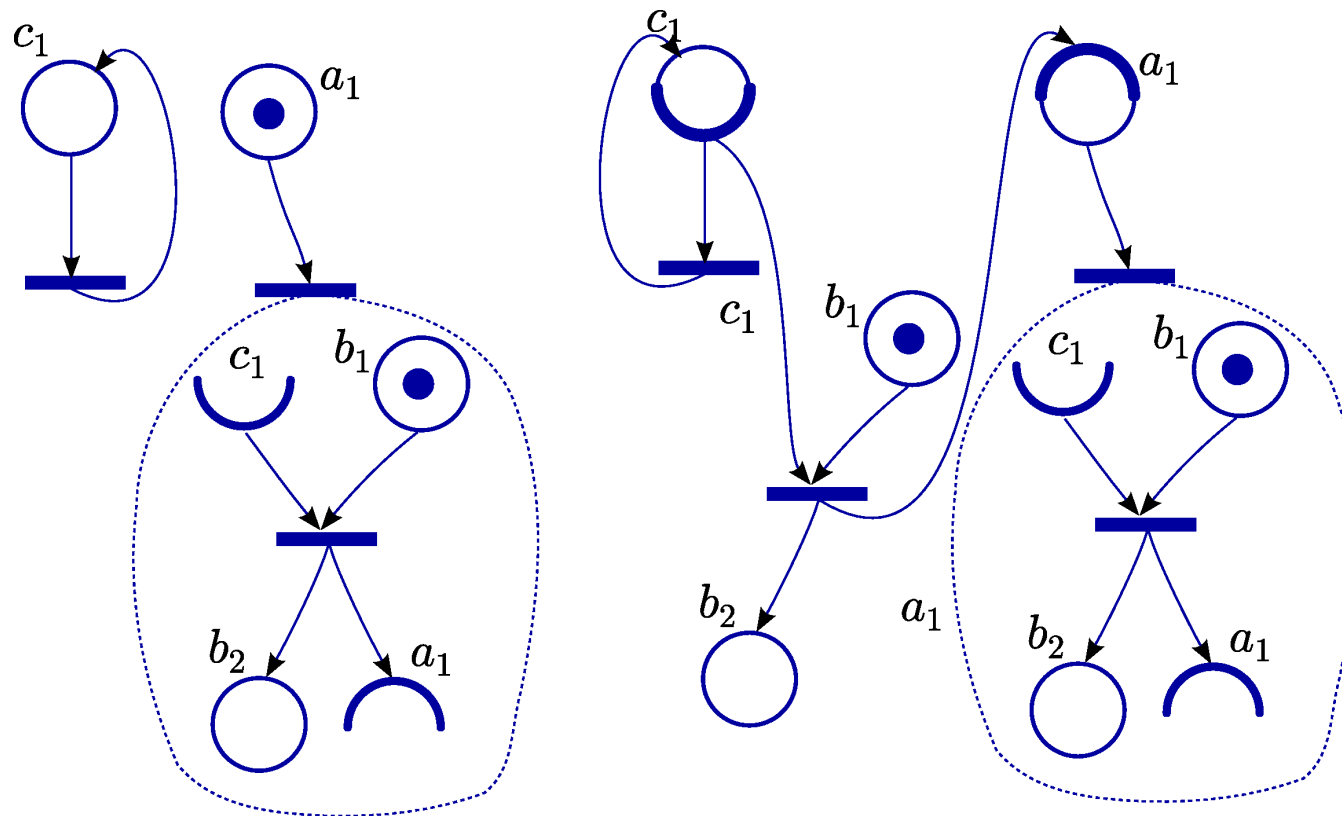
Tutorial: dynamic nets I

New nets can be **created** as the result of firing a transition:



Tutorial: dynamic nets II

Existing nets can be **extended** as the result of firing a transition:



Syntax of MAGNETS

C	$::=$	$[S, T, m] \mid N \mid C \oplus C$	system configuration
N	$::=$	\emptyset	empty net
		$\mid a\langle\bar{b}\rangle$	token
		$\mid t$	transition
		$\mid \{S, T, m\}$	pre-net
		$\mid N \oplus N$	composition of nets
t	$::=$	$p \mapsto N$	transition
m	$::=$	$\emptyset \mid a\langle\bar{b}\rangle \mid m \oplus m$	marking
p	$::=$	$a\langle\bar{x}\rangle \mid p \oplus p$	pattern

System well-formedness

A net system $[S, T, m]$ is **well-formed** if $places(m) \subseteq S$ and $pre(T) \subseteq S$.

A system configuration C is **well-formed** if $C \equiv N \oplus \bigoplus_{i=1}^k [S_i, T_i, m_i]$, and the following conditions are satisfied:

- $\forall i : 1 \leq i \leq k \Rightarrow [S_i, T_i, m_i]$ is well-formed, and
- $S_i \cap S_j = \emptyset$ for $1 \leq i, j \leq k, i \neq j$, and
- $fn(N) \subseteq \bigoplus_{i=1}^k S_i$

Pattern linearity

Patterns are **linear**: the same name is not bound more than once.

- $a(x_1, \dots, x_n)$ is linear if $\forall 1 \leq i, j \leq n: x_i = x_j \Rightarrow i = j$
- $p \oplus p'$ is linear if p and p' are linear and $bn(p) \cap bn(p') = \emptyset$

Consequence: pattern matching is determined by the presence of tokens into places, not by their colors.

Operational semantics: firing

A firing transition spawns a new {token, transition, pre-net}:

$$\frac{p \rightarrow N \in T}{[S, T, m \oplus p[\rho]] \rightarrow [S, T, m] \oplus N\rho}$$

where

$$a(x_1 \dots x_n)[\rho] = a\langle \rho(x_1) \dots \rho(x_n) \rangle$$

$$(p \oplus p')[\rho] = p[\rho] \oplus p'[\rho]$$

Note: the net $N\rho$ resulting from the firing rule is placed **outside** the net-system where firing occurs for technical simplicity.

Operational semantics: migration

A token **migrates** to the net-system where the token's place reside:

$$\frac{s \in S}{[S, T, m] \oplus s\langle \bar{x} \rangle \rightarrow [S, T, m \oplus s\langle \bar{x} \rangle]}$$

A transition migrates to the net-system where the transition's places reside:

$$\frac{places(p) \subseteq S}{[S, T, m] \oplus (p \mapsto N) \rightarrow [S, T \cup \{p \mapsto N\}, m]}$$

Locality: the pre-set of a transition must contain places from the same net-system

Operational semantics: net creation and extension

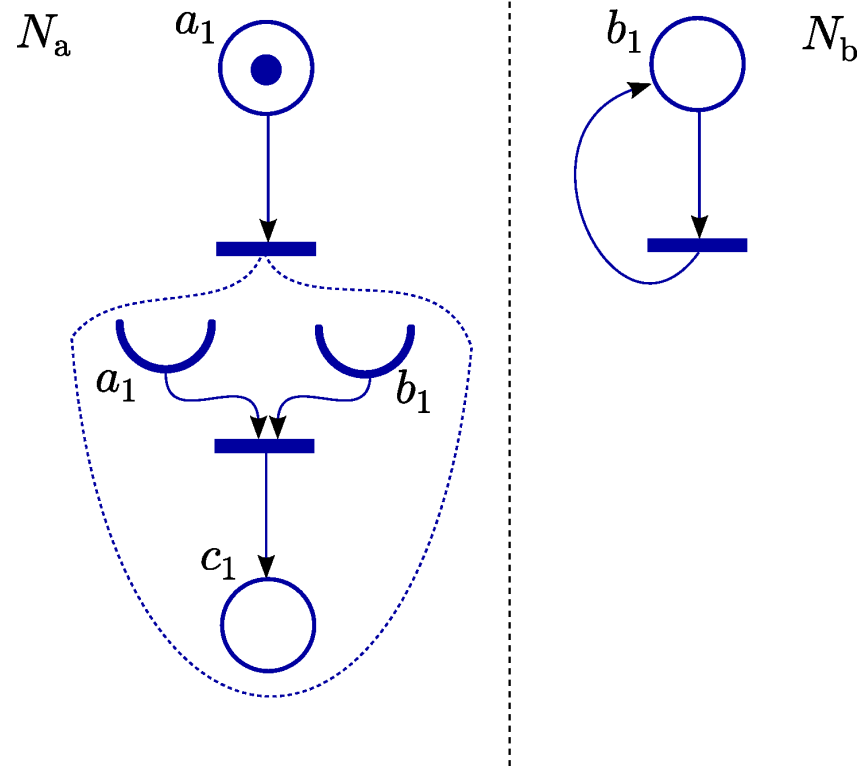
A **closed pre-net** evolves into a fully functional net-system:

$$\frac{\begin{array}{c} S \text{ fresh} \\ \forall t \in T, \forall s \in S : s \in \text{pre}(t) \Rightarrow \text{pre}(t) \subseteq S \end{array}}{\{S, T, m\} \rightarrow [S, \emptyset, \emptyset] \oplus T \oplus m} \quad \text{closeness}$$

An **open pre-net** extends an existing net-system:

$$\frac{\begin{array}{c} S \text{ fresh} \\ \text{pre}(T) \cap S' \neq \emptyset \\ \forall t \in T, \forall s \in S : s \in \text{pre}(t) \Rightarrow \text{pre}(t) \subseteq S \cup S' \end{array}}{\{S, T, m\} \oplus [S', T', m'] \rightarrow [S \cup S', T', m'] \oplus T \oplus m} \quad \begin{array}{l} \text{openness} \\ \text{locality} \end{array}$$

Net extension and locality



Example: applet

A client wants to execute an applet which is found on a remote system:

$$Client = [\{runHere\}, \dots, runHere] \oplus appletX \langle runHere \rangle$$

$$Server = [\{appletX\}, \{appletX(run) \mapsto \{\emptyset, \{run \mapsto A\}, \emptyset\}\}, \emptyset]$$

Example: applet

A client wants to execute an applet which is found on a remote system:

$$\begin{aligned}
 \textit{Client} &= [\{\textit{runHere}\}, \dots, \textit{runHere}] \oplus \textit{appletX} \langle \textit{runHere} \rangle \\
 \textit{Server} &= [\{\textit{appletX}\}, \{\textit{appletX}(\textit{run}) \mapsto \{\emptyset, \{\textit{run} \mapsto A\}, \emptyset\}\}, \emptyset]
 \end{aligned}$$

The client's request reaches the server:

$$\begin{aligned}
 \textit{Client} &= [\{\textit{runHere}\}, \dots, \textit{runHere}] \\
 \textit{Server} &= [\{\textit{appletX}\}, \{\textit{appletX}(\textit{run}) \mapsto \{\emptyset, \{\textit{run} \mapsto A\}, \emptyset\}\}, \\
 &\quad \textit{appletX} \langle \textit{runHere} \rangle]
 \end{aligned}$$

Example: applet (cont.)

An applet is spawned by the server:

$$Client = [\{runHere\}, \dots, runHere]$$

$$Server = [\{appletX\}, \{appletX(run) \mapsto \{\emptyset, \{run \mapsto A\}, \emptyset\}\}, \emptyset] \\ \oplus \{\emptyset, \{runHere \mapsto A\}, \emptyset\}$$

Example: applet (cont.)

An applet is spawned by the server:

$$Client = [\{runHere\}, \dots, runHere]$$

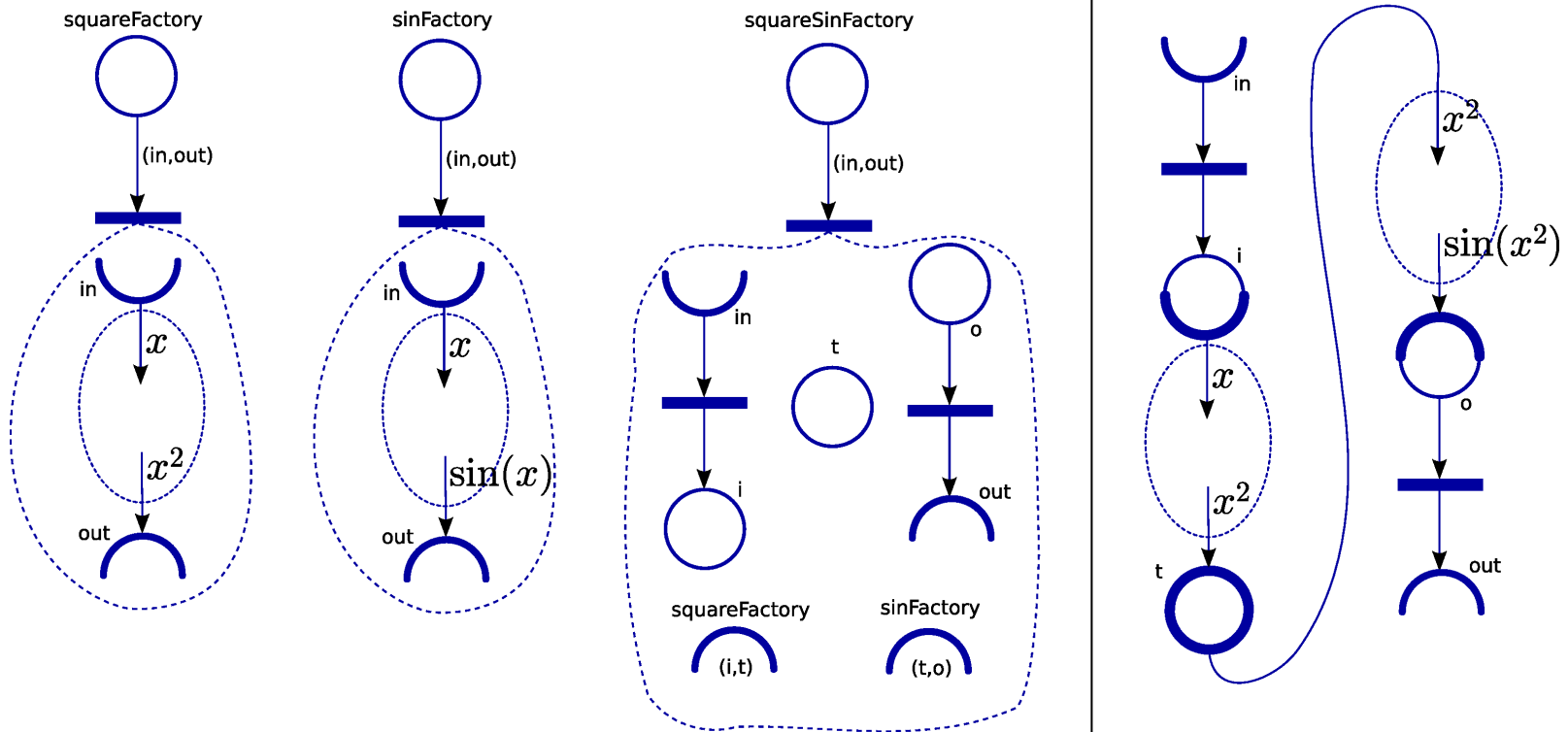
$$Server = [\{appletX\}, \{appletX(run) \mapsto \{\emptyset, \{run \mapsto A\}, \emptyset\}\}, \emptyset] \\ \oplus \{\emptyset, \{runHere \mapsto A\}, \emptyset\}$$

The applet migrates into the client:

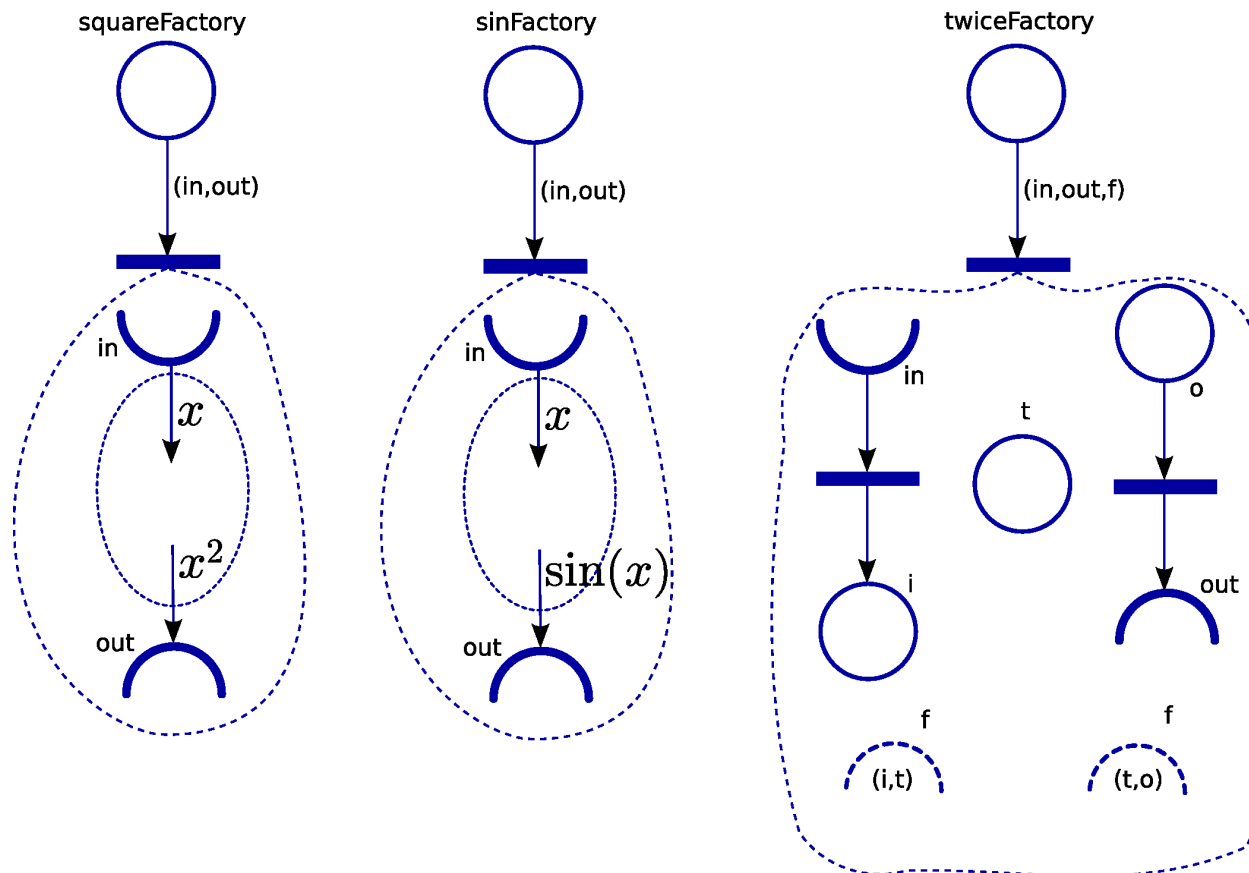
$$Client = [\{runHere\}, \dots, runHere \mapsto A \dots, runHere]$$

$$Server = [\{appletX\}, \{appletX(run) \mapsto \{\emptyset, \{run \mapsto A\}, \emptyset\}\}, \emptyset]$$

Example: Web service generation



Example: Web service generation



Implementation: nets as agents

Model	Implementation	made of...
token	message	destination + tuple of names (color)
place	port	message queue
transition	rule	multiset of ports + blocked thread
net system	agent	threads + data
–	thread	sequential unit of computation (code + data + stack + pc)

Implementation: the virtual machine

Instruction	Stack before	Stack after
port	S	$pid :: S$
thread(n) c	$x_1 :: \dots :: x_n :: S$	$tid :: S$
rule(n)	$tid :: pid_1 :: \dots :: pid_n :: S$	S
location	S	$lid :: S$
send(n)	$pid :: x_1 :: \dots :: x_n :: S$	S
spawn	$lid :: tid :: S$	S
load(n)	S	$x :: S$
store(n)	$x :: S$	S
whereis	$pid :: S$	$lid :: S$
move	$lid :: S$	S

Compiling MAGNETs: tokens and transitions

Token \Rightarrow asynchronous communication:

$$\llbracket x \langle y_1, \dots, y_n \rangle \rrbracket = [\text{load}(y_n); \dots; \text{load}(y_1); \\ \text{load}(x); \text{send}(n)]$$

Synchronous communication can be achieved by means of **continuations**

Transition \Rightarrow runtime rule for thread spawning:

$$\llbracket x_1(\overline{y_1}), \dots, x_n(\overline{y_n}) \mapsto N \rrbracket = [\text{load}(z_k); \dots; \text{load}(z_1); \\ \text{thread}(k) \llbracket N \rrbracket; \\ \text{load}(x_n); \dots; \text{load}(x_1); \text{rule}(n)]$$

Compiling MAGNETs: closed pre-nets

A closed pre-net $\{\{x_1, \dots, x_n\}, \{t_1, \dots, t_k\}, m\}$ evolves into a standalone agent (closeness $\Rightarrow \text{pre}(\{t_1, \dots, t_k\}) \subseteq \{x_1, \dots, x_n\}$):

$$\begin{aligned} & \llbracket \{\{x_1, \dots, x_n\}, \{t_1, \dots, t_k\}, m\} \rrbracket = \\ & \quad [\text{load}(z_l); \dots; \text{load}(z_1); \\ & \quad \text{thread}(l) \\ & \quad \quad [\text{port}; \text{store}(x_1); \dots; \text{port}; \text{store}(x_n)] \\ & \quad \quad @\llbracket t_1 \rrbracket @ \dots @\llbracket t_k \rrbracket @\llbracket m \rrbracket; \\ & \quad \text{location}; \text{spawn}] \end{aligned}$$

Compiling MAGNETs: open pre-nets

An open pre-net $\{\{x_1, \dots, x_n\}, \{t_1, \dots, t_k\}, m\}$ extends an existing net (openness $\Rightarrow \exists z \in pre(\{t_1, \dots, t_k\}) \setminus \{x_1, \dots, x_n\}$):

$$\begin{aligned} & \llbracket \{\{x_1, \dots, x_n\}, \{t_1, \dots, t_k\}, m\} \rrbracket = \\ & \quad [\text{load}(y_l); \dots; \text{load}(y_1); \quad (* \text{ closure } *) \\ & \quad \text{thread}(l) \\ & \quad \quad [\text{load}(z); \text{whereis}; \text{move}; \quad (* \text{ migration } *) \\ & \quad \quad \text{load}(y_l); \dots; \text{load}(y_1); \\ & \quad \quad \text{thread}(l) \\ & \quad \quad \quad [\text{port}; \text{store}(x_1); \dots; \text{port}; \text{store}(x_n)] \\ & \quad \quad \quad @\llbracket t_1 \rrbracket @ \dots @\llbracket t_k \rrbracket @\llbracket m \rrbracket; \\ & \quad \quad \quad \text{load}(z); \text{whereis}; \text{spawn}] \\ & \quad \text{location}; \text{spawn}] \end{aligned}$$

Code improvements

- reuse of continuation ports in the same thread
- parallel, asynchronous (non-blocking) operations are performed by the same thread
- static analysis of patterns (synchronous message communication as function call, patterns with private places as FSA)

Concluding remarks

- MAGNETs provide a model with nice properties from both theoretical and practical point of views
- strictly related to the distributed join-calculus with locations
- mobility is not explicit in the model syntax
- the virtual machine permits a more general form of mobility than is actually needed

To do:

- investigate Web service orchestration and composition
- clean up the virtual machine and provide a correctness proof
- type-system and capabilities