Dynamic and Local Typing for Mobile Ambients

MYTHS / MIKADO / DART Meeting, Venice, June 14, 2004

M. Coppo[1], M. Dezani[1], E. Giovannetti[1], R. Pugliese[2]

(1) Dipartimento di Informatica – Università di Torino

(2) Dip. di Sistemi e Informatica – Università di Firenze

Modelling of wide-area distributed and mobile computing:

- interacting components from different locations are unknown or only partially known to each other

$$\Downarrow$$

- each component must carry behavioural information, to be checked at runtime

$$\Gamma \quad \vdash \quad t \quad : \quad T$$

assumptions on the world $\vdash$ component : behavioural properties

A proposal

A typed ambient calculus with:

- behavioural type assumptions local to each ambient
- no global type assumptions on ambient names
- ambient types attached to single ambient constructions, not to ambient names
- runtime types used to check compatibility between components from different localities

Specific features of the calculus:

- no ambient opening
- only local communication
- general process mobility

The typed calculus: mobility primitives

- ambient mobility: in, out

$$\text{(R-in)} \quad n[\,\mathsf{in}\ m\,.\,P \mid Q\,] \mid m[R] \;\longrightarrow\; m[\,n[\,P \mid Q\,] \mid R\,]$$

$$\text{(R-out)} \quad m[\,n[\,\mathsf{out}\ m\,.\,P \mid Q\,] \mid R\,] \;\longrightarrow\; n[\,P \mid Q\,] \mid m[R]$$

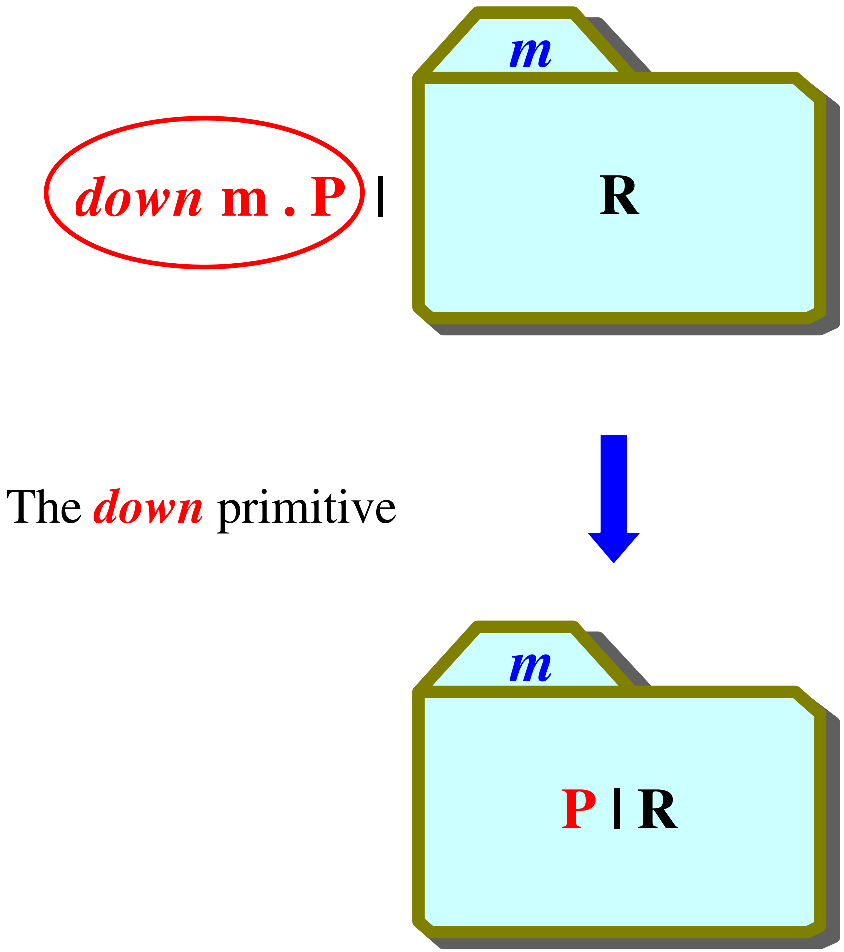
- process mobility: down, up

$$\mathsf{down}\ n\,.\,P \mid n[Q] \;\rightarrow\; n[P \mid Q]$$

$$m[n[\mathsf{up}\ m\,.\,P \mid Q] \mid R] \;\rightarrow\; m[P \mid n[Q] \mid R]$$

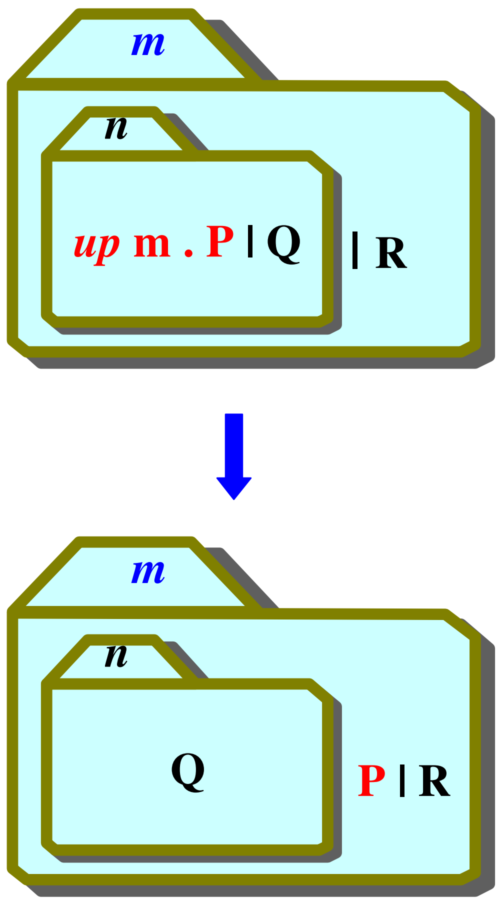
Process mobility: **down**

down m moves the continuation process
from its ambient down to an enclosed ambient m



Process mobility: up

up m moves the continuation process
from its ambient up to the enclosing ambient m



The *up* primitive

Types

- describe communication and mobility properties

  An ambient has:

Types

- describe communication and mobility properties

  An ambient has:

  active mobility – static typing: the ambients it may cross and the ambients it may send processes to;

Types

- describe communication and mobility properties

  An ambient has:

  active mobility – static typing: the ambients it may cross and the ambients it may send processes to;

  passive mobility – dynamic typing: the ambients by which it may be crossed and the ones by which it may be sent processes to.

Types

- describe communication and mobility properties

  An ambient has:

  active mobility – static typing: the ambients it may cross and the ambients it may send processes to;

  passive mobility – dynamic typing: the ambients by which it may be crossed and the ones by which it may be sent processes to.

  A process has:

  active mobility – static typing: the ambients to which it may drive its enclosing ambient, and the ones to which it may go.

Types

- describe communication and mobility properties

  An ambient has:

  active mobility – static typing: the ambients it may cross and the ambients it may send processes to;

  passive mobility – dynamic typing: the ambients by which it may be crossed and the ones by which it may be sent processes to.

  A process has:

  active mobility – static typing: the ambients to which it may drive its enclosing ambient, and the ones to which it may go.

- are based on ambient groups:

Types

- describe communication and mobility properties

  An ambient has:

  active mobility – static typing: the ambients it may cross and the ambients it may send processes to;

  passive mobility – dynamic typing: the ambients by which it may be crossed and the ones by which it may be sent processes to.

  A process has:

  active mobility – static typing: the ambients to which it may drive its enclosing ambient, and the ones to which it may go.

- are based on ambient groups:
  - a group is a name that labels a set of ambients

Types

- describe communication and mobility properties

  An ambient has:

  active mobility – static typing: the ambients it may cross and the ambients it may send processes to;

  passive mobility – dynamic typing: the ambients by which it may be crossed and the ones by which it may be sent processes to.

  A process has:

  active mobility – static typing: the ambients to which it may drive its enclosing ambient, and the ones to which it may go.

- are based on ambient groups:
  - a group is a name that labels a set of ambients
  - different ambients with the same name
    may belong to different groups

Types

- describe communication and mobility properties

  An ambient has:

  active mobility – static typing: the ambients it may cross and the ambients it may send processes to;

  passive mobility – dynamic typing: the ambients by which it may be crossed and the ones by which it may be sent processes to.

  A process has:

  active mobility – static typing: the ambients to which it may drive its enclosing ambient, and the ones to which it may go.

- are based on ambient groups:
  - a group is a name that labels a set of ambients
  - different ambients with the same name may belong to different groups
  - mobility properties are expressed via groups

Static types: active mobility

- process types $\quad Pro \quad ::= \quad g(G)$
  - g group names
  - $G ::= \mathsf{mc}(\mathscr{C}, \mathscr{E}, T)$ *mobcom* (mobility + communication) types, where:

Static types: active mobility

- process types $\quad Pro \quad ::= \quad g(G)$
  - g group names
  - $G ::= \mathsf{mc}(\mathscr{C}, \mathscr{E}, T)$ *mobcom* (mobility + communication) types, where:

    $\mathscr{C}$ set of groups of ambients into which the process may drive (through an in or out action) its enclosing ambient

Static types: active mobility

- process types $\quad Pro \quad ::= \quad g(G)$
  - g group names
  - $G ::= \mathsf{mc}(\mathscr{C}, \mathscr{E}, T)$ *mobcom* (mobility + communication) types, where:

    $\mathscr{C}$ set of groups of ambients into which the process may drive (through an in or out action) its enclosing ambient

    $\mathscr{E}$ is the set of (groups of) ambients to which it may send (through a down or up action) a continuation process

Static types: active mobility

- process types $\quad Pro \quad ::= \quad g(G)$
  - g group names
  - $G ::= \mathsf{mc}(\mathscr{C}, \mathscr{E}, T)$ *mobcom* (mobility + communication) types, where:

    $\mathscr{C}$ set of groups of ambients into which the process may drive (through an in or out action) its enclosing ambient

    $\mathscr{E}$ is the set of (groups of) ambients to which it may send (through a down or up action) a continuation process

    T is the process communication type

Static types: active mobility

- process types $\quad Pro \quad ::= \quad g(G)$
  - g group names
  - $G ::= \mathsf{mc}(\mathscr{C}, \mathscr{E}, T)$ *mobcom* (mobility + communication) types, where:

    $\mathscr{C}$ set of groups of ambients into which the process may drive (through an in or out action) its enclosing ambient

    $\mathscr{E}$ is the set of (groups of) ambients to which it may send (through a down or up action) a continuation process

    T is the process communication type
- ambient type $\quad$ amb (atomic type)

  no mobcom types for ambient names

Runtime packing of static types

- ambient mobility actions specify target's name and group:

  in/out $\alpha{:}g$

Runtime packing of static types

- ambient mobility actions specify target's name and group:

$$\mathsf{in/out}\ \alpha{:}g$$

- parallel processes must have, as usual, the same process type;

Runtime packing of static types

- ambient mobility actions specify target's name and group:

  in/out $\alpha{:}g$

- parallel processes must have, as usual, the same process type;
- if P is well typed with type $g(G)$ and m is an ambient name, the ambient construction of skeleton $m[P]$ is always well typed, and $g(G)$ is the ambient inner type;

Runtime packing of static types

- ambient mobility actions specify target's name and group:

  $$\text{in/out } \alpha{:}g$$

- parallel processes must have, as usual, the same process type;
- if P is well typed with type $g(G)$ and m is an ambient name, the ambient construction of skeleton $m[P]$ is always well typed, and $g(G)$ is the ambient inner type;
- a process going up or down into an ambient must have a type compatible with the ambient's inner type:
  runtime checking is needed, therefore:

Runtime packing of static types

- ambient mobility actions specify target's name and group:

  $$\text{in/out } \alpha{:}g$$

- parallel processes must have, as usual, the same process type;
- if P is well typed with type $g(G)$ and m is an ambient name, the ambient construction of skeleton $m[P]$ is always well typed, and $g(G)$ is the ambient inner type;
- a process going up or down into an ambient must have a type compatible with the ambient's inner type:
  runtime checking is needed, therefore:
  - ambient
    carries at runtime its inner type: $m{:}g(G)[X]$

Runtime packing of static types

- ambient mobility actions specify target's name and group:

  $$\text{in/out } \alpha{:}g$$

- parallel processes must have, as usual, the same process type;
- if P is well typed with type $g(G)$ and m is an ambient name, the ambient construction of skeleton $m[P]$ is always well typed, and $g(G)$ is the ambient inner type;
- a process going up or down into an ambient must have a type compatible with the ambient's inner type:
  runtime checking is needed, therefore:
  - ambient
    carries at runtime its inner type: $m{:}g(G)[X]$
  - process mobility action
    carries the type of its continuation: down/up $\alpha{:}g$ with G

The syntax of types

G	$::=$	$\mathsf{mc}(\mathscr{C}, \mathscr{E}, T)$	*mobcom* type: mobility and communication type
Pro	$::=$	$g(G)$	process type: processes of group g with mobcom type G
Cap	$::=$	$g(G) \rightsquigarrow g'(G')$	capabilities that can be consumed by processes of type $g(G)$ and leave processes of type $g'(G')$ as continuations
W	$::=$		message type
		Cap	capability type
		group	group
		amb	ambient type
T	$::=$		communication type
		shh	no communication
		$\overrightarrow{W}$	communication of messages of type $\overrightarrow{W}$
Σ	$::=$	$\varnothing$	variable environment
		$\Sigma, x : W$	

Dynamic types: passive mobility

Each ambient contains a dynamic characterization of its passive mobility.

Complete form of the ambient construct:

$$\alpha{:}g(G)[\mathsf{c}, \mathsf{e} \| P]$$

Dynamic types: passive mobility

Each ambient contains a dynamic characterization of its passive mobility.

Complete form of the ambient construct:

$$\alpha{:}g(G)[\mathsf{c}, \mathsf{e} \| P]$$

- c,e: two multisets of dynamic mobility permissions (w.r.t. itself) granted to other ambients:

Dynamic types: passive mobility

Each ambient contains a dynamic characterization of its passive mobility.

Complete form of the ambient construct:

$$\alpha{:}g(G)[\mathsf{c},\mathsf{e}\|P]$$

- c,e: two multisets of dynamic mobility permissions (w.r.t. itself) granted to other ambients:

- execution of a statically allowed action:
  - only possible if a corresponding dynamic permit is present;
  - consumes the permit

Dynamic types: passive mobility

Each ambient contains a dynamic characterization of its passive mobility.

Complete form of the ambient construct:

$$\alpha{:}g(G)[\mathsf{c}, \mathsf{e} \| P]$$

- c,e: two multisets of dynamic mobility permissions (w.r.t. itself) granted to other ambients:
- execution of a statically allowed action:
  - only possible if a corresponding dynamic permit is present;
  - consumes the permit
- elements of infinite multiplicity, representing permanent permits, may be present in multisets.

Dynamic types: passive mobility

Each ambient contains a dynamic characterization of its passive mobility.

Complete form of the ambient construct:

$$\alpha{:}g(G)[\mathsf{c},\mathsf{e}\|P]$$

- c,e: two multisets of dynamic mobility permissions (w.r.t. itself) granted to other ambients:
  - c: multiset of groups of ambients allowed to go in or out of it;
  - e: multiset of permits for processes to go up or down into it:
    - element of e: a pair $\langle g', G'\rangle$, entrance permit for a g'-process with a G'-behaviour; constraint: $G' \leq G$
- execution of a statically allowed action:
  - only possible if a corresponding dynamic permit is present;
  - consumes the permit
- elements of infinite multiplicity, representing permanent permits, may be present in multisets.

Dynamic modification of mobility permits

c and e allow and forbid movements at runtime. They can therefore be changed dynamically without breaking the subject reduction, by:

Dynamic modification of mobility permits

c and e allow and forbid movements at runtime. They can therefore be changed dynamically without breaking the subject reduction, by:

- consuming one (non-permanent) permit: automatic when a movement action is performed;

Dynamic modification of mobility permits

c and e allow and forbid movements at runtime. They can therefore be changed dynamically without breaking the subject reduction, by:

- consuming one (non-permanent) permit: automatic when a movement action is performed;
- adding a (possibly multiple) permit: explicitly by means of the permit-adding primitives:

Dynamic modification of mobility permits

c and e allow and forbid movements at runtime. They can therefore be changed dynamically without breaking the subject reduction, by:

- consuming one (non-permanent) permit: automatic when a movement action is performed;
- adding a (possibly multiple) permit: explicitly by means of the permit-adding primitives:
  - add$^{\mathsf{c}}$ g^{φ} in $m{:}g_m$ adds the group g with multiplicity φ to the c component of a local ambient of name m and group g_m

Dynamic modification of mobility permits

c and e allow and forbid movements at runtime. They can therefore be changed dynamically without breaking the subject reduction, by:

- consuming one (non-permanent) permit: automatic when a movement action is performed;
- adding a (possibly multiple) permit: explicitly by means of the permit-adding primitives:
  - $\mathsf{add}^{\mathsf{c}}\ g^{\varphi}\ \mathsf{in}\ m{:}g_m$ adds the group g with multiplicity φ to the c component of a local ambient of name m and group g_m
  - $\mathsf{add}^{\mathsf{e}}\ \langle g, G_1\rangle^{\varphi}\ \mathsf{in}\ m{:}g_m$
    adds the group/type pair $\langle g, G_1 \sqcap G_m\rangle$ with multiplicity φ to the e component of a local ambient $m{:}g_m(G_m)[\mathsf{c},\mathsf{e}\|P]$
    - intersection preserves the invariant $\langle g', G'\rangle \in \mathsf{e} \ \Rightarrow\ G' \leq G_m$

An example: a public transportation system

- a top-level untrusted ambient, named $world$, which includes named ambients representing cities, countryside, etc.

An example: a public transportation system

- a top-level untrusted ambient, named $world$, which includes named ambients representing cities, countryside, etc.
- cities in turn contain stations (which are ambients)

An example: a public transportation system

- a top-level untrusted ambient, named $world$, which includes named ambients representing cities, countryside, etc.
- cities in turn contain stations (which are ambients)
- trains are mobile ambients moving between stations

An example: a public transportation system

- a top-level untrusted ambient, named $world$, which includes named ambients representing cities, countryside, etc.
- cities in turn contain stations (which are ambients)
- trains are mobile ambients moving between stations
- travellers, represented by mobile processes, get into and off trains at the stations in order to move between cities.

An example: a public transportation system

- a top-level untrusted ambient, named $world$, which includes named ambients representing cities, countryside, etc.
- cities in turn contain stations (which are ambients)
- trains are mobile ambients moving between stations
- travellers, represented by mobile processes, get into and off trains at the stations in order to move between cities.

A simplified system

- two cities tur and flo, with their respective stations st_{tur} and st_{flo}
- one train commuting between st_{tur} and st_{flo}.

A public transportation system: stations

A station is an ambient of group g_{st} and of name st_X, with $X = tur, flor$:

$st_X{:}g_{st}(G_{st})[\mathsf{c}_{st},\ \mathsf{e}_{st} \parallel \ldots]$, where:

A public transportation system: stations

A station is an ambient of group g_{st} and of name st_X, with $X = tur, flor$:

$$st_X{:}g_{st}(G_{\mathsf{st}})[\mathsf{c}_{\mathsf{st}},\, \mathsf{e}_{\mathsf{st}} \parallel \ldots],$$ where:

- G_{st} specifies the station's active properties, statically checked:

A public transportation system: stations

A station is an ambient of group g_{st} and of name st_X, with $X = tur, flor$:

$$st_X{:}g_{st}(G_{\mathit{st}})[\mathsf{c}_{\mathit{st}},\ \mathsf{e}_{\mathit{st}} \parallel \ldots],\quad \text{where:}$$

- G_{st} specifies the station's active properties, statically checked:

$$G_{\mathit{st}} = \mathsf{mc}(\varnothing, \qquad\qquad)$$

1. a station cannot go in or out of ambients (is immobile);

A public transportation system: stations

A station is an ambient of group g_{st} and of name st_X, with $X = tur, flor$:

$$st_X : g_{st}(G_{\mathsf{st}})[\mathsf{c}_{\mathsf{st}},\ \mathsf{e}_{\mathsf{st}} \parallel \ldots],$$ where:

- G_{st} specifies the station's active properties, statically checked:

$$G_{\mathsf{st}} = \mathsf{mc}(\varnothing, \{g_{tr}, g_{city}\})$$

  1. a station cannot go in or out of ambients (is immobile);
  2. may send out processes (i.e., passengers) to the train and to the surrounding city;

A public transportation system: stations

A station is an ambient of group g_{st} and of name st_X, with $X = tur, flor$:

$$st_X{:}g_{st}(G_{\mathsf{st}})[\mathsf{c}_{\mathsf{st}},\ \mathsf{e}_{\mathsf{st}} \parallel \ldots],$$ where:

- G_{st} specifies the station's active properties, statically checked:

  $$G_{\mathsf{st}} = \mathsf{mc}(\varnothing, \{g_{tr}, g_{city}\})$$

  1. a station cannot go in or out of ambients (is immobile);
  2. may send out processes (i.e., passengers) to the train and to the surrounding city;

- c_{st}, e_{st} specify station passive properties, dynamically checked:

A public transportation system: stations

A station is an ambient of group g_{st} and of name st_X, with $X = tur, flor$:

$$st_X{:}g_{st}(G_{st})[\mathsf{c}_{st}, \mathsf{e}_{st} \parallel \ldots], \quad \text{where:}$$

- G_{st} specifies the station's active properties, statically checked:

  $$G_{st} = \mathsf{mc}(\varnothing, \{g_{tr}, g_{city}\})$$

  1. a station cannot go in or out of ambients (is immobile);
  2. may send out processes (i.e., passengers) to the train and to the surrounding city;

- c_{st}, e_{st} specify station passive properties, dynamically checked:

  1. $\mathsf{c}_{st} = \{g_{tr}^*\}$ may be crossed by trains (which are ambients);

A public transportation system: stations

A station is an ambient of group g_{st} and of name st_X, with $X = tur, flor$:

$$st_X{:}g_{st}(G_{\mathsf{st}})[\mathsf{c}_{\mathsf{st}},\ \mathsf{e}_{\mathsf{st}} \parallel \ldots],$$ where:

- G_{st} specifies the station's active properties, statically checked:

  $$G_{\mathsf{st}} = \mathsf{mc}(\varnothing, \{g_{tr}, g_{city}\})$$

  1. a station cannot go in or out of ambients (is immobile);
  2. may send out processes (i.e., passengers) to the train and to the surrounding city;

- c_{st}, e_{st} specify station passive properties, dynamically checked:
  1. $\mathsf{c}_{\mathsf{st}} = \{g^*_{tr}\}$ may be crossed by trains (which are ambients);
  2. $\mathsf{e}_{\mathsf{st}} = \{\langle g_{tr}, G_{\mathsf{arr}}\rangle^*, \langle g_{city}, G_{\mathsf{dep}}\rangle^*\}$ may receive processes (i.e., travellers) both from the city and from the train, in an unlimited number.

A public transportation system: stations

A station is an ambient of group g_{st} and of name st_X, with $X = tur, flor$:

$st_X{:}g_{st}(G_{\mathsf{st}})[\mathsf{c}_{\mathsf{st}}, \mathsf{e}_{\mathsf{st}} \parallel \ldots]$, where:

- G_{st} specifies the station's active properties, statically checked:

  $$G_{\mathsf{st}} = \mathsf{mc}(\varnothing, \{g_{tr}, g_{city}\})$$

  1. a station cannot go in or out of ambients (is immobile);
  2. may send out processes (i.e., passengers) to the train and to the surrounding city;

- $\mathsf{c}_{\mathsf{st}}, \mathsf{e}_{\mathsf{st}}$ specify station passive properties, dynamically checked:
  1. $\mathsf{c}_{\mathsf{st}} = \{g_{tr}^*\}$ may be crossed by trains (which are ambients);
  2. $\mathsf{e}_{\mathsf{st}} = \{\langle g_{tr}, G_{\mathsf{arr}}\rangle^*, \langle g_{city}, G_{\mathsf{dep}}\rangle^*\}$ may receive processes (i.e., travellers) both from the city and from the train, in an unlimited number.
     - $G_{\mathsf{arr}}, G_{\mathsf{dep}}$ are accepted behaviours for processes entering the station: the constraints $G_{\mathsf{dep}} \leq G_{\mathsf{st}}$ and $G_{\mathsf{arr}} \leq G_{\mathsf{st}}$ are statically checked.

A public transportation system: the train

A train $\text{TRAIN}_{X,Y}$ commuting between X and Y is a mobile ambient:

$\text{TRAIN} \triangleq$

$tr{:}g_{tr}(G_{tr})[\mathsf{c}_{tr}, \mathsf{e}_{tr} \| \,!\, \mathsf{out}\ st_X{:}g_{st}\,.\,\text{PATH}_{XY}\,.\,\mathsf{in}\ st_Y{:}g_{st}\,.\,\mathsf{out}\ st_Y{:}g_{st}\,.\,\text{PATH}_{YX}\,.\,\mathsf{in}\ st_X{:}g_{st}]$

where:

A public transportation system: the train

A train $\text{TRAIN}_{X,Y}$ commuting between X and Y is a mobile ambient:

$\text{TRAIN} \triangleq$

$tr{:}g_{tr}(G_{tr})[\mathsf{c}_{tr}, \mathsf{e}_{tr} \| \, !\, \mathsf{out}\ st_X{:}g_{st} \,.\, \text{PATH}_{XY} \,.\, \mathsf{in}\ st_Y{:}g_{st} \,.\, \mathsf{out}\ st_Y{:}g_{st} \,.\, \text{PATH}_{YX} \,.\, \mathsf{in}\ st_X{:}g_{st}]$

where:

- $G_{tr} = \mathsf{mc}(\{g_{st}, g_{city}, \ldots\},$ is the train's active mobility:
    1. the train may cross stations, cities, etc.;

A public transportation system: the train

A train $\mathsf{TRAIN}_{X,Y}$ commuting between X and Y is a mobile ambient:

$\mathsf{TRAIN} \triangleq$

$tr{:}g_{tr}(G_{tr})[\mathsf{c}_{tr}, \mathsf{e}_{tr} \| \,!\, \mathsf{out}\ st_X{:}g_{st}\ .\ \mathsf{PATH}_{XY}\ .\ \mathsf{in}\ st_Y{:}g_{st}\ .\ \mathsf{out}\ st_Y{:}g_{st}\ .\ \mathsf{PATH}_{YX}\ .\ \mathsf{in}\ st_X{:}g_{st}]$

where:

- $G_{tr} = \mathsf{mc}(\{g_{st}, g_{city}, \ldots\}, \{g_{st}\})$ is the train's active mobility:
  1. the train may cross stations, cities, etc.;
  2. may send out processes (i.e., passengers) only to stations

A public transportation system: the train

A train $\textsf{TRAIN}_{\mathsf{X,Y}}$ commuting between X and Y is a mobile ambient:

$\textsf{TRAIN} \triangleq$

$tr{:}g_{tr}(G_{tr})[\mathsf{c}_{tr}, \mathsf{e}_{tr} \| \, !\,\mathsf{out}\ st_X{:}g_{st}\,.\,\textsf{PATH}_{\textsf{XY}}\,.\,\mathsf{in}\ st_Y{:}g_{st}\,.\,\mathsf{out}\ st_Y{:}g_{st}\,.\,\textsf{PATH}_{\textsf{YX}}\,.\,\mathsf{in}\ st_X{:}g_{st}]$

where:

- $G_{tr} = \mathsf{mc}(\{g_{st}, g_{city}, \ldots\}, \{g_{st}\})$ is the train's active mobility:
  1. the train may cross stations, cities, etc.;
  2. may send out processes (i.e., passengers) only to stations
- $\mathsf{c}_{tr}, \mathsf{e}_{tr}$ is the train's passive mobility:

A public transportation system: the train

A train $\mathsf{TRAIN}_{\mathsf{X,Y}}$ commuting between X and Y is a mobile ambient:

$\mathsf{TRAIN} \triangleq$

$tr{:}g_{tr}(G_{\mathit{tr}})[\mathsf{c}_{\mathit{tr}}, \mathsf{e}_{\mathit{tr}} \| \,!\,\mathsf{out}\ st_X{:}g_{st}\,.\,\mathsf{PATH}_{\mathsf{XY}}\,.\,\mathsf{in}\ st_Y{:}g_{st}\,.\,\mathsf{out}\ st_Y{:}g_{st}\,.\,\mathsf{PATH}_{\mathsf{YX}}\,.\,\mathsf{in}\ st_X{:}g_{st}]$

where:

- $G_{\mathit{tr}} = \mathsf{mc}(\{g_{st}, g_{city}, \ldots\}, \{g_{st}\})$ is the train's active mobility:
  1. the train may cross stations, cities, etc.;
  2. may send out processes (i.e., passengers) only to stations
- c_{tr}, e_{tr} is the train's passive mobility:
  - $\mathsf{c}_{\mathit{tr}} = \varnothing$ the cannot be crossed by ambients

A public transportation system: the train

A train $\mathsf{TRAIN}_{X,Y}$ commuting between X and Y is a mobile ambient:

$\mathsf{TRAIN} \triangleq$

$tr{:}g_{tr}(G_{tr})[\mathsf{c}_{tr}, \mathsf{e}_{tr} \| \,!\,\mathsf{out}\ st_X{:}g_{st}\,.\,\mathsf{PATH}_{XY}\,.\,\mathsf{in}\ st_Y{:}g_{st}\,.\,\mathsf{out}\ st_Y{:}g_{st}\,.\,\mathsf{PATH}_{YX}\,.\,\mathsf{in}\ st_X{:}g_{st}]$

where:

- $G_{tr} = \mathsf{mc}(\{g_{st}, g_{city}, \ldots\}, \{g_{st}\})$ is the train's active mobility:
  1. the train may cross stations, cities, etc.;
  2. may send out processes (i.e., passengers) only to stations
- c_{tr}, e_{tr} is the train's passive mobility:
  - $\mathsf{c}_{tr} = \varnothing$ the cannot be crossed by ambients
  - $\mathsf{e}_{tr} = \{\langle g_{st}, G_{psng}\rangle^{\mathsf{n}}\}$
    may be entered by at most n processes coming from stations and exhibiting a certified good passenger behaviour G_{psng}

A public transportation system: the train

A train $\textsc{train}_{X,Y}$ commuting between X and Y is a mobile ambient:

$\textsf{TRAIN} \triangleq$

$tr{:}g_{tr}(G_{tr})[\mathsf{c}_{tr}, \mathsf{e}_{tr} \| \,!\,\mathsf{out}\ st_X{:}g_{st}\,.\,\mathsf{PATH}_{XY}\,.\,\mathsf{in}\ st_Y{:}g_{st}\,.\,\mathsf{out}\ st_Y{:}g_{st}\,.\,\mathsf{PATH}_{YX}\,.\,\mathsf{in}\ st_X{:}g_{st}]$

where:

- $G_{tr} = \mathsf{mc}(\{g_{st}, g_{city}, \ldots\}, \{g_{st}\})$ is the train's active mobility:
  1. the train may cross stations, cities, etc.;
  2. may send out processes (i.e., passengers) only to stations
- c_{tr}, e_{tr} is the train's passive mobility:
  - $\mathsf{c}_{tr} = \varnothing$ the cannot be crossed by ambients
  - $\mathsf{e}_{tr} = \{\langle g_{st}, G_{psng}\rangle^{\mathsf{n}}\}$

    may be entered by at most n processes coming from stations and exhibiting a certified good passenger behaviour G_{psng}
    - $G_{psng} = \mathsf{mc}(\varnothing, \{g_{st}\})$

      a good passenger cannot drive any ambient (no train hijacking), and may only get off the train into a station

A public transportation system: travellers

traveller $\textsf{TRAVELER}_{\mathsf{X,Y}}$ from city X to city Y: is a mobile *process* that goes into X's station where he becomes a passenger bound for Y:

$$\textsf{TRAVELER}_{\mathsf{X,Y}} \triangleq \mathsf{down}\ st_X{:}g_{st}\ \mathsf{with}\ G_{\mathit{dep}}\,.\,\textsf{PSNG}_{\mathsf{Y}}$$

$G_{\mathit{dep}} = \mathsf{mc}(\varnothing, \{g_{tr}\})$: good behaviour for departing passengers (cannot move the station, may only leave the station by getting into a train)

A public transportation system: travellers

traveller $\mathsf{TRAVELER}_{X,Y}$ from city X to city Y: is a mobile *process* that goes into X's station where he becomes a passenger bound for Y:

$$\mathsf{TRAVELER}_{X,Y} \triangleq \mathsf{down}\ st_X{:}g_{st}\ \mathsf{with}\ G_{\mathit{dep}}\,.\,\mathsf{PSNG}_Y$$

$G_{\mathit{dep}} = \mathsf{mc}(\varnothing, \{g_{tr}\})$: good behaviour for departing passengers (cannot move the station, may only leave the station by getting into a train)

$$\mathsf{PSNG}_Y \triangleq \qquad\qquad .$$

A passenger bound for Y is a process that:

A public transportation system: travellers

traveller $\mathsf{TRAVELER}_{X,Y}$ from city X to city Y: is a mobile *process* that goes into X's station where he becomes a passenger bound for Y:

$$\mathsf{TRAVELER}_{X,Y} \triangleq \mathsf{down}\ st_X{:}g_{st}\ \mathsf{with}\ G_{dep}\,.\,\mathsf{PSNG}_Y$$

$G_{dep} = \mathsf{mc}(\varnothing, \{g_{tr}\})$: good behaviour for departing passengers (cannot move the station, may only leave the station by getting into a train)

$$\mathsf{PSNG}_Y \triangleq \quad \mathsf{down}\ tr{:}g_{tr}\ \mathsf{with}\ G_{psng}\,.$$

A passenger bound for Y is a process that:

1. boards the train; G_{psng} certifies good train-passenger behaviour;

A public transportation system: travellers

traveller $\text{TRAVELER}_{X,Y}$ from city X to city Y: is a mobile *process* that goes into X's station where he becomes a passenger bound for Y:

$$\text{TRAVELER}_{X,Y} \triangleq \text{down } st_X{:}g_{st} \text{ with } G_{\mathit{dep}} \,.\, \text{PSNG}_Y$$

$G_{\mathit{dep}} = \text{mc}(\varnothing, \{g_{tr}\})$: good behaviour for departing passengers (cannot move the station, may only leave the station by getting into a train)

$$\text{PSNG}_Y \triangleq \quad \text{down } tr{:}g_{tr} \text{ with } G_{\mathit{psng}} \,.\, \text{up } st_Y{:}g_{st} \text{ with } G_{\mathit{arr}}$$

A passenger bound for Y is a process that:

1. boards the train; G_{psng} certifies good train-passenger behaviour;
2. gets off the train at the other station;
   $G_{\mathit{arr}} = \text{mc}(\varnothing, \{g_{city}\})$: certificate of good arriving-passenger behaviour (cannot move the station; may only exit into the city)

A public transportation system: travellers

traveller $\textsf{TRAVELER}_{X,Y}$ from city X to city Y: is a mobile *process* that goes into X's station where he becomes a passenger bound for Y:

$$\textsf{TRAVELER}_{X,Y} \triangleq \textsf{down}\ st_X{:}g_{st}\ \textsf{with}\ G_{\textit{dep}}\,.\,\textsf{PSNG}_Y$$

$G_{\textit{dep}} = \textsf{mc}(\varnothing, \{g_{tr}\})$: good behaviour for departing passengers (cannot move the station, may only leave the station by getting into a train)

$$\textsf{PSNG}_Y \triangleq \quad \textsf{down}\ tr{:}g_{tr}\ \textsf{with}\ G_{\textit{psng}}\,.\,\textsf{up}\ st_Y{:}g_{st}\ \textsf{with}\ G_{\textit{arr}}$$
$$.\,\textsf{add}^{\textsf{e}}\ \langle g_{st}, G_{\textit{psng}}\rangle\ \textsf{in}\ tr : g_{tr}$$

A passenger bound for Y is a process that:

1. boards the train; $G_{\textit{psng}}$ certifies good train-passenger behaviour;
2. gets off the train at the other station;
   $G_{\textit{arr}} = \textsf{mc}(\varnothing, \{g_{city}\})$: certificate of good arriving-passenger behaviour (cannot move the station; may only exit into the city)
3. frees its place, by explicitly adding one entrance permit to the train;

A public transportation system: travellers

traveller $\mathsf{TRAVELER}_{X,Y}$ from city X to city Y: is a mobile *process* that goes into X's station where he becomes a passenger bound for Y:

$$\mathsf{TRAVELER}_{X,Y} \triangleq \mathsf{down}\ st_X{:}g_{st}\ \mathsf{with}\ G_{\mathit{dep}}\,.\,\mathsf{PSNG}_Y$$

$G_{\mathit{dep}} = \mathsf{mc}(\varnothing, \{g_{tr}\})$: good behaviour for departing passengers (cannot move the station, may only leave the station by getting into a train)

$$\begin{aligned}\mathsf{PSNG}_Y \triangleq\ & \mathsf{down}\ tr{:}g_{tr}\ \mathsf{with}\ G_{\mathit{psng}}\,.\,\mathsf{up}\ st_Y{:}g_{st}\ \mathsf{with}\ G_{\mathit{arr}}\\ & .\,\mathsf{add}^{\mathsf{e}}\ \langle g_{st}, G_{\mathit{psng}}\rangle\ \mathsf{in}\ tr : g_{tr}\,.\,\mathsf{up}\ Y{:}g_{city}\ \mathsf{with}\ G_Y\,.\,\mathsf{P}\end{aligned}$$

A passenger bound for Y is a process that:

1. boards the train; G_{psng} certifies good train-passenger behaviour;
2. gets off the train at the other station;
   $G_{\mathit{arr}} = \mathsf{mc}(\varnothing, \{g_{city}\})$: certificate of good arriving-passenger behaviour (cannot move the station; may only exit into the city)
3. frees its place, by explicitly adding one entrance permit to the train;
4. goes out from the station into the city.

A public transportation system: initial configuration

The initial configuration is with the train in tur:

$$(\nu\ tur, flo, st_{tur}, st_{flo})\ world : g_w(G_w)[\mathsf{c}_w, \mathsf{e}_w \| \text{REST-OF-THE-WORLD}\ |$$
$$tur : g_{city}(G_{tur})[\mathsf{c}_{tur}, \mathsf{e}_{tur} \| R_t\ |\ \text{TRVLRS}_{tur,flo}\ |\ st_{tur} : g_{st}(G_{st})[\mathsf{c}_{st}, \mathsf{e}_{st} \| \text{TRAIN}]\,]\ |$$
$$flo : g_{city}(G_{flo})\ [\mathsf{c}_{flo}, \mathsf{e}_{flo} \| R_f\ |\ \text{TRVLRS}_{flo,tur}\ |\ st_{flo} : g_{st}(G_{st})[\mathsf{c}_{st}, \mathsf{e}_{st} \| 0\,]\,]\,]$$

where $\text{TRVLRS}_{X,Y}$ is a parallel composition of processes $\text{TRAVELER}_{X,Y}$.
Properties:

- at most n PSNG processes can be within the train at the same time, by the initial definitions of e_{tr} and $\text{TRAVELER}_{X,Y}$;
- no traveller can get into the train when this is outside a station: any such action is dynamically blocked by e_{tr};
- ...

Public transportation system: static and dynamic checking

Let FLORENTINE be a process whose behaviour is accepted in flor. Then:

Public transportation system: static and dynamic checking

Let FLORENTINE be a process whose behaviour is accepted in flor. Then:

- the process TOURIST $=$ up $\mathit{flor}{:}g_{city}$ with G_{flo} . FLORENTINE, willing to exit into flor from a (possibly mobile) nested ambient, is well typed

Public transportation system: static and dynamic checking

Let FLORENTINE be a process whose behaviour is accepted in flor. Then:

- the process TOURIST $=$ up $\mathit{flor}{:}g_{\mathit{city}}$ with G_{flo} . FLORENTINE, willing to exit into flor from a (possibly mobile) nested ambient, is well typed
- the process BADPSNG $=$ down $tr{:}g_{tr}$ with G_{bad} . TOURIST, where $G_{\mathit{bad}} = \mathsf{mc}(\varnothing, \{g_{\mathit{city}}\})$, is also well typed, since:

Public transportation system: static and dynamic checking

Let FLORENTINE be a process whose behaviour is accepted in flor. Then:

- the process TOURIST $=$ up $\mathit{flor}{:}g_{city}$ with G_{flo} . FLORENTINE, willing to exit into flor from a (possibly mobile) nested ambient, is well typed
- the process BADPSNG $=$ down $tr{:}g_{tr}$ with G_{bad} . TOURIST, where $G_{\mathit{bad}} = \mathrm{mc}(\varnothing, \{g_{city}\})$, is also well typed, since:
  - G_{bad} truthfully declares TOURIST's intention to get off into a city

Public transportation system: static and dynamic checking

Let FLORENTINE be a process whose behaviour is accepted in flor. Then:

- the process TOURIST $=$ up $\mathit{flor}{:}g_{city}$ with G_{flo} . FLORENTINE, willing to exit into flor from a (possibly mobile) nested ambient, is well typed
- the process BADPSNG $=$ down $tr{:}g_{tr}$ with G_{bad} . TOURIST, where $G_{\mathit{bad}} = \mathrm{mc}(\varnothing, \{g_{city}\})$, is also well typed, since:
  - G_{bad} truthfully declares TOURIST's intention to get off into a city
  - no global assumptions on the ambient names like tr (there might be other trains named tr where getting off is always allowed . . .)

Public transportation system: static and dynamic checking

Let FLORENTINE be a process whose behaviour is accepted in flor. Then:

- the process TOURIST $=$ up $\mathit{flor}{:}g_{\mathit{city}}$ with G_{flo} . FLORENTINE, willing to exit into flor from a (possibly mobile) nested ambient, is well typed
- the process BADPSNG $=$ down $tr{:}g_{tr}$ with G_{bad} . TOURIST, where $G_{\mathit{bad}} = \mathrm{mc}(\varnothing, \{g_{\mathit{city}}\})$, is also well typed, since:
  - G_{bad} truthfully declares TOURIST's intention to get off into a city
  - no global assumptions on the ambient names like tr (there might be other trains named tr where getting off is always allowed . . .)
- the process TOURIST cannot be statically put within the train, since G_{tr} doesn't allow processes to go directly to cities: $g_{\mathit{city}} \notin \mathscr{E}(G_{tr})$

Public transportation system: static and dynamic checking

Let FLORENTINE be a process whose behaviour is accepted in flor. Then:

- the process TOURIST $=$ up $\mathit{flor}{:}g_{city}$ with G_{flo} . FLORENTINE, willing to exit into flor from a (possibly mobile) nested ambient, is well typed
- the process BADPSNG $=$ down $tr{:}g_{tr}$ with G_{bad} . TOURIST, where $G_{\mathit{bad}} = \mathsf{mc}(\varnothing, \{g_{city}\})$, is also well typed, since:
  - G_{bad} truthfully declares TOURIST's intention to get off into a city
  - no global assumptions on the ambient names like tr (there might be other trains named tr where getting off is always allowed . . .)
- the process TOURIST cannot be statically put within the train, since G_{tr} doesn't allow processes to go directly to cities: $g_{city} \notin \mathscr{E}(G_{\mathit{tr}})$
- on the other hand, the process BADPSNG:

Public transportation system: static and dynamic checking

Let FLORENTINE be a process whose behaviour is accepted in flor. Then:

- the process TOURIST $=$ up $\mathit{flor}{:}g_{city}$ with G_{flo} . FLORENTINE, willing to exit into flor from a (possibly mobile) nested ambient, is well typed
- the process BADPSNG $=$ down $tr{:}g_{tr}$ with G_{bad} . TOURIST, where $G_{\mathit{bad}} = \mathrm{mc}(\varnothing, \{g_{city}\})$, is also well typed, since:
  - G_{bad} truthfully declares TOURIST's intention to get off into a city
  - no global assumptions on the ambient names like tr (there might be other trains named tr where getting off is always allowed . . .)
- the process TOURIST cannot be statically put within the train, since G_{tr} doesn't allow processes to go directly to cities: $g_{city} \notin \mathscr{E}(G_{\mathit{tr}})$
- on the other hand, the process BADPSNG:
  - statically can be put inside a station, since G_{st} allows processes to go to trains: $g_{tr} \in \mathscr{E}(G_{\mathit{st}})$

Public transportation system: static and dynamic checking

Let FLORENTINE be a process whose behaviour is accepted in flor. Then:

- the process TOURIST $=$ up $\mathit{flor}{:}g_{city}$ with G_{flo} . FLORENTINE, willing to exit into flor from a (possibly mobile) nested ambient, is well typed
- the process BADPSNG $=$ down $tr{:}g_{tr}$ with G_{bad} . TOURIST, where $G_{\mathit{bad}} = \mathrm{mc}(\varnothing, \{g_{city}\})$, is also well typed, since:
  - G_{bad} truthfully declares TOURIST's intention to get off into a city
  - no global assumptions on the ambient names like tr (there might be other trains named tr where getting off is always allowed . . .)
- the process TOURIST cannot be statically put within the train, since G_{tr} doesn't allow processes to go directly to cities: $g_{city} \notin \mathscr{E}(G_{\mathit{tr}})$
- on the other hand, the process BADPSNG:
  - statically can be put inside a station, since G_{st} allows processes to go to trains: $g_{tr} \in \mathscr{E}(G_{\mathit{st}})$
  - dynamically is prevented from boarding the train, since no permit $\langle g_{st}, G_{\mathit{bad}}\rangle$ is available in the train's e-component.

The global permit-granting hierarchy

Static global assumption of a partial order $\mathcal{O}$ over group names: the action

$$\mathsf{add}^{\bullet}\ \delta^{\varphi}\ \mathsf{in}\ m{:}g \text{ (with } \bullet = \mathsf{c},\mathsf{e}\text{)}$$

is statically allowed in a g'-process only if $g \leq_{\mathcal{O}} g'$.

$$\frac{\langle\!\langle g, g' \rangle\!\rangle \in \mathcal{O} \quad \mathcal{O};\Sigma \vdash \alpha : \mathsf{amb} \quad \mathcal{O};\Sigma \vdash \gamma : \mathsf{group}}{\mathcal{O};\Sigma \vdash \mathsf{add}^{\mathsf{c}}\ \gamma^{\varphi}\ \mathsf{in}\ \alpha{:}g : g'(G) \rightsquigarrow g'(G)} \quad (\textsc{Add-c})$$

$$\frac{\langle\!\langle g, g' \rangle\!\rangle \in \mathcal{O} \quad \mathcal{O};\Sigma \vdash \alpha : \mathsf{amb} \quad \mathcal{O};\Sigma \vdash \gamma : \mathsf{group}}{\mathcal{O};\Sigma \vdash \mathsf{add}^{\mathsf{e}}\ \langle \gamma, G \rangle^{\varphi}\ \mathsf{in}\ \alpha{:}g : g'(G') \rightsquigarrow g'(G')} \quad (\textsc{Add-e})$$

In the full paper:

Complete definitions of:

1. typing rules
2. (typed) reduction semantics
   observational equivalence (barbed congruence)
3. typed LTS
   bisimilarity

In the full paper:

Complete definitions of:

1. typing rules
2. (typed) reduction semantics

   observational equivalence (barbed congruence)
3. typed LTS

   bisimilarity

Results:

- subject reduction
- soundness of bisimilarity w.r.t. the barbed congruence

Conclusions

A typed ambient/process calculus with:

- interplay between static and dynamic type-checking, and between *active* and *passive* rights, for handling the security requirements of global computing applications:
  - static type-checking controls (communication and) *active* mobility rights;
  - dynamic type-checking controls *passive* rights;
- packing of a type within a mobile process and its check at destination as a (very) abstract modelling of the proof-carrying code approach;
- purely local static type checking, except for the global $\mathcal{O}$ hierarchy.

Some unsatisfactory aspects (future work?):

- authorization to add permits is too coarse-grain (either no adding, or adding with any multiplicity)
- absence of group restriction, useful for protection from external untrusted agents;
- lack of expressive synchronizing mechanisms (only communication), making awkward to control unwanted nondeterminism;
- ...