Foundations of Global Computing

A Personal Perspective

Vladimiro Sassone

University of Sussex

≪ ⋘ ⋙ ≫

Foundations of Global Computing

Resource Control

Programming Languages

Semantic Theories

Models for Concurrency

Global Ubiquitous Computing:

computation over a global network of mobile, bounded resources shared among mobile entities which move between highly dynamic, largely unknown, untrusted networks.

Difficulties:

Extreme dynamic reconfigurability; lack of coordination and trust; limited capabilities; partial knowledge . . .

Issues:

Protection and management of resources; privacy and confidentiality of data; . . .

Resource Control

Programming Languages

Semantic Theories

Models for Concurrency

Petri Nets Based Models and Calculi

A distributed timed-arc Petri net (DTAPN) is a Petri net together with

- a interval time constraint on transitions, either discrete or continuous;
- a clock synchronisation equivalence Σ on places.

Tokens age, and transitions are enabled accordingly. Time elapses at the same speed on p and p' if $p \, \Sigma \, p'$.

Globally Asynchronous, Locally Synchronous

Global Time: $\Sigma = P \times P$ Local Time: $\Sigma = \Delta_P$

A Separation Result: Reachability for safe LT nets is decidable, but undecidable for safe GT nets.

ICATPN 2001, FST&TCS 2001

Foundations of Global Computing

Resource Control

Programming Languages

Semantic Theories

Models for Concurrency

- A categorical machinery which allows the derivation of LTSs from reduction systems.
- Bisimulation on such LTSs is a congruence, provided a general condition is met.

Coinduction Principle Desiderata:

- Operational Corresp.: $p \searrow q$ iff $p \xrightarrow{\tau} q$
- Correctness: $p \approx q$ implies $p \cong q$
- Completeness: $p \cong q$ implies $p \approx q$

The intuition:

$$a \xrightarrow{\mathscr{C}} b \text{ iff } \mathscr{C}[a] \searrow b$$

Eg:

$$a \xrightarrow{-|\bar{a}} \mathbf{0} \qquad M \xrightarrow{(\lambda x.-)N} M\{N/x\} \qquad \mathbf{K}M \xrightarrow{-N} M$$

Foundations of Global Computing

Resource Control

Programming Languages

Semantic Theories

Models for Concurrency

- A categorical machinery which allows the derivation of LTSs from reduction systems.
- Bisimulation on such LTSs is a congruence, provided a general condition is met.

Coinduction Principle Desiderata:

- Operational Corresp.: $p \searrow q$ iff $p \xrightarrow{\tau} q$
- Correctness: $p \approx q$ implies $p \cong q$
- Completeness: $p \cong q$ implies $p \approx q$

The intuition:

$$a \xrightarrow{\mathscr{C}} b \text{ iff } \mathscr{C}[a] \searrow b$$

But: Must choose labels carefully not to mess up the bisimulation

Choose only 'minimal' redex-enabling contexts: GRPOs.

Relative pushouts in groupoidal categories:

EXPRESS 2002, FOSSACS 2003, NJC, TCS

≪ ⋘ ⋙ ≫

Foundations of Global Computing

Resource Control

Programming Languages

Semantic Theories

Models for Concurrency

Jeeg: concurrent OO with history-sensitive access control

- Java (no synchronized(), wait(), notify(), notifyAll()) for business code;
- Linear Time Temporal Logic for synchronisation code (method guards).

```
public class MyClass {
    sync {
      m : φ;
      ....
    }
    ...// Standard Java class def
  }
```

where m is a method identifier and ϕ, the guard, is an LTL formula. When m is invoked, the thread is kept on hold unless ϕ. When the condition is true, all waiting threads are awaken. m is implicitly synchronised.

JavaGRANDE 2002, WOODS 2003, Conc & Comp, OOPS 2004

Foundations of Global Computing

Resource Control

Programming Languages

Semantic Theories

Models for Concurrency

Resources: Models, Types, Logics, Languages

- Access Control (Concur 2002, ESOP 2004)
- Access Authorisation (FST&TCS 2002, Info&Co)
- Secrecy for Mobile Agents (ICALP 2003)
- Trust Management (SEFM 2003)
- Bounds Control (ASIAN 2003)

Mobile Ambients

Both administrative domains and computational environments

- Subjective movements

$$n[\ \mathtt{in}\ m.P \mid Q\ ] \mid m[\ R\ ] \longrightarrow m[\ n[\ P \mid Q\ ] \mid R\ ]$$
$$m[\ n[\ \mathtt{out}\ m.P \mid Q\ ] \mid R\ ] \longrightarrow n[\ P \mid Q\ ] \mid m[\ R\ ]$$

- Boundary dissolver

$$\mathtt{open}\ \ n.P \mid n[\ Q\ ] \longrightarrow P \mid Q.$$

- Process interaction

$$n[\ \langle M \rangle.P \mid (x).Q\ ] \longrightarrow n[\ P \mid Q\{x := M\}\ ],$$

Group Types for Mobility

Aim: *Resource Access Control*

- Detect and prevent unwanted access to resources.
- Focus on static approaches based on enforcing type disciplines.

Group Types for Mobility

Aim: *Resource Access Control*

- Detect and prevent unwanted access to resources.
- Focus on static approaches based on enforcing type disciplines.

Groups: Sets of processes with common access rights.

Constraints like $k : \mathsf{CanEnter}(n)$ are modelled as:

n belongs to group G

k may cross the border of ambients of group G.

For instance, the system:

$$k[\,\texttt{in}\, n \mid l[\,\texttt{out}\, k\,]\,] \mid n[\,\_\,]$$

is well-typed under assumptions of the form:

$$k : \mathsf{amb}[\mathbf{K}, \mathsf{cross}(\mathbf{N})]$$

$$l : \mathsf{amb}[\mathbf{L}, \mathsf{cross}(\mathbf{K})] \qquad n : \mathsf{amb}[\mathbf{N}, \ldots]$$

Indirect Border Crossing

Trojan Horses: The system

$$\text{Odysseus}[\,\mathbf{in}\ \text{Horse}.\mathbf{out}\ \text{Horse}.\mathit{Destroy}\,] \mid \text{Horse}[\,\mathbf{in}\ \text{Troy}\,] \mid \text{Troy}[\,\mathit{Trojans}\,]$$

is well-typed under assumptions:

$$\text{Odysseus} : \text{amb}[\mathtt{Achaean}, \text{cross}(\mathtt{Toy})]$$

$$\text{Horse} : \text{amb}[\mathtt{Toy}, \text{cross}(\mathtt{City})]$$

$$\text{Troy} : \text{amb}[\mathtt{City}, \_]$$

Indirect Border Crossing

Trojan Horses: The system

$$\text{Odysseus}[\,\mathbf{in}\,\text{Horse}.\mathbf{out}\,\text{Horse}.\mathit{Destroy}\,] \mid \text{Horse}[\,\mathbf{in}\,\text{Troy}\,] \mid \text{Troy}[\,\mathit{Trojans}\,]$$

is well-typed under assumptions:

$$\text{Odysseus} : \text{amb}[\mathbf{Achaean}, \text{cross}(\mathbf{Toy})]$$

$$\text{Horse} : \text{amb}[\mathbf{Toy}, \text{cross}(\mathbf{City})]$$

$$\text{Troy} : \text{amb}[\mathbf{City}, \_]$$

However, the system may evolve to

$$\text{Troy}[\,\mathit{Trojans} \mid \text{Horse}[\,\_\,] \mid \text{Odysseus}[\,\mathit{Destroy}\,]\,]$$

where Odysseus got inside Troy's Walls taking by surprise the *Trojans*.

Types

Groups: $\mathsf{G}, \mathsf{H}, \ldots$

Sets of groups: $\mathscr{P}, \mathscr{S}, \ldots$ $\quad \mathscr{U}$ The universal set of groups

Ambients types:

$A \quad ::= \quad \mathrm{amb}[\mathsf{G}, M]$ amb of group G, with mobility type M

Process types:

$\Pi \quad ::= \quad \mathrm{proc}[\mathsf{G}, M]$ process that can be enclosed in an ambient of group G, may drive to ambients whose groups are in M

Capability types:

$K \quad ::= \quad \mathrm{cap}[\mathsf{G}, M]$ capability that can appear in an ambient of group G, may drive it to ambients whose groups are in M

Mobility types:

$M \quad ::= \quad \mathrm{mob}[\mathscr{P}]$ mobility specs: where processes are allowed to reside

Access Control Properties

Mobility properties:

- If $\Gamma \vdash n[\,\mathtt{in}\, m.P \mid Q\,] \mid m[\,R\,] : \Pi$, then

$$\Gamma \vdash m : \mathsf{amb}[\mathtt{M}, \_] \quad \text{and} \quad \Gamma \vdash n : \mathsf{amb}[\_, \mathsf{mob}[\mathscr{P}]]$$

  with $\mathtt{M} \in \mathscr{P}$.

- If $\Gamma \vdash m[\,n[\,\mathtt{out}\, m.P \mid Q\,] \mid R\,] : \Pi$, then

$$\Gamma \vdash m : \mathsf{amb}[\mathtt{M}, \mathsf{mob}[\mathscr{P}_m]] \text{ and } \Gamma \vdash n : \mathsf{amb}[\mathtt{N}, \mathsf{mob}[\mathscr{P}_n]]$$

  with $\mathtt{M} \in \mathscr{P}_n$ and $\mathscr{P}_m \subseteq \mathscr{P}_n$.

Detecting Odysseus' intentions

Now, in order to assign a type to

$$\mathsf{Odysseus}[\,\mathbf{in}\,\mathsf{Horse}.\mathbf{out}\,\mathsf{Horse}.\mathit{Destroy}\,] \mid \mathsf{Horse}[\,\mathbf{in}\,\mathsf{Troy}\,] \mid \mathsf{Troy}[\,\mathit{Trojans}\,]$$

we need assumptions of the form:

$$\mathsf{Odysseus} : \mathsf{amb}[\mathtt{Achaean}, \mathsf{mob}[\{\mathtt{Ground}, \mathtt{Toy}, \mathtt{City}\}]]$$

$$\mathsf{Horse} : \mathsf{amb}[\mathtt{Toy}, \mathsf{mob}[\{\mathtt{Ground}, \mathtt{City}\}]]$$

$$\mathsf{Troy} : \mathsf{amb}[\mathtt{City}, \_]$$

representing that Odysseus is an `Achaean` intentioned to move into a `City`!

≪ ⋘ ⋙ ≫

Detecting Odysseus' intentions

Now, in order to assign a type to

$$\text{Odysseus}[\,\mathbf{in}\ \text{Horse}.\mathbf{out}\ \text{Horse}.\mathit{Destroy}\,] \mid \text{Horse}[\,\mathbf{in}\ \text{Troy}\,] \mid \text{Troy}[\,\mathit{Trojans}\,]$$

we need assumptions of the form:

$$\text{Odysseus} : \text{amb}[\texttt{Achaean}, \text{mob}[\{\texttt{Ground}, \texttt{Toy}, \texttt{City}\}]]$$

$$\text{Horse} : \text{amb}[\texttt{Toy}, \text{mob}[\{\texttt{Ground}, \texttt{City}\}]]$$

$$\text{Troy} : \text{amb}[\texttt{City}, \_]$$

representing that Odysseus is an Achaean intentioned to move into a City!

On the other hand, under assumptions of the form

$$\text{Odysseus} : \text{amb}[\texttt{Achaean}, \text{mob}[\{\texttt{Ground}, \texttt{Toy}\}]]$$

the *Trojans* should not fear any attack from Odysseus.

Dependent Types for Access Control

Dependent Mobility Types: ($\mathscr{P}$ and $\mathscr{C}$ are sets of names, not groups.)

$$A \quad ::= \quad \mathsf{amb}[\mathsf{hasFathers}(\mathscr{P}), \mathsf{hasChildren}(\mathscr{C})]$$

Dependent Types for Access Control

Dependent Mobility Types: ($\mathscr{P}$ and $\mathscr{C}$ are sets of names, not groups.)

$$A \quad ::= \quad \mathsf{amb}[\mathsf{hasFathers}(\mathscr{P}), \mathsf{hasChildren}(\mathscr{C})]$$

Dependent types allow personalised services and dynamic access control.

$$\mathsf{HorseServer} \triangleq\ !(x)(\nu\, \mathsf{Horse} : \mathsf{amb}[\_, \mathsf{hasChildren}\{x\}])\mathsf{Horse}[\ \mathbf{out}\ \mathsf{Troy}.\mathbf{in}\ \mathsf{Troy}.\mathbf{0}\ ]$$

≪ ⋘ ⋙ ≫

Dependent Types for Access Control

Dependent Mobility Types: ($\mathscr{P}$ and $\mathscr{C}$ are sets of names, not groups.)

$$A \quad ::= \quad \mathsf{amb}[\mathsf{hasFathers}(\mathscr{P}), \mathsf{hasChildren}(\mathscr{C})]$$

Dependent types allow personalised services and dynamic access control.

$$\mathsf{HorseServer} \triangleq !(x)(\nu\, \mathsf{Horse} : \mathsf{amb}[\_, \mathsf{hasChildren}\{x\}])\mathsf{Horse}[\, \mathbf{out}\, \mathsf{Troy}.\mathbf{in}\, \mathsf{Troy}.\mathbf{0}\,]$$

- Names have several possible types, depending on the actual communications occurred.
- The set of names exchanged over a channel must be tracked. Luckily, The set of possible types for a name has a maximum and a minimum element.

Secrecy in Mobile Ambients

Names ≈ Cryptokeys: Carrying messages inside private ambients preserves message integrity and privacy. Or, does it?

$$(\nu n)(\, a[\; n[\, \texttt{out}\ a.\texttt{in}\ b.\langle M \rangle\,]\;] \mid b[\; \texttt{open}\ n.(x)P\;]\,)$$

≪ ⋘ ⋙ ≫

Secrecy in Mobile Ambients

Names $\approx$ Cryptokeys: Carrying messages inside private ambients preserves message integrity and privacy. Or, does it?

$$(\nu n)(\, a[\, n[\, \mathtt{out}\ a.\mathtt{in}\ b.\langle M\rangle\,]\,] \mid b[\, \mathtt{open}\ n.(x)P\,]\,)$$

It actually offers no guarantees for software agents, as n must be revealed along the move, and servers may peek inside.

How to provide stronger protection?

Secrecy in Mobile Ambients

Names $\approx$ Cryptokeys: Carrying messages inside private ambients preserves message integrity and privacy. Or, does it?

$$(\nu n)(\, a[\; n[\, \mathtt{out}\ a.\mathtt{in}\ b.\langle M \rangle\,]\;] \mid b[\; \mathtt{open}\ n.(x)P\;]\,)$$

It actually offers no guarantees for software agents, as n must be revealed along the move, and servers may peek inside.

How to provide stronger protection?

A new crypto-primitive: subjective access control using co-capabilities + data encryption to preserve secrecy of data while agents move autonomously

$$n[\, \mathsf{seal}\ k.P \mid Q\,] \;\longrightarrow\; n\{\!\!\{\, P \mid Q\, \}\!\!\}_k$$

sealed under k

crypto-key

Effects:

- blocks message exchanges and encrypts their contents;
- the sealed ambient cannot communicate, but it may move.

Sealed Ambients

- The mechanism to resume to a fully operational state is associated to movements and co-capabilities containing keys

$$n\{\!\mid \mathtt{in}\ m.P \mid Q \mid\!\}_k \mid m\{\ \overline{\mathtt{in}}\ \{x\}_k.R \mid R'\ \} \longrightarrow m\{\ n[\ P \mid Q\ ] \mid R\{x := n\} \mid R'\ \}$$

Sealed Ambients

➤ The mechanism to resume to a fully operational state is associated to movements and co-capabilities containing keys

$$n\{\!|\, \mathtt{in}\ m.P \mid Q \,|\!\}_k \mid m\{\, \overline{\mathtt{in}}\ \{x\}_k.R \mid R' \,\} \longrightarrow m\{\, n[\, P \mid Q \,] \mid R\{x := n\} \mid R' \,\}$$

Sealed Ambients

➤ The mechanism to resume to a fully operational state is associated to movements and co-capabilities containing keys

$$n\{\!\!\{ \mathtt{in}\ m.P \mid Q \}\!\!\}_k \mid m\{ \overline{\mathtt{in}}\ \{x\}_k.R \mid R' \} \longrightarrow m\{ n[\, P \mid Q \,] \mid R\{x := n\} \mid R' \}$$

Example:

$$(\nu k)a[\ n[\,\mathsf{seal}\ k.\mathtt{out}\ a.\mathtt{in}\ b.\langle M\rangle^{\hat{}}\,]\ ] \mid b[\ \overline{\mathtt{in}}\ \{x\}_k.(y)^x.P\ ]$$

Sealed Ambients

➤ The mechanism to resume to a fully operational state is associated to movements and co-capabilities containing keys

$$n\{\!|\, \mathtt{in}\; m.P \mid Q \,|\!\}_k \mid m\{\, \overline{\mathtt{in}}\; \{x\}_k.R \mid R' \,\} \longrightarrow m\{\, n[\, P \mid Q \,] \mid R\{x := n\} \mid R' \,\}$$

Example:

$$(\nu k)a[\; n[\,\mathsf{seal}\; k.\mathtt{out}\; a.\mathtt{in}\; b.\langle M\rangle^{\hat{}}\,]\;] \mid b[\, \overline{\mathtt{in}}\; \{x\}_k.(y)^x.P\,]$$

$$\longrightarrow (\nu k)a[\; n\{\!|\, \mathtt{out}\; a.\mathtt{in}\; b.\langle M\rangle^{\hat{}} \,|\!\}_k\;] \mid b[\, \overline{\mathtt{in}}\; \{x\}_k.(y)^x.P\,]$$

Sealed Ambients

➤ The mechanism to resume to a fully operational state is associated to movements and co-capabilities containing keys

$$n\{\!|\, \mathtt{in}\ m.P \mid Q \,|\!\}_k \mid m\{\, \overline{\mathtt{in}}\ \{x\}_k.R \mid R' \,\} \longrightarrow m\{\, n[\, P \mid Q \,] \mid R\{x := n\} \mid R' \,\}$$

Example:

$$(\nu k)a[\ n[\,\mathsf{seal}\ k.\mathtt{out}\ a.\mathtt{in}\ b.\langle M\rangle^{\hat{}}\,]\ ] \mid b[\,\overline{\mathtt{in}}\ \{x\}_k.(y)^x.P\,]$$

$$\longrightarrow (\nu k)a[\ n\{\!|\,\mathtt{out}\ a.\mathtt{in}\ b.\langle M\rangle^{\hat{}}\,|\!\}_k\ ] \mid b[\,\overline{\mathtt{in}}\ \{x\}_k.(y)^x.P\,]$$

$$\longrightarrow (\nu k)a[\ \ ] \mid n\{\!|\,\mathtt{in}\ b.\langle M\rangle^{\hat{}}\,|\!\}_k \mid b[\,\overline{\mathtt{in}}\ \{x\}_k.(y)^x.P\,]$$

Sealed Ambients

➤ The mechanism to resume to a fully operational state is associated to movements and co-capabilities containing keys

$$n\{\!|\,\mathtt{in}\ m.P \mid Q\,|\!\}_k \mid m\{\,\overline{\mathtt{in}}\ \{x\}_k.R \mid R'\,\} \longrightarrow m\{\,n[\,P \mid Q\,] \mid R\{x := n\} \mid R'\,\}$$

Example:

$$(\nu k)a[\ n[\,\mathsf{seal}\ k.\mathtt{out}\ a.\mathtt{in}\ b.\langle M\rangle^{\hat{}}\,]\ ] \mid b[\,\overline{\mathtt{in}}\ \{x\}_k.(y)^x.P\,]$$

$$\longrightarrow (\nu k)a[\ n\{\!|\,\mathtt{out}\ a.\mathtt{in}\ b.\langle M\rangle^{\hat{}}\,|\!\}_k\ ] \mid b[\,\overline{\mathtt{in}}\ \{x\}_k.(y)^x.P\,]$$

$$\longrightarrow (\nu k)a[\ \ ] \mid n\{\!|\,\mathtt{in}\ b.\langle M\rangle^{\hat{}}\,|\!\}_k \mid b[\,\overline{\mathtt{in}}\ \{x\}_k.(y)^x.P\,]$$

$$\longrightarrow (\nu k)a[\ \ ] \mid b[\,n[\,\langle M\rangle^{\hat{}}\,] \mid (y)^n.P\{x := n\}\,]$$

Sealed Ambients

➤ The mechanism to resume to a fully operational state is associated to movements and co-capabilities containing keys

$$n\{\!|\, \mathtt{in}\ m.P \mid Q \,|\!\}_k \mid m\{\, \overline{\mathtt{in}}\ \{x\}_k.R \mid R' \,\} \longrightarrow m\{\, n[\, P \mid Q \,] \mid R\{x := n\} \mid R' \,\}$$

Example:

$$(\nu k)a[\ n[\, \mathsf{seal}\ k.\mathtt{out}\ a.\mathtt{in}\ b.\langle M\rangle^{\hat{}}\,]\ ] \mid b[\, \overline{\mathtt{in}}\ \{x\}_k.(y)^x.P \,]$$

$$\longrightarrow (\nu k)a[\ n\{\!|\, \mathtt{out}\ a.\mathtt{in}\ b.\langle M\rangle^{\hat{}} \,|\!\}_k\ ] \mid b[\, \overline{\mathtt{in}}\ \{x\}_k.(y)^x.P \,]$$

$$\longrightarrow (\nu k)a[\quad] \mid n\{\!|\, \mathtt{in}\ b.\langle M\rangle^{\hat{}} \,|\!\}_k \mid b[\, \overline{\mathtt{in}}\ \{x\}_k.(y)^x.P \,]$$

$$\longrightarrow (\nu k)a[\quad] \mid b[\, n[\, \langle M\rangle^{\hat{}}\,] \mid (y)^n.P\{x := n\} \,]$$

Secrecy and Adversaries

Intuitively:

> A process preserves the secrecy of a piece of data M if it does not publish M, or anything that would permit the computation of M.

≪ ⋘ ⋙ ≫

Secrecy and Adversaries

Intuitively:

> A process preserves the secrecy of a piece of data M if it does not publish M, or anything that would permit the computation of M.

S-Adversary: A context $\mathbf{A}(-)$ which initially knows names and capabilities in S.

Revealing Names: P may reveal n to S if there exists an S-adversary $\mathbf{A}(-)$ and a name $c \in S$ such that:

$$\mathbf{A}(P) \Longrightarrow \mathbf{C}(c[\,\langle n\rangle^{\wedge} \mid Q\,]) \qquad (\,c \text{ free}\,)$$

Secrecy and Adversaries

Intuitively:

> A process preserves the secrecy of a piece of data M if it does not publish M, or anything that would permit the computation of M.

S-Adversary: A context $\mathbf{A}(-)$ which initially knows names and capabilities in S.

Revealing Names: P may reveal n to S if there exists an S-adversary $\mathbf{A}(-)$ and a name $c \in S$ such that:

$$\mathbf{A}(P) \Longrightarrow \mathbf{C}(c[\,\langle n \rangle^{\hat{}} \mid Q\,]) \qquad (\,c \text{ free}\,)$$

Typing System: Secrecy is captured by a type system $\vdash$ which may classify processes as untrusted $\Gamma \vdash P : \mathsf{Un}$, and data as public $a : \mathsf{Public}$ if it can be exchanged with untrusted process.

Secrecy Theorem: Well-typed processes do not reveal their secrets publicly. Formally, if $\Gamma \vdash P : \mathsf{Un}$ and $\Gamma \nvdash s : \mathsf{Public}$, then P preserves the secrecy of s from all public channels, i.e. from $\{a \mid \Gamma \vdash a : \mathsf{Public}\}$. (Payload s won't be entrusted to a public a.)

That's all good, but ...

Between Theory and Practice

That's all good, but . . .

> . . . one cannot type the Internet

The gap between theory and practice matters in practice.

All this only works as long as you trust the certified types, or are willing to typecheck migrating code yourself (bytecode verification, PCC,. . .), . . .

Between Theory and Practice

That's all good, but ...

...one cannot type the Internet

The gap between theory and practice matters in practice.

All this only works as long as you trust the certified types, or are willing to typecheck migrating code yourself (bytecode verification, PCC,...), ...

- Verification (relates to type checking/inference)
- Certificates (groups as certified roles)
- Trust: In UbiComp, security must work be coupled with trust management.

Which is hard, because of delegation and dynamic policies

Trust Management

Focus on Trust Evolution and Delegation in Dynamic Networks

$$a\{\ A\ \}_{\pi}$$

≪ ⋘ ⋙ ≫

Trust Management

Focus on Trust Evolution and Delegation in Dynamic Networks

principal $\mathcal{P}$ → $a\{\ A\ \}_\pi$ ← trust policy: expressions on lattice $(D, \leq)$

↑ agent/behaviour

Trust Management

Focus on Trust Evolution and Delegation in Dynamic Networks

principal $\mathcal{P}$ → $a\{\ A\ \}_\pi$ ← trust policy: expressions on lattice $(D, \leq)$

agent/behaviour ↗

Trust Based Services:

$$a\{\ b.\ell\langle v\rangle\,.\,A\ \}_{\_} \mid b\{\ \ell(x)\,.\,\Sigma_t B_t\ \}_\pi \longrightarrow a\{\ A\ \}_{\_} \mid b\{\ B_{\pi(a)}\{x := v\}\ \}_\pi$$

Trust Evolution:

$$a\{\ \zeta\,.\,A \mid B\ \}_\pi \longrightarrow a\{\ A \mid B\ \}_{\zeta(\pi)}$$

Trust Management

Focus on Trust Evolution and Delegation in Dynamic Networks

principal $\mathcal{P}$ → $a\{\ A\ \}_\pi$ ← trust policy: expressions on lattice $(D, \leq)$

agent/behaviour ↗

Trust Based Services:

$$a\{\ b.\ell\langle v\rangle\,.\,A\ \}_\_ \mid b\{\ \ell(x)\,.\,\Sigma_t B_t\ \}_\pi \longrightarrow a\{\ A\ \}_\_ \mid b\{\ B_{\pi(a)}\{x := v\}\ \}_\pi$$

Trust Evolution:

$$a\{\ \zeta\,.\,A \mid B\ \}_\pi \longrightarrow a\{\ A \mid B\ \}_{\zeta(\pi)}$$

Policies and Expressions:

$\pi ::=$	$\ulcorner p \urcorner$	delegation	$\tau ::=$	$t \in D$	value/var
	$\lambda x : \mathrm{P}.\tau$	abstraction		$\pi(p)$	policy value
	$\mathrm{op}(\pi_1, \ldots, \pi_n)$	lattice op		$e \mapsto \tau;\tau$	choice
$p ::=$	$a \in \mathcal{P},\ \ x : \mathrm{P}$	principal/vars	$e ::=$	$\tau\ \mathrm{cmp}\ \tau\ ,\ p\ \mathrm{eq}\ p$	comparisons
				$e\ \mathrm{bop}\ e$	boolean op

Understanding Delegation

Example:

a :	$p \mapsto \mathtt{trusted}$;	b :	$p \mapsto \ulcorner a \urcorner(p)$;
	$q \mapsto \ulcorner b \urcorner(q)$;		$q \mapsto \mathtt{untrusted}$;
	$z \mapsto \ulcorner p \urcorner(z)$;		$z \mapsto \ulcorner a \urcorner(z)$;

Understanding Delegation

Example:

$a:$	$p \mapsto \text{trusted};$	$b:$	$p \mapsto \ulcorner a \urcorner(p);$
	$q \mapsto \ulcorner b \urcorner(q);$		$q \mapsto \text{untrusted};$
	$z \mapsto \ulcorner p \urcorner(z);$		$z \mapsto \ulcorner a \urcorner(z);$

Delegation, formally: Global trust as a fixpoint.

$$\pi : (\mathcal{P} \to \mathcal{P} \to D) \to (\mathcal{P} \to D)$$ Local Policy

$$\Xi : (\mathcal{P} \to \mathcal{P} \to D) \to (\mathcal{P} \to \mathcal{P} \to D)$$ Collected Policies

Global Trust: $\mathit{fix}(\Xi) : \mathcal{P} \to \mathcal{P} \to D$. But, is this good enough?

Understanding Delegation

Example:

$a:$	$p \mapsto$ trusted;	$b:$	$p \mapsto \ulcorner a \urcorner(p)$;
	$q \mapsto \ulcorner b \urcorner(q)$;		$q \mapsto$ untrusted;
	$z \mapsto \ulcorner p \urcorner(z)$;		$z \mapsto \ulcorner a \urcorner(z)$;

Delegation, formally: Global trust as a fixpoint.

$$\pi : (\mathcal{P} \to \mathcal{P} \to D) \to (\mathcal{P} \to D)$$ Local Policy

$$\Xi : (\mathcal{P} \to \mathcal{P} \to D) \to (\mathcal{P} \to \mathcal{P} \to D)$$ Collected Policies

Global Trust: $\mathit{fix}(\Xi) : \mathcal{P} \to \mathcal{P} \to D$. But, is this good enough?

p : trusted	q : untrusted	z : ???

Cannot confuse don't trust with don't know: the value of $\ulcorner p \urcorner(z)$ could become available later.

Need to account for uncertain knowledge of $\ulcorner p \urcorner(z) \in D$.

Trust Structures

$(D, \leq, \sqsubseteq)$, where $\bigvee$ is $\sqsubseteq$-continous

trust lattice — approximation cpo

Thm. $(D, \leq, \sqsubseteq)$ yields an adequate semantics $[\![-]\!] : \mathsf{Policies} \to \mathsf{Env} \to (\mathcal{P} \to \mathcal{P} \to D)$. The trust structure is derived canonically from $(D, \leq)$. The fixpoint is computed with respect to $\sqsubseteq$.

$$[\![\pi_{p_1}, \ldots, \pi_{p_n}]\!]_\sigma = \mathit{fix}_{\sqsubseteq}(\lambda m.\lambda p.(\![\, \pi_p \,]\!)_{\sigma m})$$

Ongoing work:

- Approximate the fixpoint in the presence of partial information.
- Use Kripke style semantics to capture trust evolution in time.
- Static safety guarantees: processes do not undermine their site policies.

Dimensions, Capacities, Mobility

Central Notion: Resource Usage

Focus: Capacity Bounds Awareness.

- Bounded Capacity Ambients
- Fine control of capacity.
- Space as a linear co-capability.

$$a[\,\mathtt{in}\,b.P \mid Q\,] \mid b[\,\blacksquare \mid R\,] \searrow \blacksquare \mid b[\,a[\,P \mid Q\,] \mid R\,]$$

move capability

space co-capability

Computation takes space, dynamically, and we'd like to model it.

Fundamentals: Space Conscious Movement

$$a[\ \mathtt{in}\, b.P \mid Q\ ] \mid b[\ \rule{1em}{0.3em} \mid R\ ] \quad \searrow \quad \rule{1em}{0.3em} \mid b[\ a[\ P \mid Q\ ] \mid R\ ]$$

$$\rule{1em}{0.3em} \mid b[\ a[\ \mathtt{out}\, b.P \mid Q\ ] \mid R\ ] \quad \searrow \quad a[\ P \mid Q\ ] \mid b[\ \rule{1em}{0.3em} \mid R\ ]$$

A Calculus of Bounded Capacities: Movement

Fundamentals: Space Conscious Movement

$$a[\,\mathtt{in}\,b.P \mid Q\,] \mid b[\,\blacksquare \mid R\,] \;\searrow\; \blacksquare \mid b[\,a[\,P \mid Q\,] \mid R\,]$$

$$\blacksquare \mid b[\,a[\,\mathtt{out}\,b.P \mid Q\,] \mid R\,] \;\searrow\; a[\,P \mid Q\,] \mid b[\,\blacksquare \mid R\,]$$

Example: Travelling needs but consumes no space.

$$a[\,\mathtt{in}\,b.\mathtt{in}\,c.\mathtt{out}\,c.\mathtt{out}\,b.\mathbf{0}\,] \mid b[\,\blacksquare \mid c[\,\blacksquare\,]\,]$$

$$\searrow\searrow\; \blacksquare \mid b[\,\blacksquare \mid c[\,a[\,\mathtt{out}\,c.\mathtt{out}\,b.\mathbf{0}\,]\,]\,]$$

$$\searrow\searrow\; a[\,\mathbf{0}\,] \mid b[\,\blacksquare \mid c[\,\blacksquare\,]\,]$$

A Calculus of Bounded Capacities: Sizes

Fundamentals: Space Conscious Movement

- But the size of travellers matters!

$$a^k[\,\mathtt{in}\,b.P \mid Q\,] \mid b[\,\overbrace{\_ \mid \ldots \mid \_}^{k \text{ times}} \mid R\,] \;\searrow\; \overbrace{\_ \mid \ldots \mid \_}^{k \text{ times}} \mid b[\,a^k[\,P \mid Q\,] \mid R\,]$$

$$\underbrace{\_ \mid \ldots \mid \_}_{k \text{ times}} \mid b[\,a^k[\,\mathtt{out}\,b.P \mid Q\,] \mid R\,] \;\searrow\; a^k[\,P \mid Q\,] \mid b[\,\underbrace{\_ \mid \ldots \mid \_}_{k \text{ times}} \mid R\,]$$

A Calculus of Bounded Capacities: Sizes

Fundamentals: Space Conscious Movement

➤ But the size of travellers matters!

$$a^k[\,\mathtt{in}\,b.P \mid Q\,] \mid b[\,\overbrace{\_ \mid \ldots \mid \_}^{k \text{ times}} \mid R\,] \searrow \overbrace{\_ \mid \ldots \mid \_}^{k \text{ times}} \mid b[\,a^k[\,P \mid Q\,] \mid R\,]$$

$$\underbrace{\_ \mid \ldots \mid \_}_{k \text{ times}} \mid b[\,a^k[\,\mathtt{out}\,b.P \mid Q\,] \mid R\,] \searrow a^k[\,P \mid Q\,] \mid b[\,\underbrace{\_ \mid \ldots \mid \_}_{k \text{ times}} \mid R\,]$$

What is the a^k? A type annotation measuring the size of P.

Notation. We use $\_^k$ as a shorthand for $\overbrace{\_ \mid \ldots \mid \_}^{k \text{ times}}$.

Fundamentals: Space Conscious Process Activation

$$\triangleright^k P \mid \blacksquare^k \searrow P$$

Fundamentals: Space Conscious Process Activation

$$\triangleright^{k} P \mid \blacksquare^{k} \searrow P$$

passive process: weighs 0

P weighs k

A Calculus of Bounded Capacities: Spawning

Fundamentals: Space Conscious Process Activation

$$\triangleright^k P \mid \rule{1em}{0.3em}^k \;\searrow\; P$$

passive process: weighs 0 ($\triangleright^k$)

P weighs k

Example: Replication: $!^k \triangleq\ !\triangleright^k$

$$!\triangleright^k P \mid \rule{1em}{0.3em}^k \;\searrow\; !\triangleright^k P \mid P$$

Types ensure only 0-weighted processes are replicable: One must use spawning, so that replication needs space proportional to the process' weight.

A Calculus of Bounded Capacities: Transfer

Fundamentals: Space Acquisition and Release

$$\mathtt{put}^{\check{}} a.P \mid \blacksquare \mid a^k[\,\mathtt{get}^{\hat{}}.Q \mid R\,] \searrow P \mid a^{k+1}[\,Q \mid \blacksquare \mid R\,]$$

$$a^{k+1}[\,\mathtt{put}.P \mid \blacksquare \mid S\,] \mid b^h[\,\mathtt{get}\,a.Q \mid R\,] \searrow a^k[\,P \mid S\,] \mid b^{h+1}[\,Q \mid \blacksquare \mid R\,]$$

≪ ⋘ ⋙ ≫

Fundamentals: Space Acquisition and Release

$$\mathtt{put}^{\check{}}\, a\,.\,P \mid \blacksquare \mid a^k[\,\mathtt{get}^{\hat{}}\,.\,Q \mid R\,] \searrow P \mid a^{k+1}[\,Q \mid \blacksquare \mid R\,]$$

$$a^{k+1}[\,\mathtt{put}\,.\,P \mid \blacksquare \mid S\,] \mid b^h[\,\mathtt{get}\,a\,.\,Q \mid R\,] \searrow a^k[\,P \mid S\,] \mid b^{h+1}[\,Q \mid \blacksquare \mid R\,]$$

Example: A Memory Module

$$\mathsf{memMod} \triangleq \mathsf{mem}[\,\blacksquare^{256MB} \mid !\mathtt{put} \mid !\mathtt{get}\,\mathsf{free}\,]$$

$$\mathsf{malloc} \triangleq \mathsf{m}[\,!\mathtt{get}\,\mathsf{mem}\,.\,\mathsf{free}[\,\mathtt{out}\,\mathsf{m}\,.\,\mathtt{get}\,\mathsf{m}\,.\,\mathtt{put}\,] \mid !\mathtt{put}\,]$$

A Calculus of Bounded Capacities: Transfer

Fundamentals: Space Acquisition and Release

$$\mathtt{put}^{\check{}}\, a\,.\,P \mid \_ \mid a^{k}[\,\mathtt{get}^{\hat{}}\,.\,Q \mid R\,] \;\searrow\; P \mid a^{k+1}[\,Q \mid \_ \mid R\,]$$

$$a^{k+1}[\,\mathtt{put}\,.\,P \mid \_ \mid S\,] \mid b^{h}[\,\mathtt{get}\;a\,.\,Q \mid R\,] \;\searrow\; a^{k}[\,P \mid S\,] \mid b^{h+1}[\,Q \mid \_ \mid R\,]$$

Example: A Memory Module

$$\mathsf{memMod} \triangleq \mathsf{mem}[\,\_^{256MB} \mid !\mathtt{put} \mid !\mathtt{get}\,\mathsf{free}\,]$$

$$\mathsf{malloc} \triangleq \mathsf{m}[\,!\mathtt{get}\,\mathsf{mem}\,.\,\mathsf{free}[\,\mathtt{out}\,\mathsf{m}\,.\,\mathtt{get}\,\mathsf{m}\,.\,\mathtt{put}\,] \mid !\mathtt{put}\,]$$

$$\mathsf{memMod} \mid \mathsf{malloc} \searrow^{256MB} \mathsf{mem}[\,!\mathtt{put} \mid !\mathtt{get}\,\mathsf{free}\,] \mid \mathsf{m}[\,\_^{256MB} \mid \ldots\,] \searrow^{2\times256MB}$$

$$\mathsf{mem}[\,!\mathtt{put} \mid !\mathtt{get}\,\mathsf{free}\,] \mid \mathsf{malloc} \mid \mathsf{free}^{256MB}[\,\_ \mid \mathtt{put}\,] \searrow^{256MB} \mathsf{memMod} \mid \mathsf{malloc} \mid \ldots$$

A System of Capacity Types

Capacity Types: $\phi, \ldots$ are pairs of nats $[n, N]$, with $n \leq N$.

Effect Types $\mathcal{E}, \ldots$ are pairs of nats (d, i), representing decs and incs.

Exchange Types: $\chi ::= \mathsf{Shh} \mid \mathsf{Amb}\langle \sigma, \chi \rangle \mid \mathsf{Cap}\langle \mathcal{E}, \chi \rangle$

Process and Ambient and Capability Types:

$a : \mathsf{Amb}\langle \phi, \chi \rangle$	a has no less than ϕ_{m} and no more than ϕ_{M} spaces
$P : \mathsf{Proc}\langle k, \mathcal{E}, \chi \rangle$	P weighs k and produces the effect $\mathcal{E}$ on ambients
$C : \mathsf{Cap}\langle \mathcal{E}, \chi \rangle$	C transforms processes adding $\mathcal{E}$ to their effects

A System of Capacity Types

Capacity Types: $\phi, \ldots$ are pairs of nats $[n, N]$, with $n \leq N$.

Effect Types $\mathcal{E}, \ldots$ are pairs of nats (d, i), representing decs and incs.

Exchange Types: $\chi ::= \mathsf{Shh} \mid \mathsf{Amb}\langle \sigma, \chi \rangle \mid \mathsf{Cap}\langle \mathcal{E}, \chi \rangle$

Process and Ambient and Capability Types:

$a : \mathsf{Amb}\langle \phi, \chi \rangle$	a has no less than ϕ_{m} and no more than ϕ_{M} spaces
$P : \mathsf{Proc}\langle k, \mathcal{E}, \chi \rangle$	P weighs k and produces the effect $\mathcal{E}$ on ambients
$C : \mathsf{Cap}\langle \mathcal{E}, \chi \rangle$	C transforms processes adding $\mathcal{E}$ to their effects

Thm: Subject Reduction: Well-typed processes preserve space.

If $\Gamma \vdash P : \mathsf{Proc}\langle k, \mathcal{E}, \chi \rangle$ and $P \searrow Q$ then $\Gamma \vdash Q : \mathsf{Proc}\langle k, \mathcal{E}', \chi \rangle$ for some $\mathcal{E}' \lessdot \mathcal{E}$.

Future Work

- What: Third-Party Resources: Models, Languages and Techniques
  - Resource-Aware Computation: resource bounds negotiation & enforcement
  - Resource Usage: calculi & logics for quantitative analysis
  - Resource Safety: languages for security & trust policies; general resource logics
  - Resource Trust: history-based: theory and infrastructures

Future Work

- What: Third-Party Resources: Models, Languages and Techniques
  - Resource-Aware Computation: resource bounds negotiation & enforcement
  - Resource Usage: calculi & logics for quantitative analysis
  - Resource Safety: languages for security & trust policies; general resource logics
  - Resource Trust: history-based: theory and infrastructures
- How: Integrated approach: Behavioural, Execution, Abstract Models.

Future Work

- What: Third-Party Resources: Models, Languages and Techniques
  - Resource-Aware Computation: resource bounds negotiation & enforcement
  - Resource Usage: calculi & logics for quantitative analysis
  - Resource Safety: languages for security & trust policies; general resource logics
  - Resource Trust: history-based: theory and infrastructures
- How: Integrated approach: Behavioural, Execution, Abstract Models.
- Who: Communities involved:
  - EU FET Global Computing
  - UKCRC Great Challenges: Science for Global Ubiquitous Computing
  - EPSRC UK eScience & UK UbiNet

Contexts as Labels

The intuition:

$$a \xrightarrow{\mathscr{C}} b \text{ iff } \mathscr{C}[a] \searrow b$$

For instance:

$$a \xrightarrow{-|\bar{a}} \mathbf{0} \qquad M \xrightarrow{(\lambda x.-)N} M\{N/x\} \qquad \mathbf{K}M \xrightarrow{-N} M$$

Yep, but not quite:

- Too many labels not desirable:
  - Useless combinatorial explosion: $\lambda x.xx \xrightarrow{-MN} MMN$
  - Messes up the bisimulation (too coarse): $l \xrightarrow{\mathscr{D}} \mathscr{D}[r]$ for all rules $l \searrow r$.

≪ ⋘ ⋙ ≫

Contexts as Labels

The intuition:

$$a \xrightarrow{\mathscr{C}} b \text{ iff } \mathscr{C}[a] \searrow b$$

For instance:

$$a \xrightarrow{-|\bar{a}} \mathbf{0} \qquad M \xrightarrow{(\lambda x.-)N} M\{N/x\} \qquad \mathbf{K}M \xrightarrow{-N} M$$

Yep, but not quite:

- Too many labels not desirable:
  - Useless combinatorial explosion: $\lambda x.xx \xrightarrow{-MN} MMN$
  - Messes up the bisimulation (too coarse): $l \xrightarrow{\mathscr{D}} \mathscr{D}[r]$ for all rules $l \searrow r$.

Choose only 'minimal' redex-enabling contexts

- Case analysis of basic situations: Sewell. Abstract approach: Leifer-Milner

$$\boxed{\boxed{\boxed{a} \mid \bar{a}} \mid a}$$

A Categorical Approach

Lawvere theory on Σ

- the natural numbers for objects
- a morphism $t\colon m \to n$, for t a n-tuple of m-holed terms.
- Composition is substitution of terms into holes.

E.G. for Σ the signature for arithmetics:

- term $(-_1 \times x) + -_2$ is an arrow $2 \to 1$ (two holes yielding one term)
- $\langle 3, 2 \times y \rangle$ is an arrow $0 \to 2$ (a pair of terms with no holes).
- Their composition is the term $(3 \times x) + (2 \times y)$, an arrow of type $0 \to 1$.

A Categorical Approach

Lawvere theory on Σ

- the natural numbers for objects
- a morphism $t\colon m \to n$, for t a n-tuple of m-holed terms.
- Composition is substitution of terms into holes.

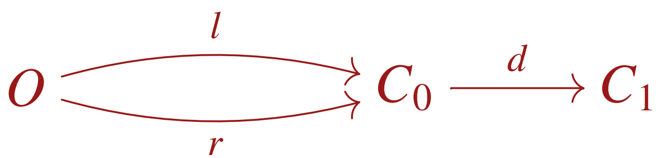
A generalisation from term rewriting systems to categories.

- A category $\mathbb{C}$ with distinguished object 0.
- A set of reaction rules $\mathcal{R} \subseteq \bigcup_{C \in \mathbb{C}} \mathbb{C}(0, C) \times \mathbb{C}(0, C)$.
- A set $\mathbb{D}$ of arrows of $\mathbb{C}$ called the reactive contexts.

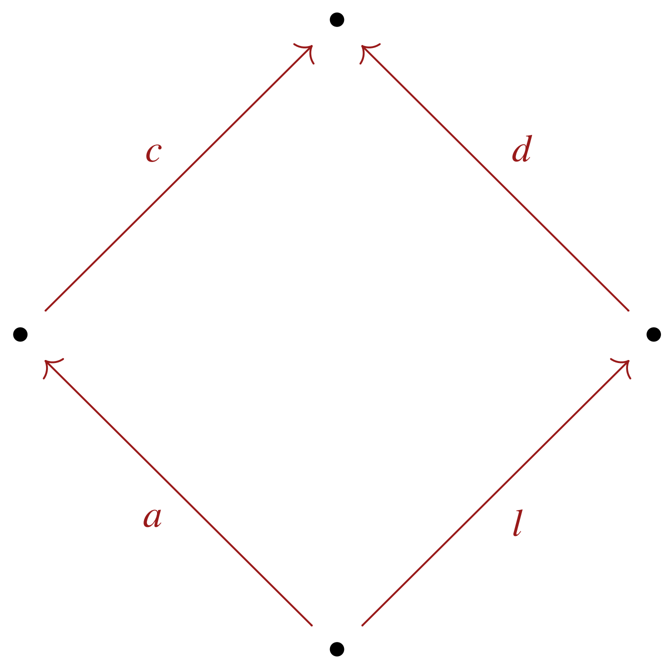
  Assume that $d_0 \,.\, d_1 \in \mathbb{D}$ implies d_0 and $d_1 \in \mathbb{D}$.

The reaction relation is defined as

$$a \longrightarrow b \quad \text{iff} \quad a = d\,.\,l,\; b = d\,.\,r,\; d \in \mathbb{D} \text{ and } \langle l, r\rangle \in \mathcal{R}.$$

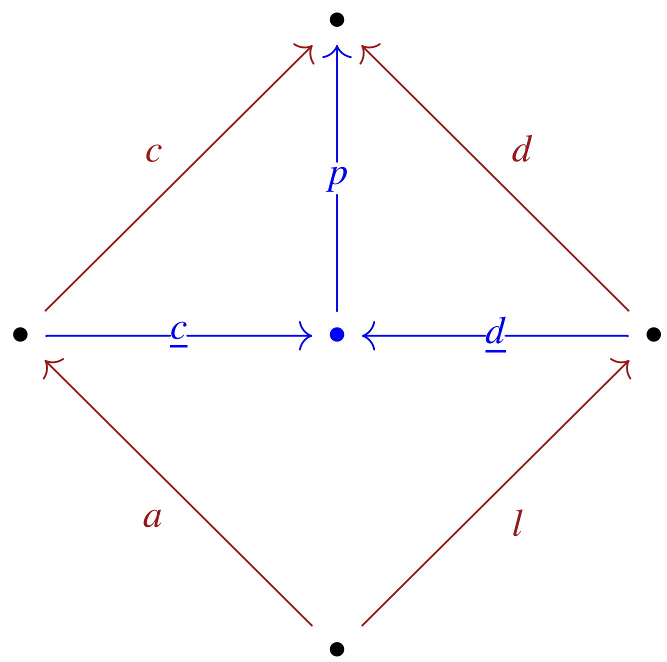


Suppose that $\mathbb{C}$ is a (bi)category and consider a redex square



(G)RPOs

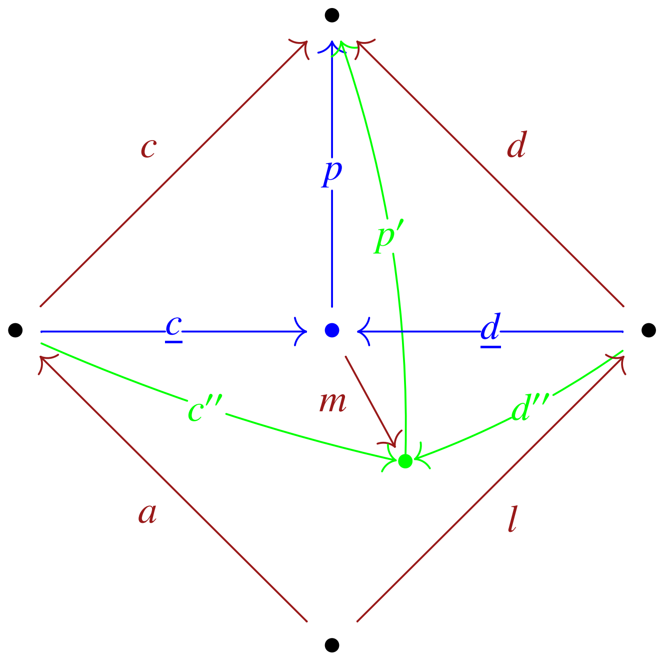
Suppose that $\mathbb{C}$ is a (bi)category and consider a redex square



- a relative pushout (RPO) is a tuple $\langle \underline{c}, \underline{d}, p \rangle$ which satisfies the universal property that:

(G)RPOs

Suppose that $\mathbb{C}$ is a (bi)category and consider a redex square

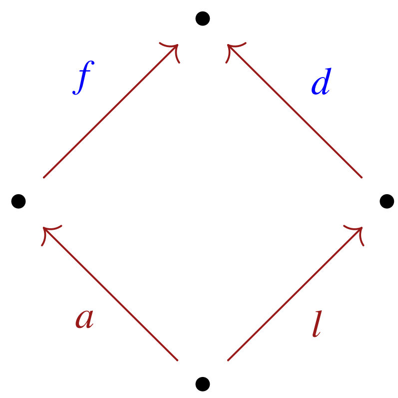


- a relative pushout (RPO) is a tuple $\langle \underline{c}, \underline{d}, p \rangle$ which satisfies the universal property that:
- for any other such $\langle c', d', p' \rangle$ there exists a unique mediating morphism m.

Deriving LTS

The LTS derived from the reactive system has:

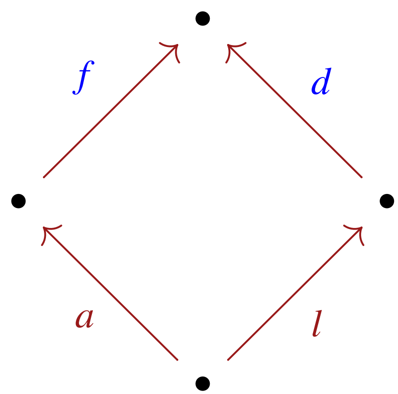
- Nodes: $a : O \to N$
- Transitions: $a \xrightarrow{f} dr$ iff for $\langle l, r \rangle \in \mathcal{R}$ and $d \in \mathbb{D}$, $\langle f, d, \mathrm{id} \rangle$ is a relative pushout of the square



Deriving LTS

The LTS derived from the reactive system has:

- Nodes: $a : O \rightarrow N$
- Transitions: $a \xrightarrow{f} dr$ iff for $\langle l, r \rangle \in \mathcal{R}$ and $d \in \mathbb{D}$, $\langle f, d, \mathrm{id} \rangle$ is a relative pushout of the square



- Thm. If all redex squares like the above have (G)RPOs then the bisimulation on the derived LTS is a congruence.
  [Leifer-Milner 00, Sassone-Sobocinski 02]

Applying (G)RPOs

- Milner (2001) worked out RPOs for a graphical formalism called bigraphs.
- Sassone and Sobocinski (2002) introduced GRPOs to handle calculi with non-trivial structural congruences.
- Jensen and Milner (2003) derived (essentially) the usual π labelled bisimulation on asynchronous π using RPOs.
- Sassone and Sobocinski (2003) worked out an easy encoding of Milner's pre-category approach into the G-world.
- Jensen and Milner (2004) found (G)RPOs for ambient-calculus and for weak bisimulations.
- Sassone and Sobocinski (2004) derived GRPOs for generic graph structures and graph rewrite systems.

Applying (G)RPOs

- Milner (2001) worked out RPOs for a graphical formalism called bigraphs.
- Sassone and Sobocinski (2002) introduced GRPOs to handle calculi with non-trivial structural congruences.
- Jensen and Milner (2003) derived (essentially) the usual π labelled bisimulation on asynchronous π using RPOs.
- Sassone and Sobocinski (2003) worked out an easy encoding of Milner's pre-category approach into the G-world.
- Jensen and Milner (2004) found (G)RPOs for ambient-calculus and for weak bisimulations.
- Sassone and Sobocinski (2004) derived GRPOs for generic graph structures and graph rewrite systems.

So far, the price of the initial 2-categorical investment seems worth paying. . .

Future Work

- Extend to more complicated process calculi with complex structural congruences (e.g. replication); Apply to specific graph rewriting systems.