A l'attention du lecteur: cette présentation n'a pas pour but d'être lue sans se référer aux articles sur les types pour DREAM disponibles sur la page `http://sardes.inrialpes.fr/kells/`.

Alan Schmitt

DreamTypes

Systèmes de Types pour Dream - ICAR 2006

Alan Schmitt, alan.schmitt@polytechnique.org

INRIA Rhône-Alpes

1er septembre 2006

Formal methods will never have a significant impact until they can be used by people that don't understand them.

T. Melham

DREAM

Today, most software exists, not to solve a problem, but to interface with other software.

I. O. Angell

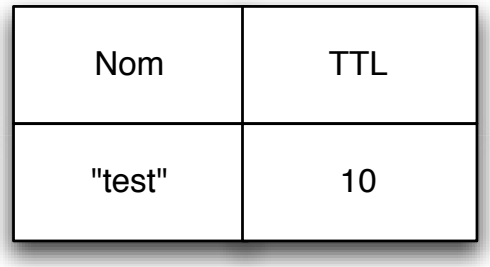
Dream is a component-based framework dedicated to the construction of communication middleware. It provides a component library and a set of tools to build, configure and deploy middleware implementing various communication paradigms: group communications, message passing, event-reaction, publish-subscribe, etc. Dream builds upon the Fractal component framework, which provides support for hierarchical and dynamic composition.

http://dream.objectweb.org/

Manipulation de messages

Intergiciel de communication: programme manipulant des messages

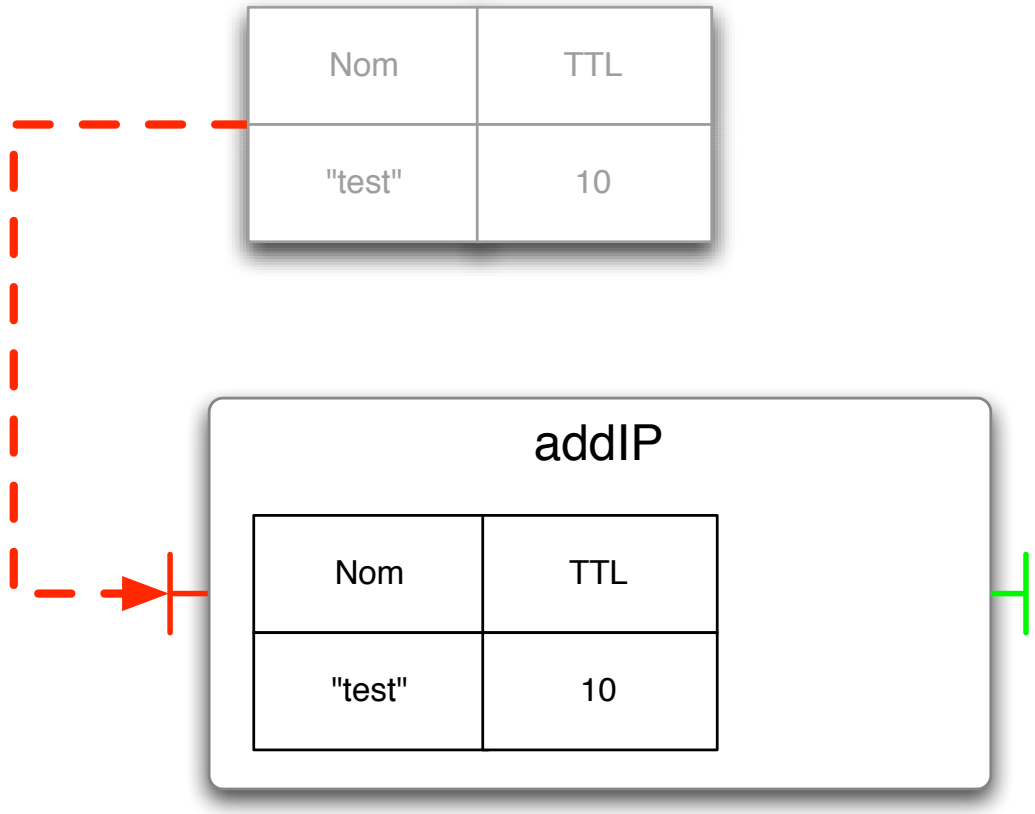
Message: ensemble de blocs nommés



Manipulation de messages

Intergiciel de communication: programme manipulant des messages

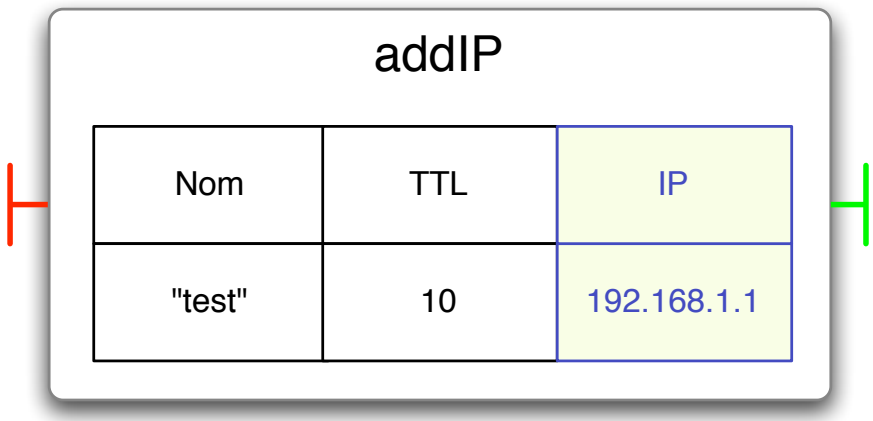
Message reçu par un composant sur une interface serveur



Manipulation de messages

Intergiciel de communication: programme manipulant des messages

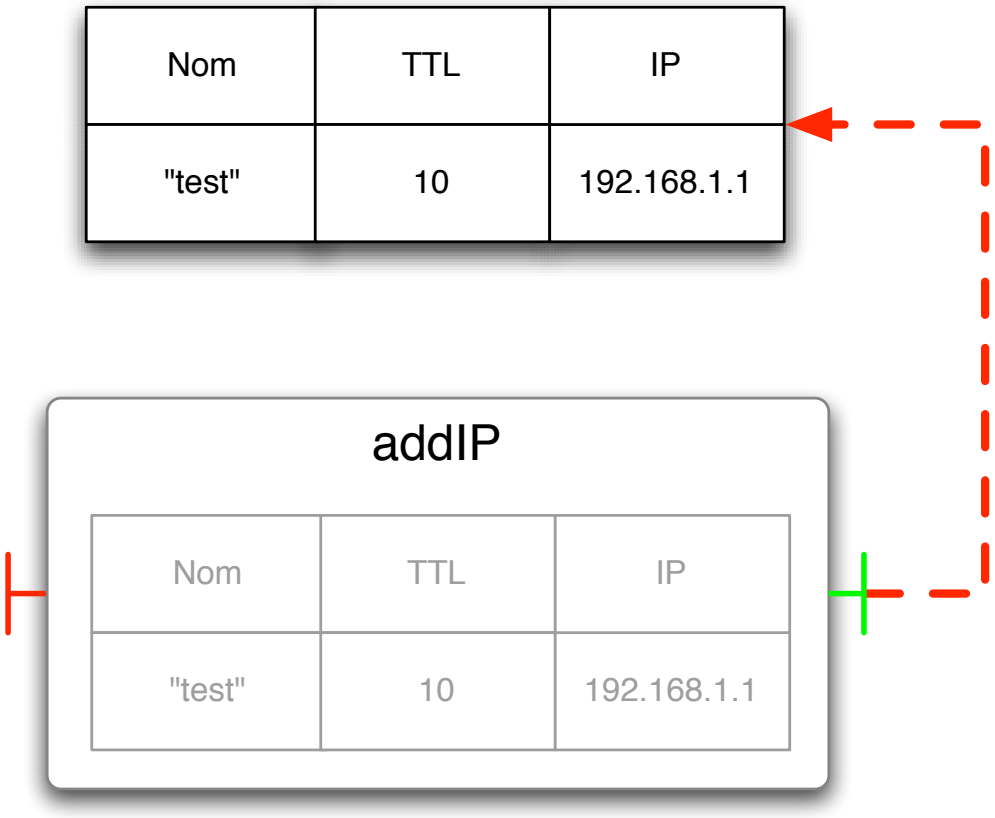
Blocs du message supprimés ou ajoutés



Manipulation de messages

Intergiciel de communication: programme manipulant des messages

Un ou plusieurs messages émis sur une interface client



Opérations sur les messages

Accès à un bloc: définit si le bloc est présent (Nom)

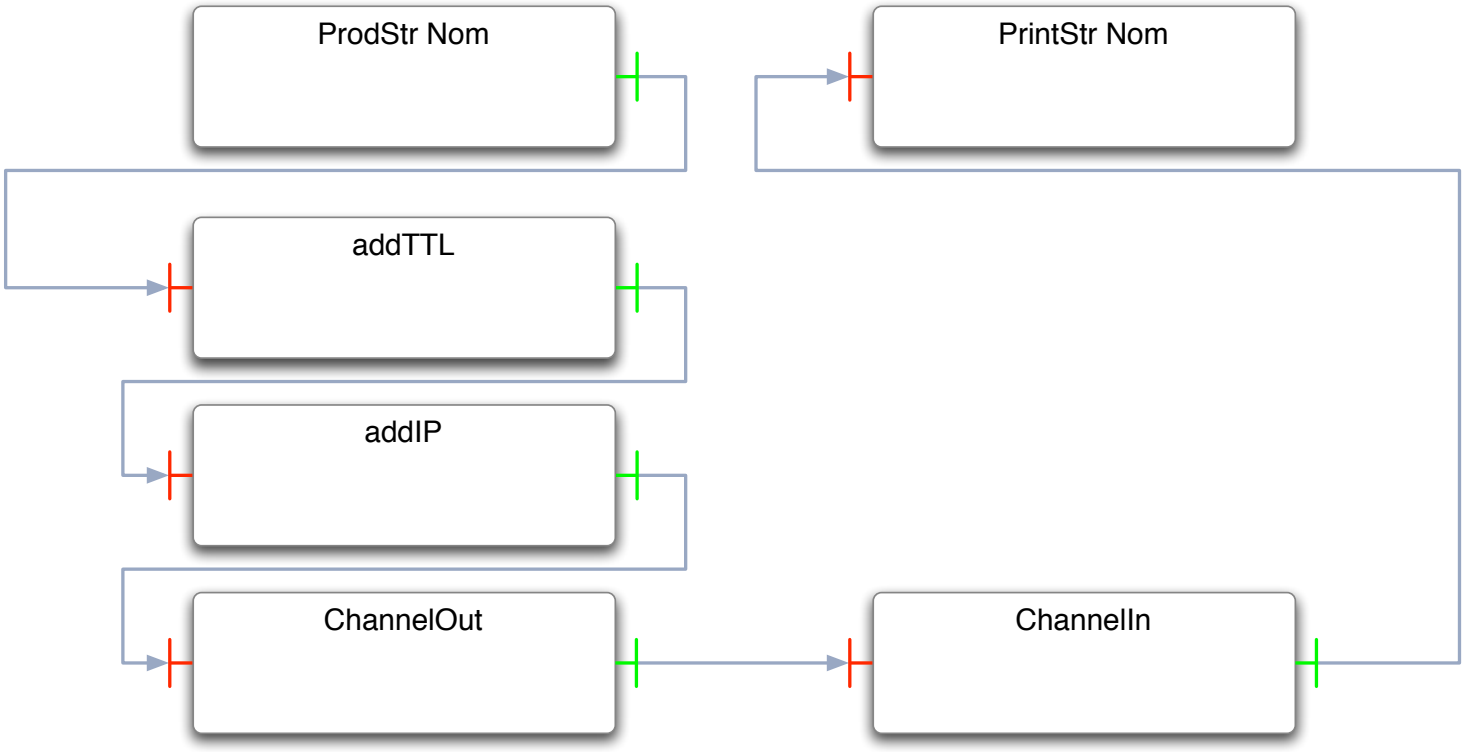
Suppression d'un bloc: définit si le bloc est présent (TTL)

Ajout d'un bloc: définit si le bloc est absent (IP)

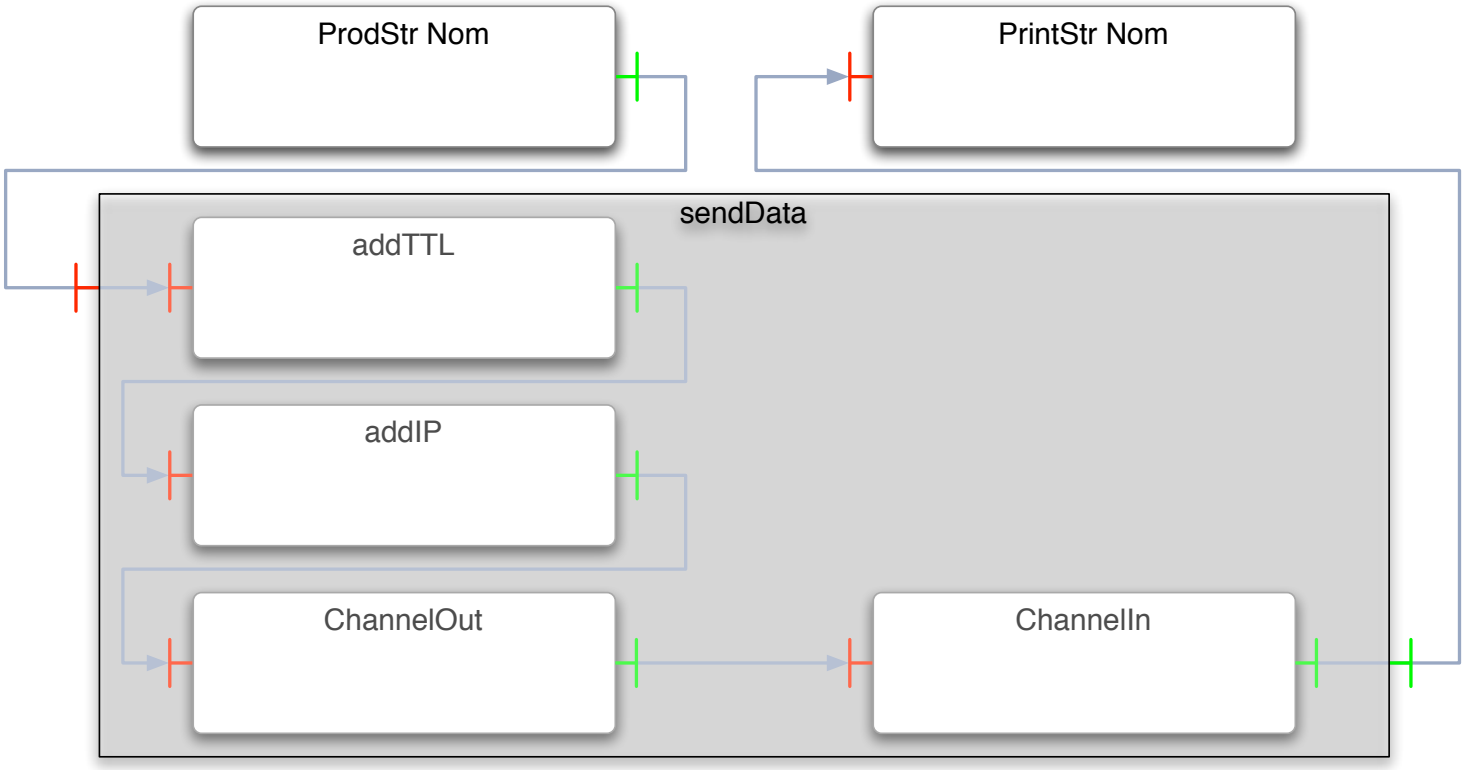
Nom	TTL
"test"	10

Dans le cas contraire erreur à l'exécution

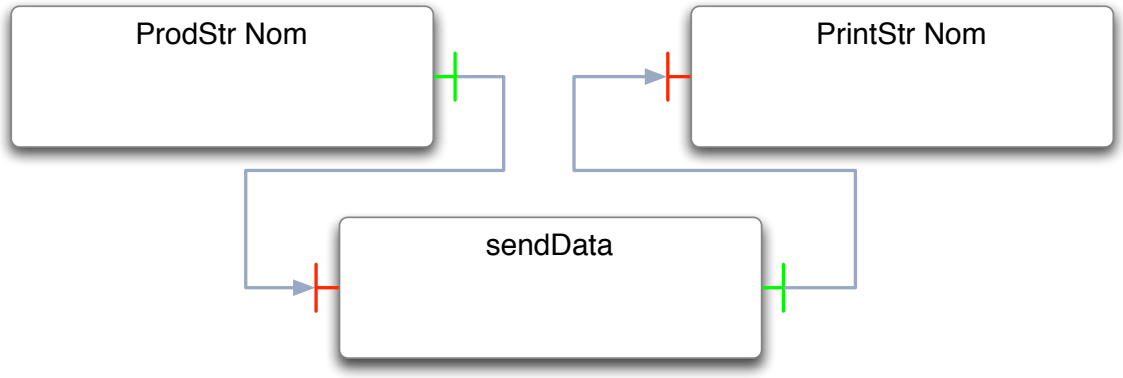
Envoi d'un nom



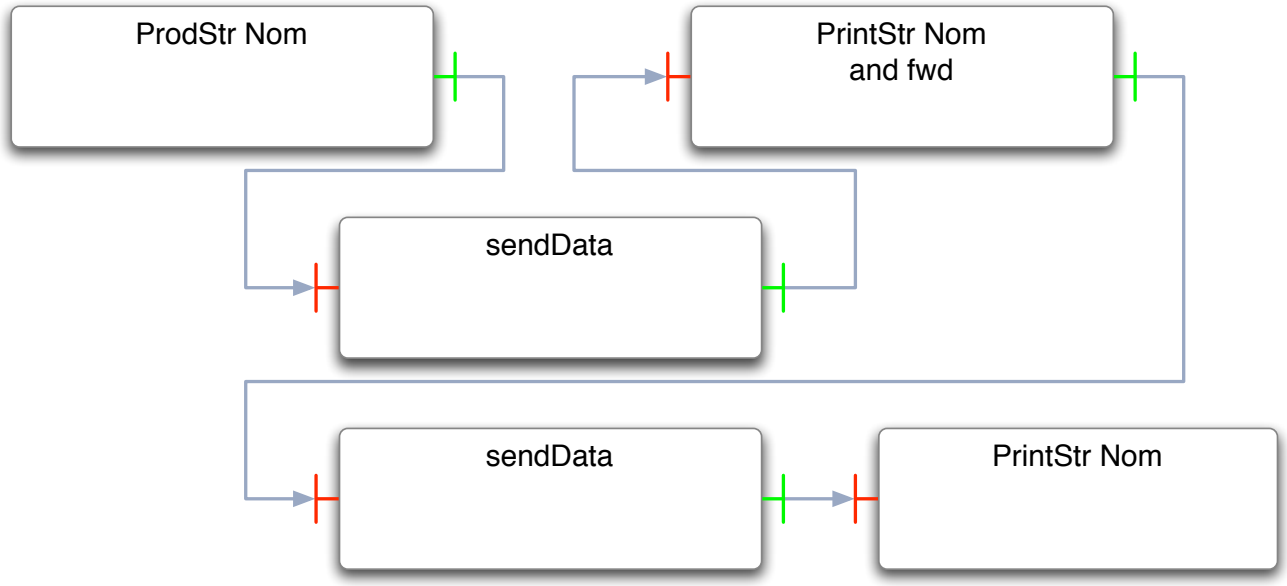
Envoi d'un nom



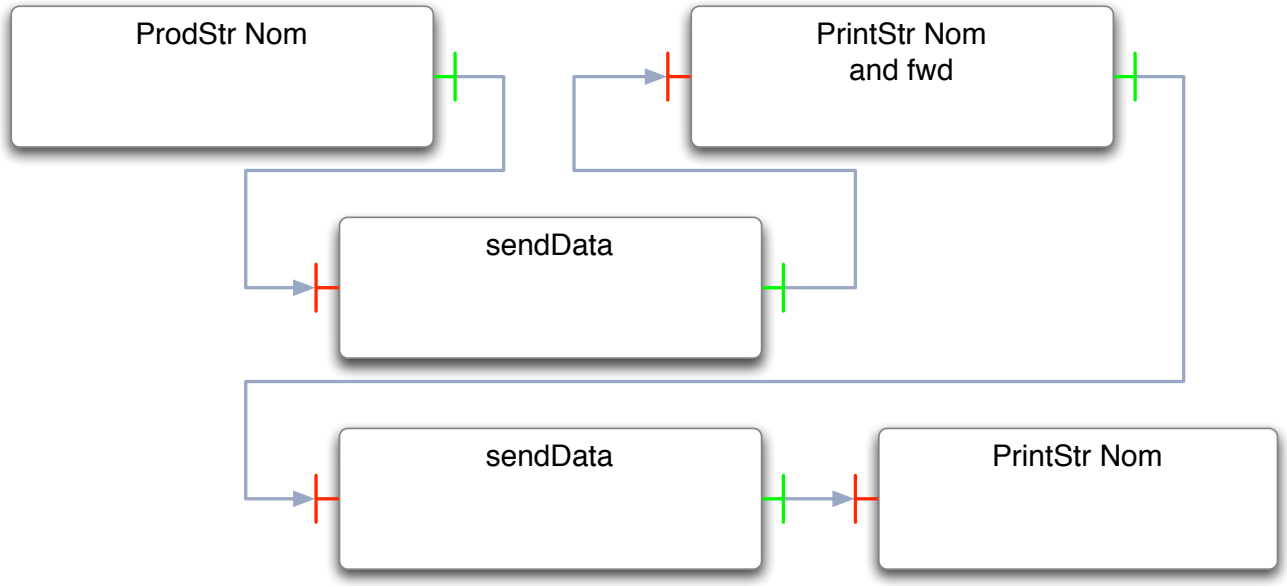
Envoi d'un nom



Envoi d'un nom

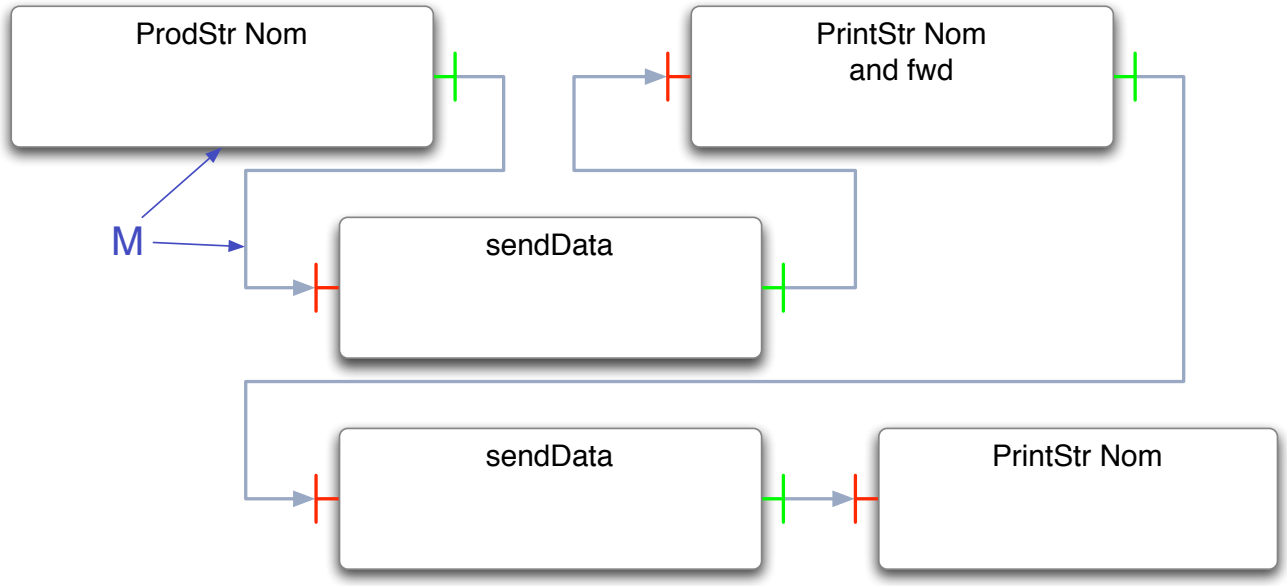


Envoi d'un nom

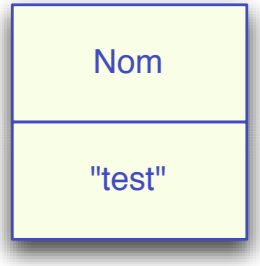


Correct ?

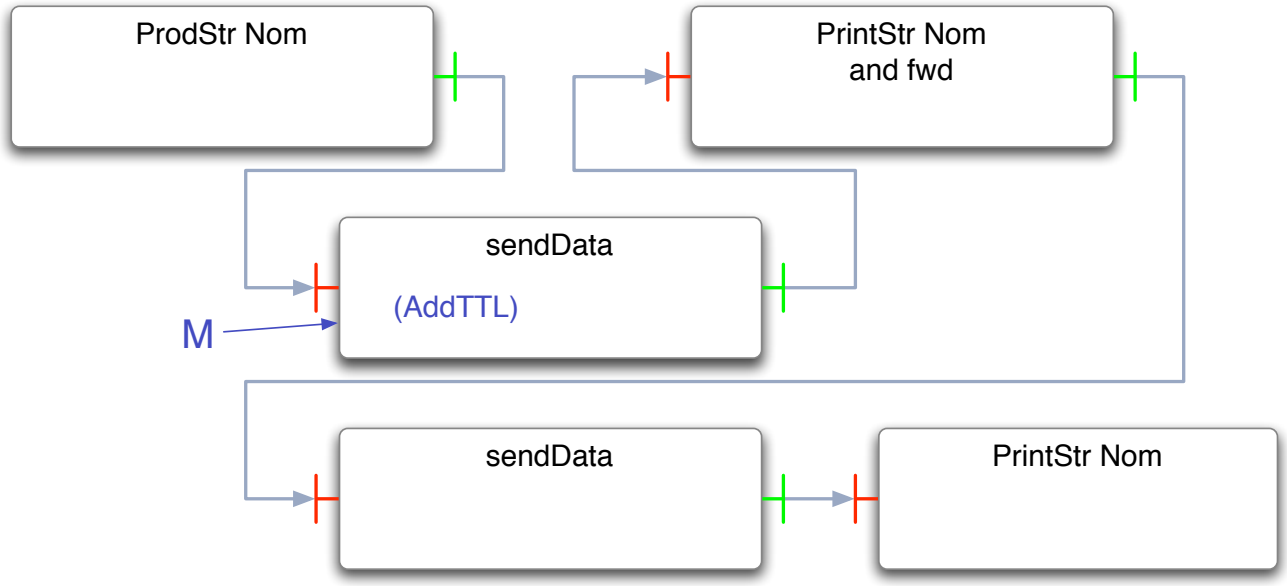
Envoi d'un nom



Correct ? Suivons un message ...



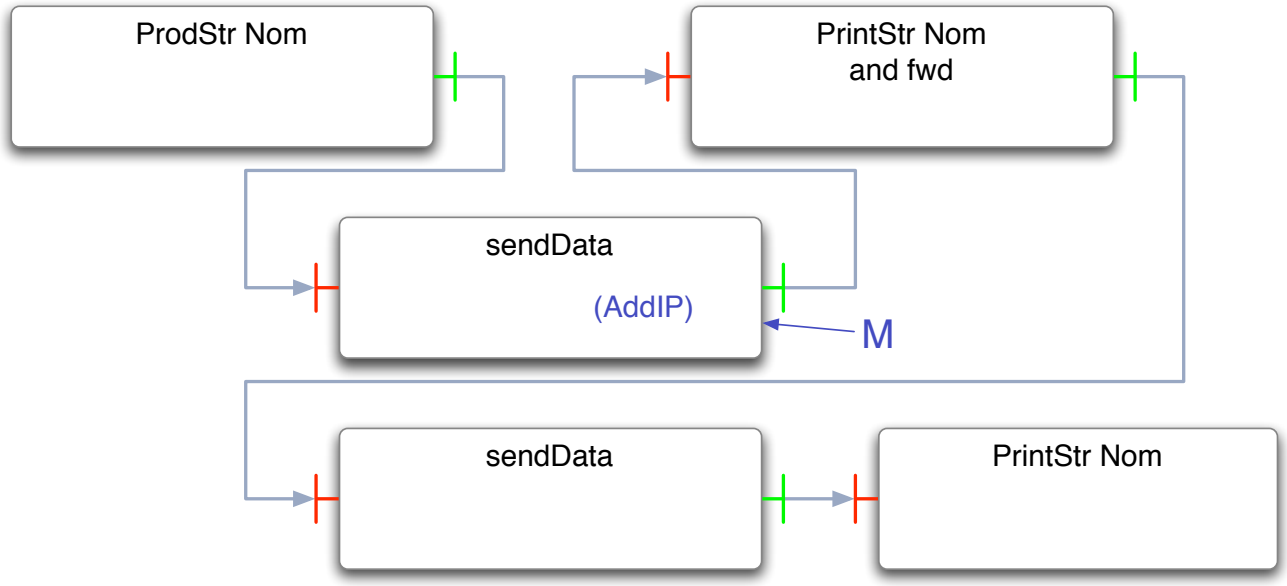
Envoi d'un nom



Correct ? Suivons un message ...

Nom	TTL
"test"	10

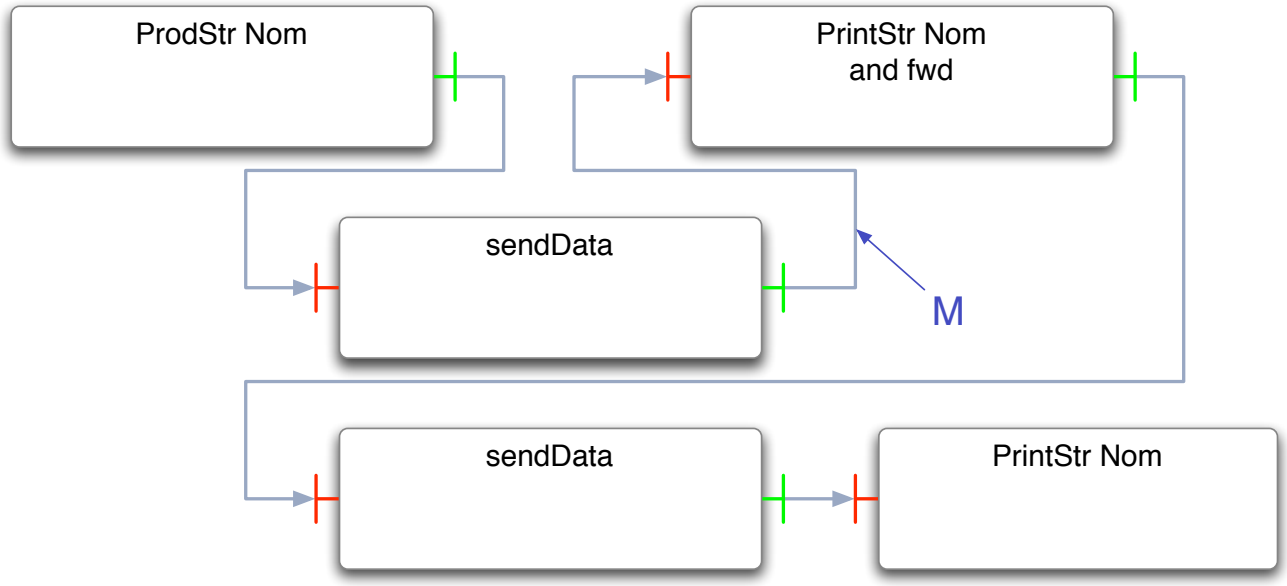
Envoi d'un nom



Correct ? Suivons un message ...

Nom	TTL	IP
"test"	10	192.168.1.1

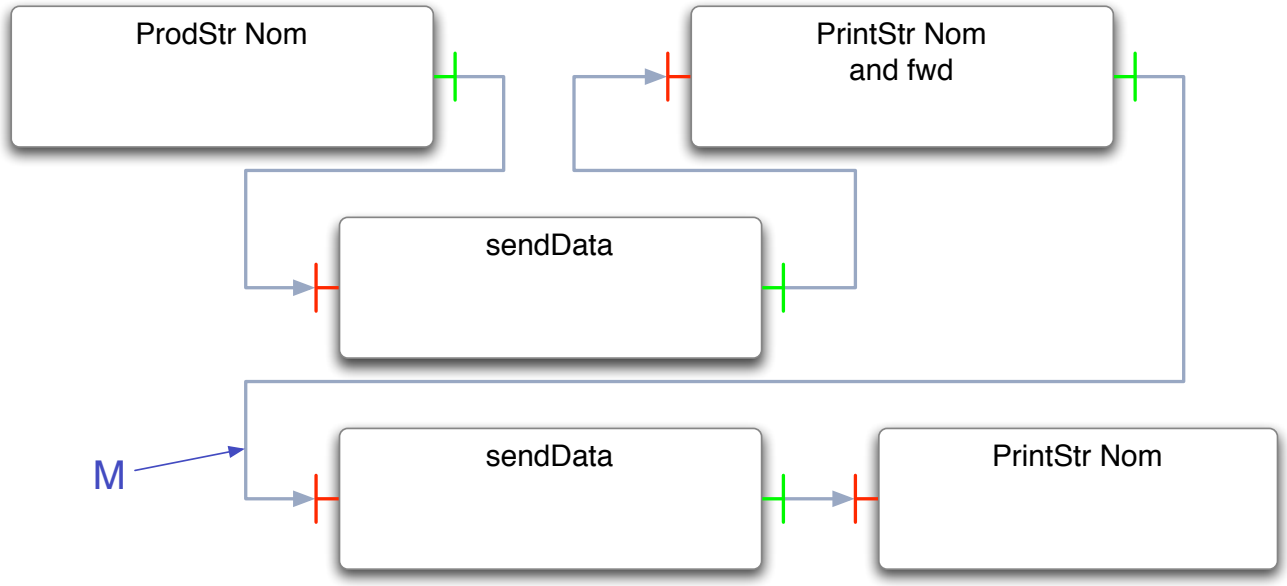
Envoi d'un nom



Correct ? Suivons un message ...

Nom	TTL	IP
"test"	10	192.168.1.1

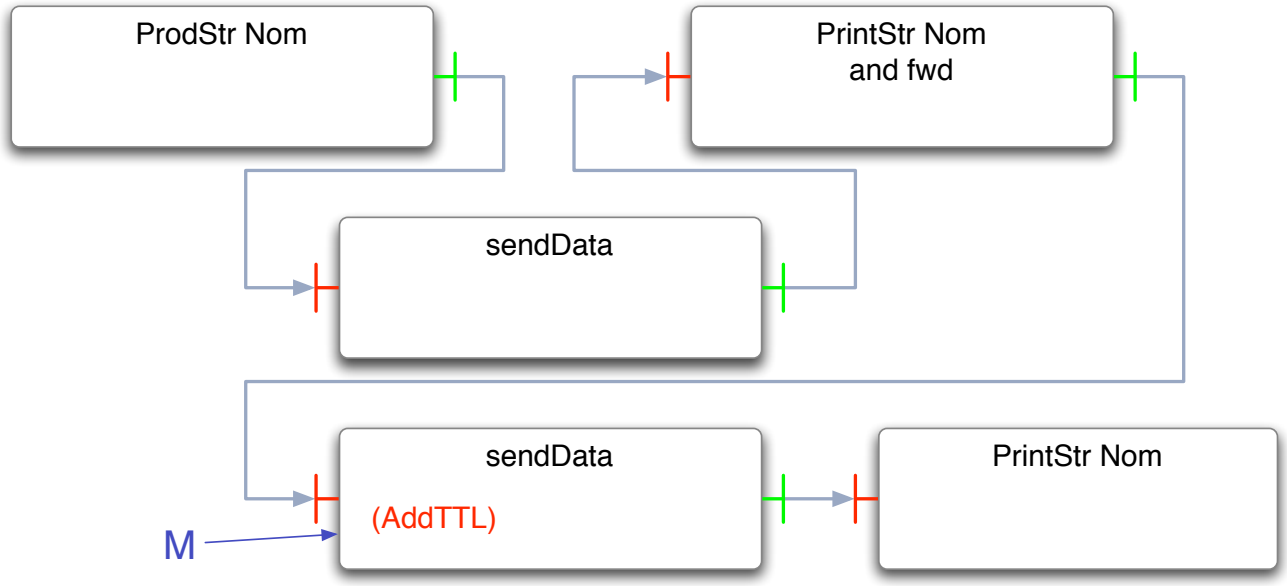
Envoi d'un nom



Correct ? Suivons un message ...

Nom	TTL	IP
"test"	10	192.168.1.1

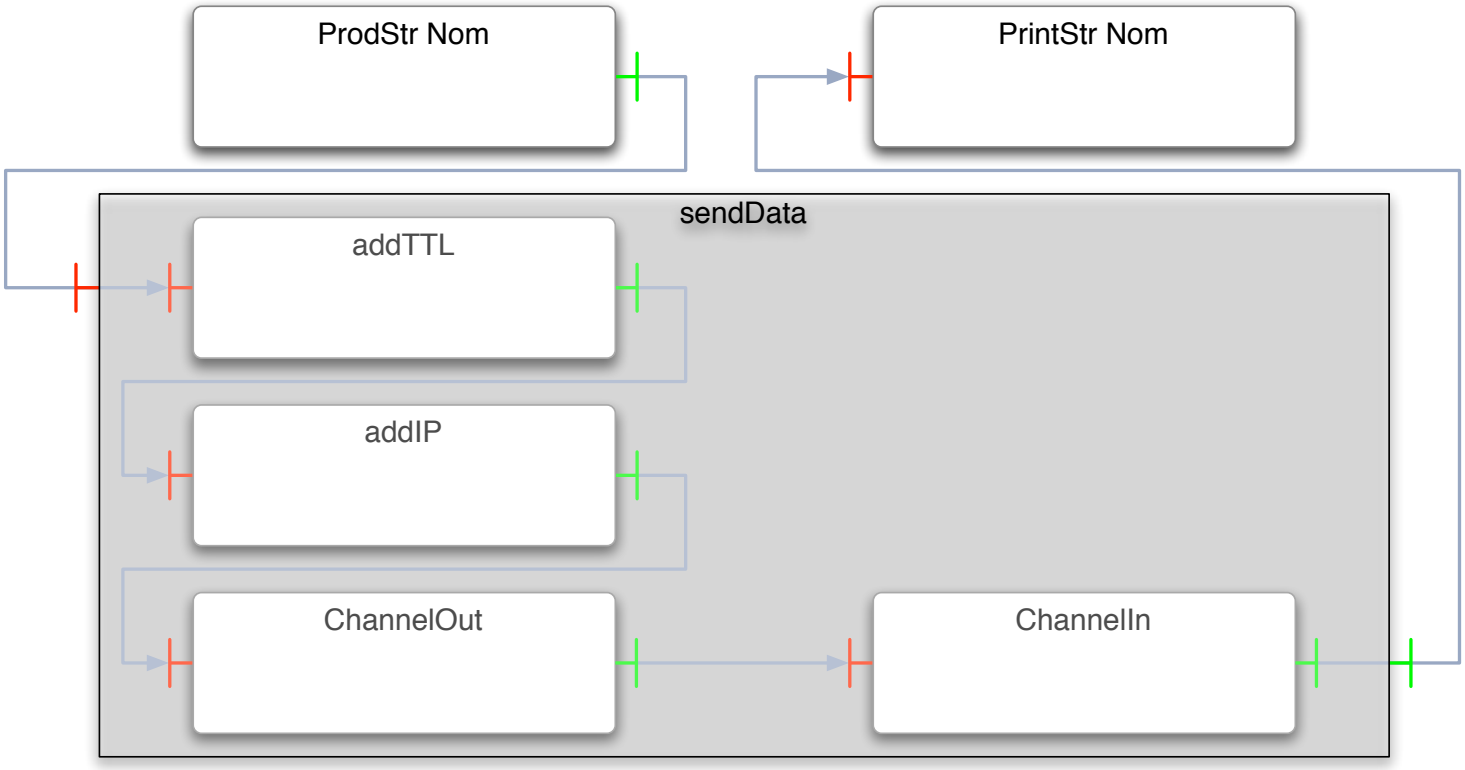
Envoi d'un nom



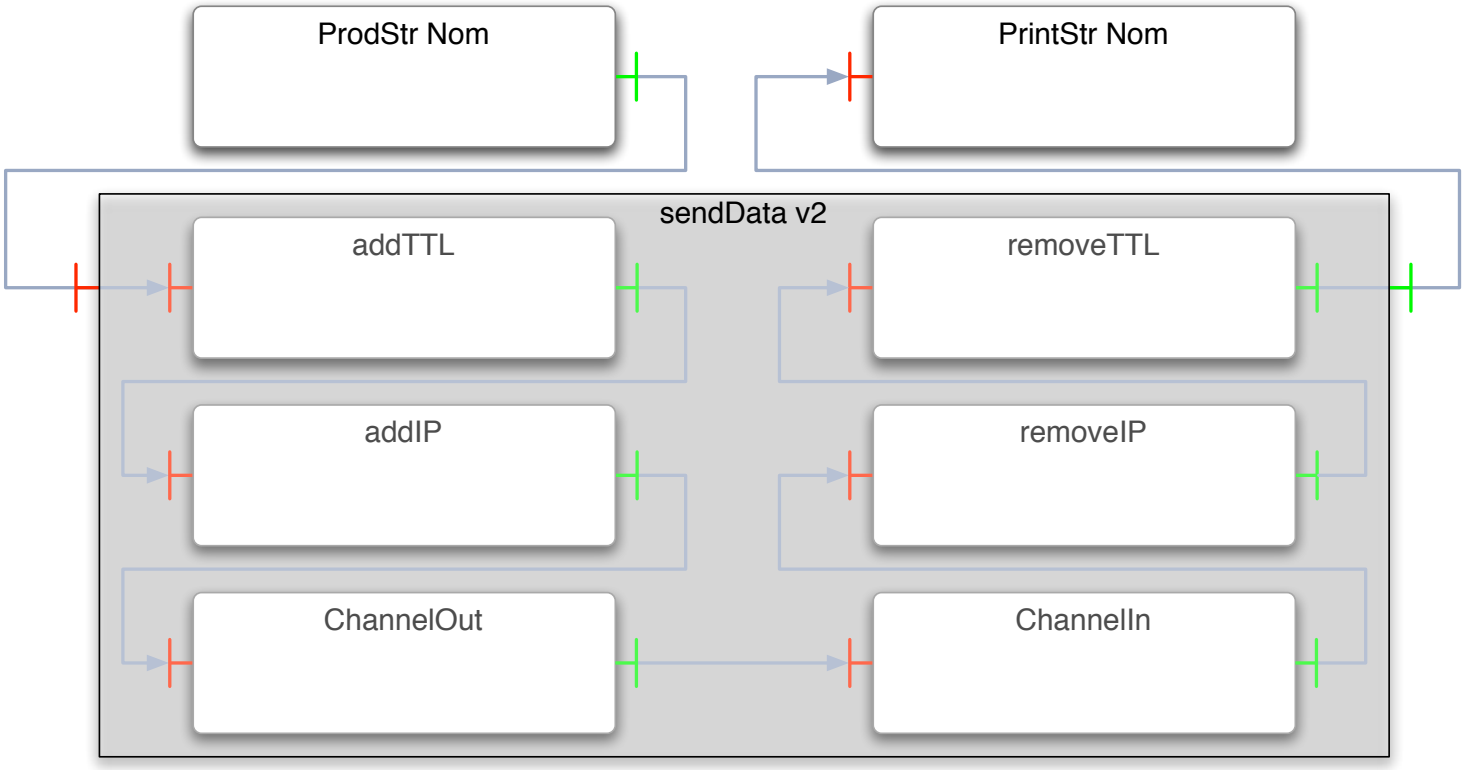
Correct ? Suivons un message ... Non !

Nom	TTL	IP
"test"	10	192.168.1.1

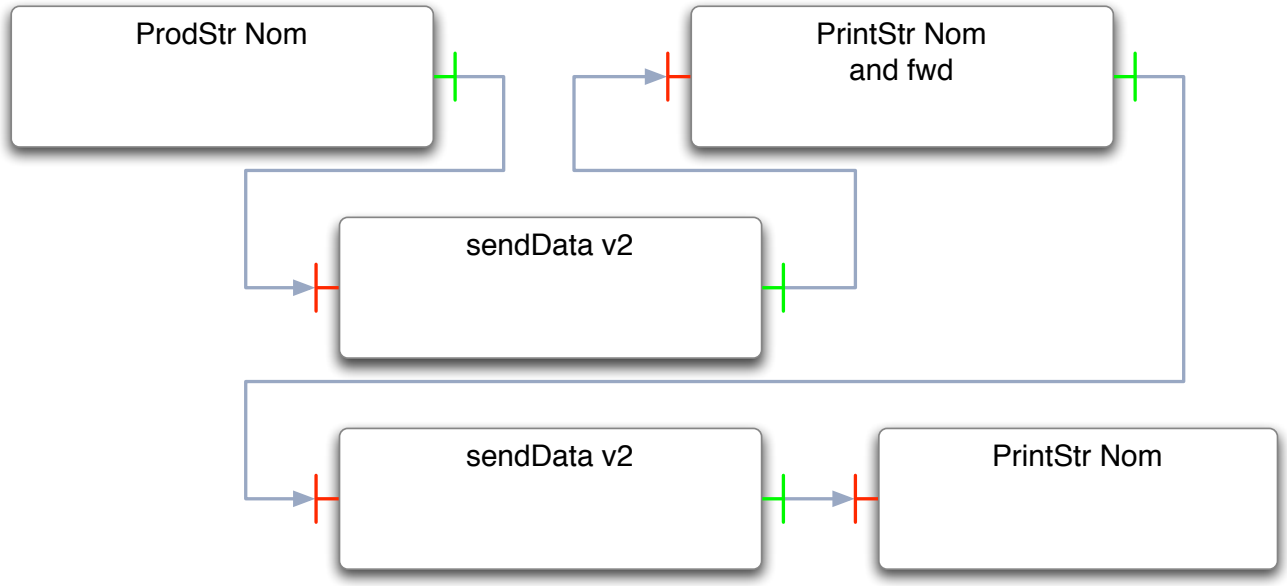
Envoi d'un nom



Envoi d'un nom



Envoi d'un nom



Correct !

DreamTypes

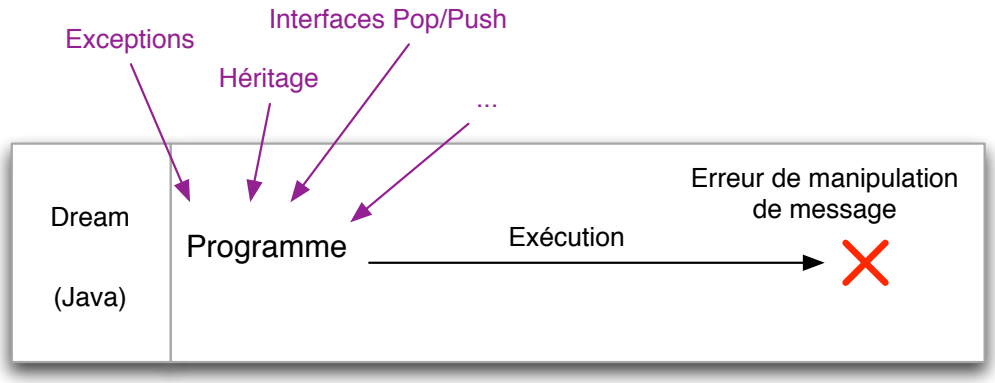
Objectif: vérification statique de l'absence de "mauvaise" opération sur les messages

Propriétés d'un Système de Types

1. un programme typable se réduit en un programme typage (c'est un invariant)
2. un programme typable n'est pas dans une configuration d'erreur

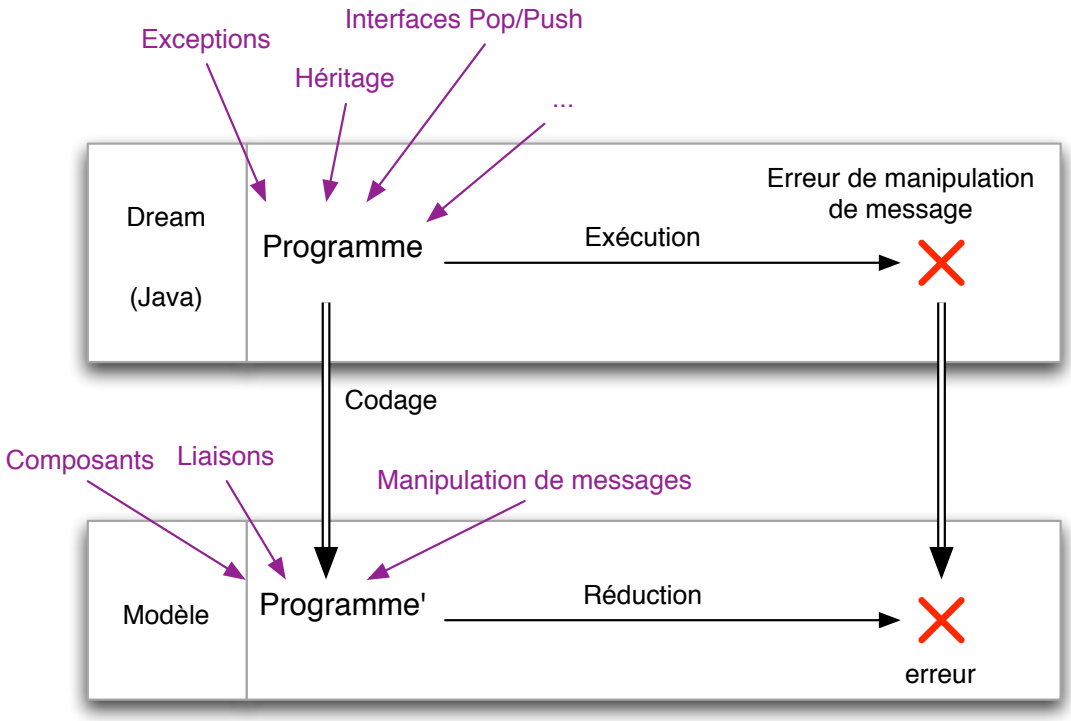
Système de types: approximation du comportement d'un programme

Notre approche



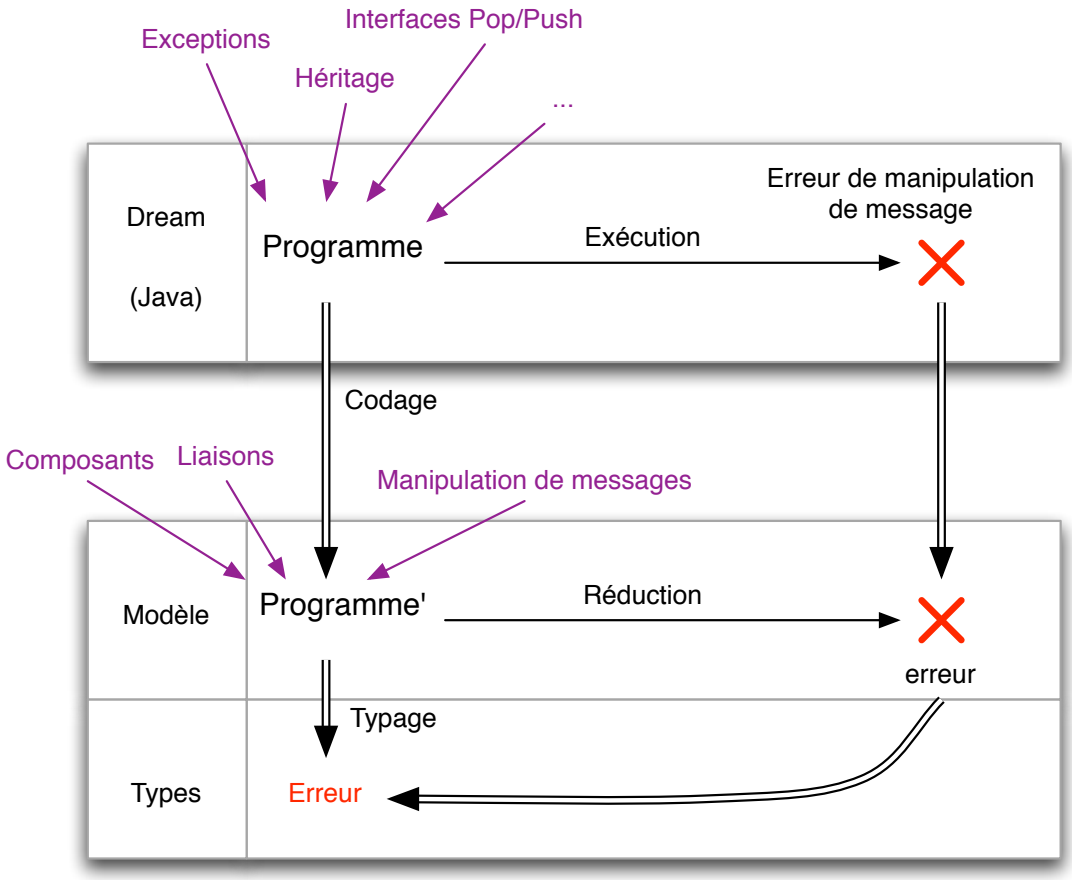
Un programme effectuant une opération incorrecte sur les messages

Notre approche



Un programme effectuant une opération incorrecte sur les messages est encodé en un programme simplifié effectuant une erreur

Notre approche

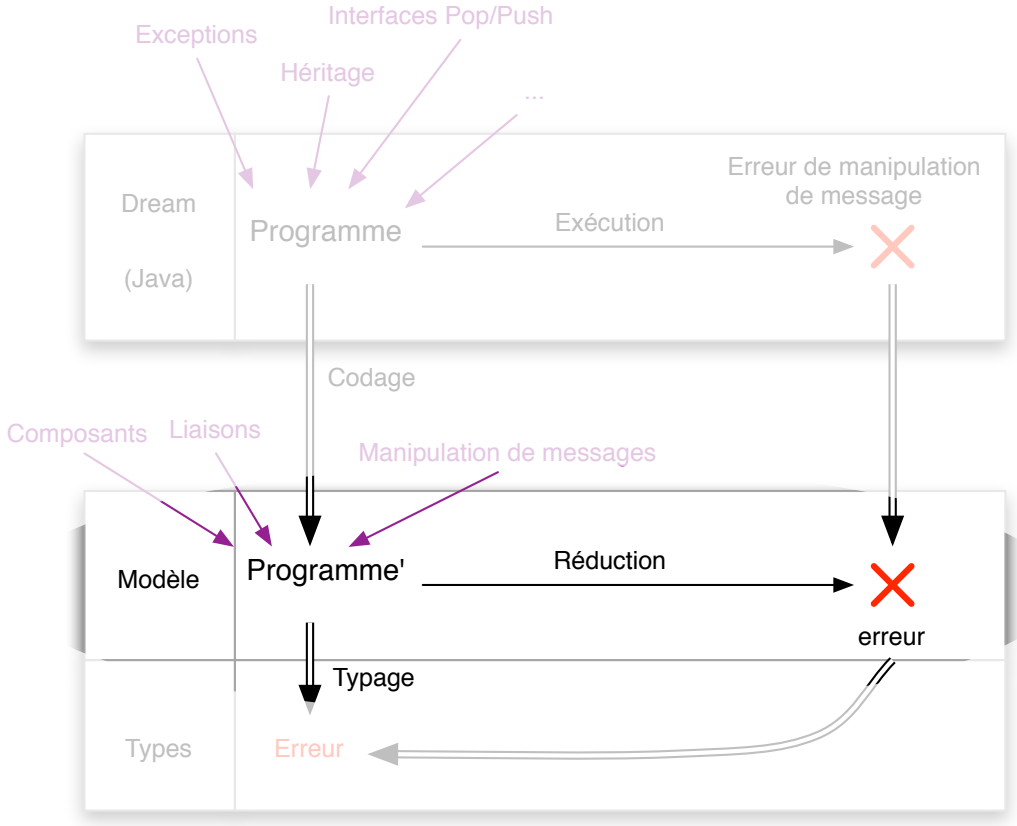


Un programme effectuant une opération incorrecte sur les messages est encodé en un programme simplifié effectuant une erreur qui est détectée statiquement.

Et le système de types de Java ?

```
Message addChunk (Message message,
                  String chunkName,
                  AbstractChunk chunk);
AbstractChunk getChunk(Message message,
                  String chunkName);
AbstractChunk removeChunk(Message message,
                  String chunkName);
```

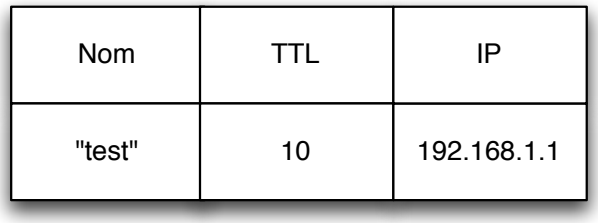
Le type `Message` est opaque: pas d'information sur les blocs présents



All models are wrong; some models are useful.

G. Box

Messages et composants

Message		$\left\{ \begin{array}{l} \text{Nom} = \text{"test"}; \\ \text{TTL} = 10; \\ \text{IP} = 192.168.1.1 \end{array} \right\}$
Composant		addIP[*Contenu*]
Liaison		$s(x).x@e$

$s(x).x@e$ prend un message x sur l'interface s et l'envoie sur l'interface e

Opérations sur les messages

Contenu d'un composant

Un message est reçu sur l'interface e,

Opérations sur les messages

Contenu d'un composant

Un message est reçu sur l'interface e, des calculs sur ce message sont effectués

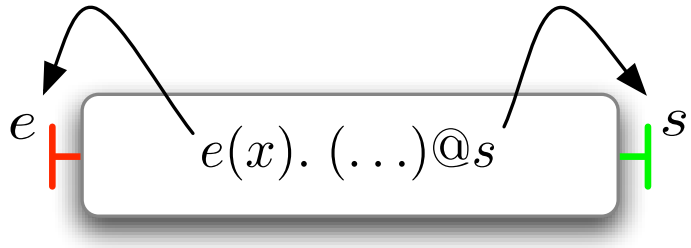
Opérations sur les messages

Contenu d'un composant

Un message est reçu sur l'interface e, des calculs sur ce message sont effectués et un message est envoyé sur l'interface s.

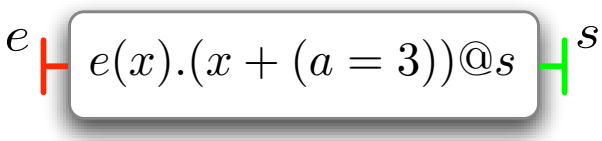
Opérations sur les messages

Contenu d'un composant



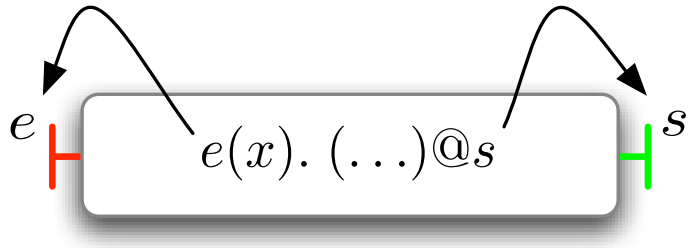
Un message est reçu sur l'interface *e*, des calculs sur ce message sont effectués et un message est envoyé sur l'interface *s*.

Ajout d'un bloc "a"



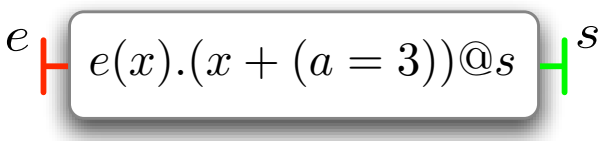
Opérations sur les messages

Contenu d'un composant

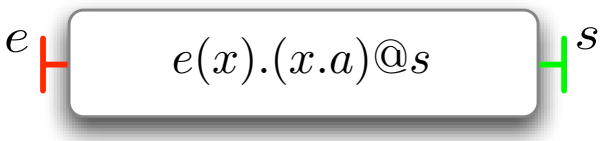


Un message est reçu sur l'interface *e*, des calculs sur ce message sont effectués et un message est envoyé sur l'interface *s*.

Ajout d'un bloc "a"

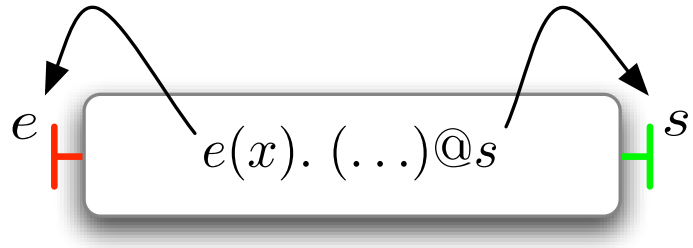


Accès au bloc "a"



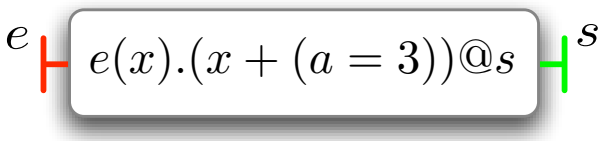
Opérations sur les messages

Contenu d'un composant

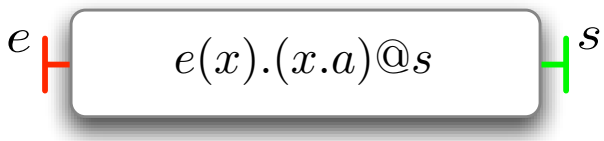


Un message est reçu sur l'interface *e*, des calculs sur ce message sont effectués et un message est envoyé sur l'interface *s*.

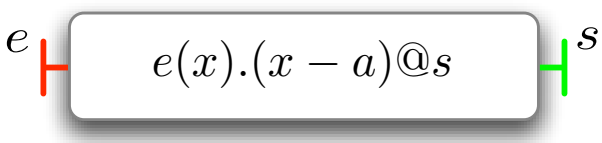
Ajout d'un bloc "a"



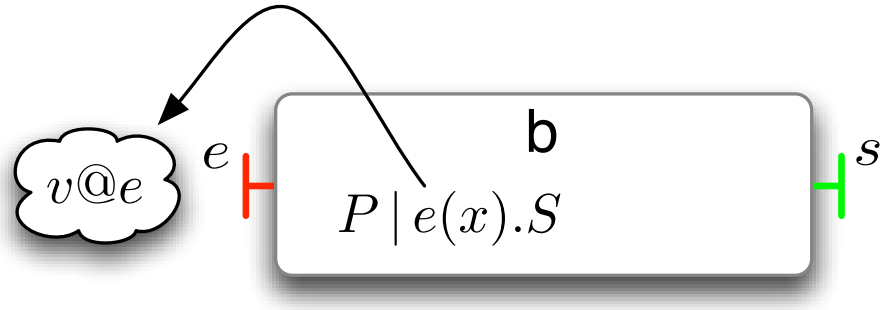
Accès au bloc "a"



Suppression d'un bloc "a"

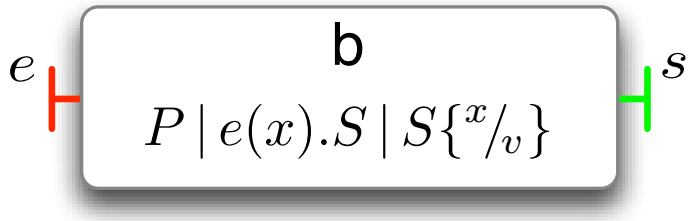


La réduction, formellement



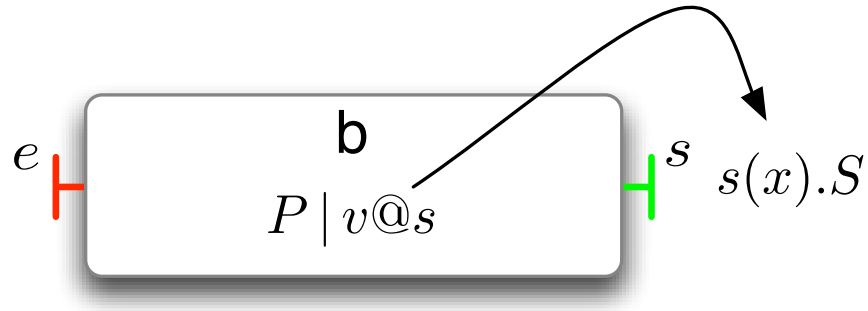
$$b[P \mid e(x).S] \mid v@e \longrightarrow b[P \mid e(x).S \mid S\{^{v}/_{x}\}]$$

La réduction, formellement



$$b[P \mid e(x).S] \mid v@e \longrightarrow b[P \mid e(x).S \mid S\{^v/_x\}]$$

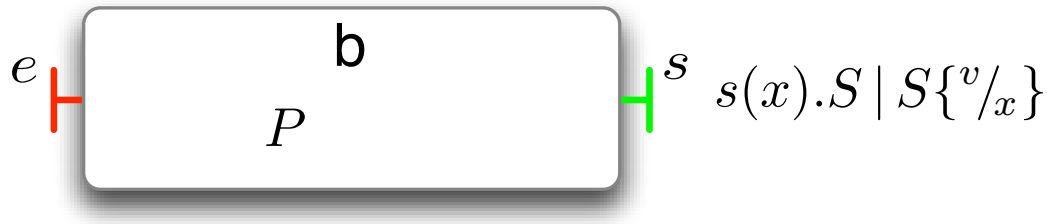
La réduction, formellement



$$b[P \mid e(x).S] \mid v@e \longrightarrow b[P \mid e(x).S \mid S\{^{v}/_{x}\}]$$

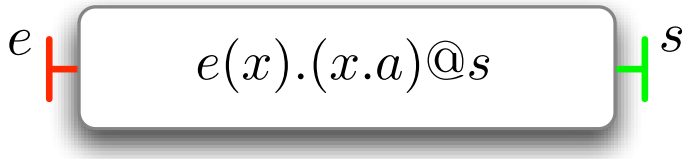
$$b[P \mid v@s] \mid s(x).S \longrightarrow b[P] \mid S\{^{v}/_{x}\}$$

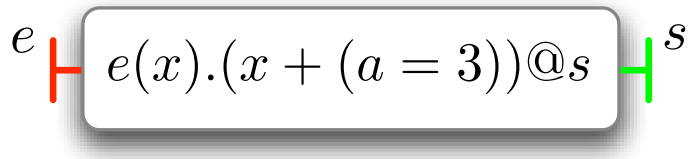
La réduction, formellement

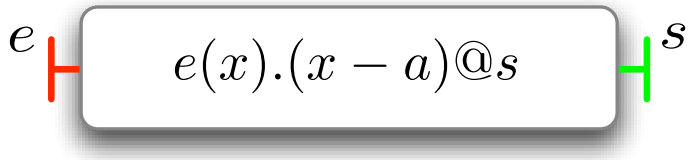


$$b[P \mid e(x).S] \mid v@e \longrightarrow b[P \mid e(x).S \mid S\{^{v}/_{x}\}]$$
$$b[P \mid v@s] \mid s(x).S \longrightarrow b[P] \mid S\{^{v}/_{x}\}$$

La réduction, formellement





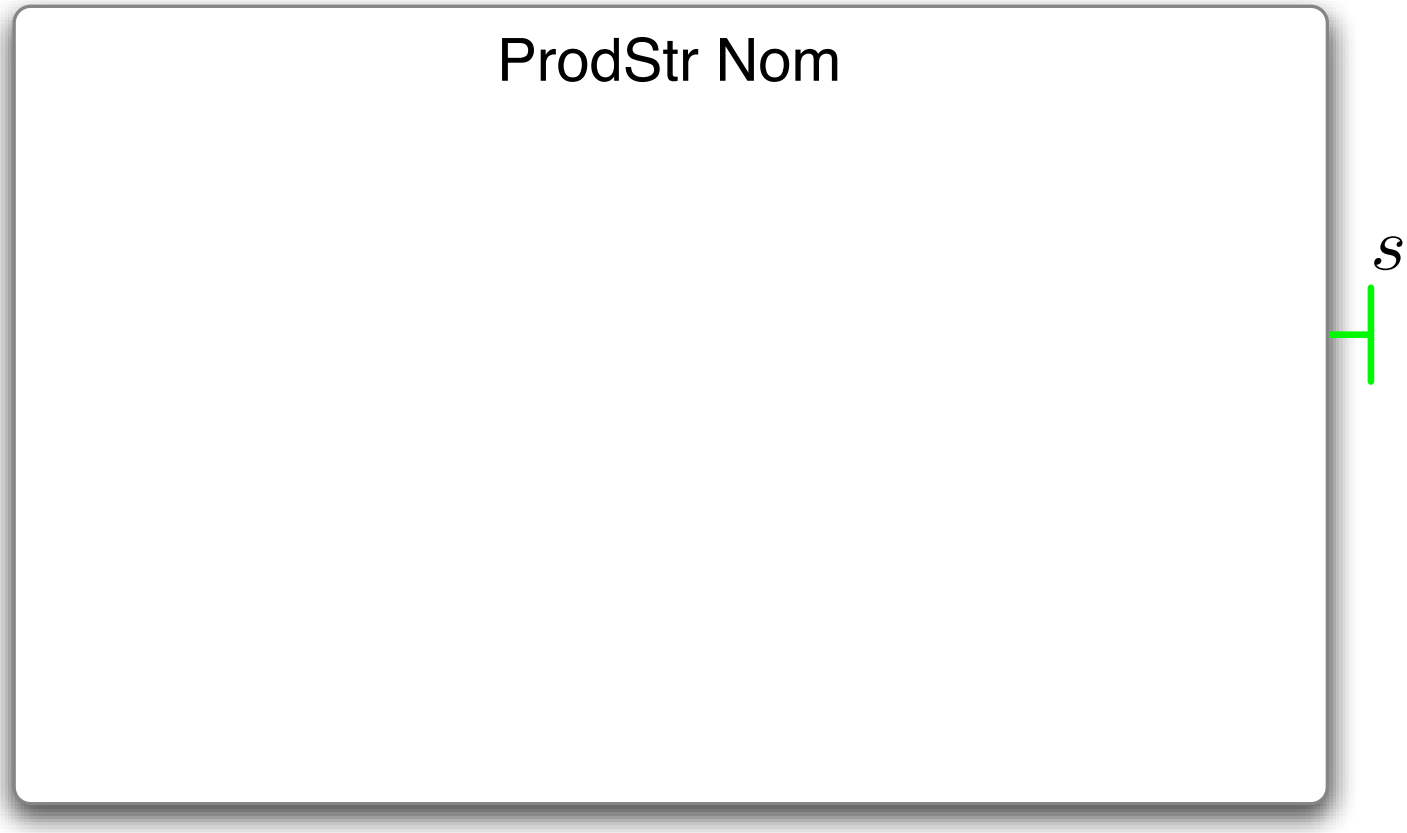


$$\{a_1 = v_1; \ldots; a_i = v_i; \ldots; a_n = v_n\}.a_i \longrightarrow v_i$$

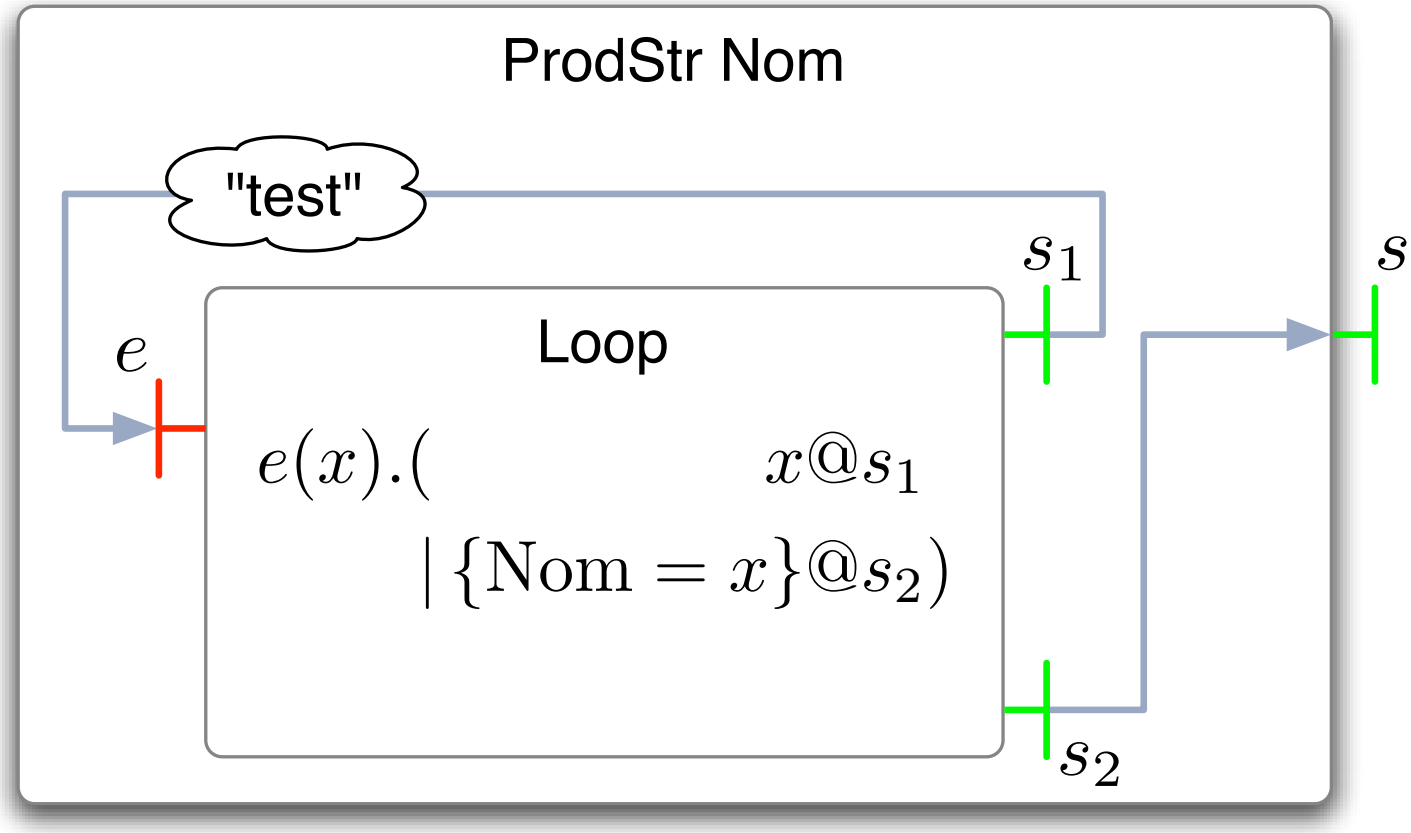
$$\{a_1 = v_1; \ldots; a_{i-1} = v_{i-1}; a_i = v_i; a_{i+1} = v_{i+1}; \ldots; a_n = v_n\} - a_i$$
$$\longrightarrow \{a_1 = v_1; \ldots; a_{i-1} = v_{i-1}; a_{i+1} = v_{i+1}; \ldots; a_n = v_n\}$$

$$\{a_1 = v_1; \ldots; a_n = v_n\} + (a = v)$$
$$\longrightarrow \{a_1 = v_1; \ldots; a_n = v_n; a = v\} \qquad \text{si } a \notin \{a_1, \ldots, a_n\}$$

Production par des boucles

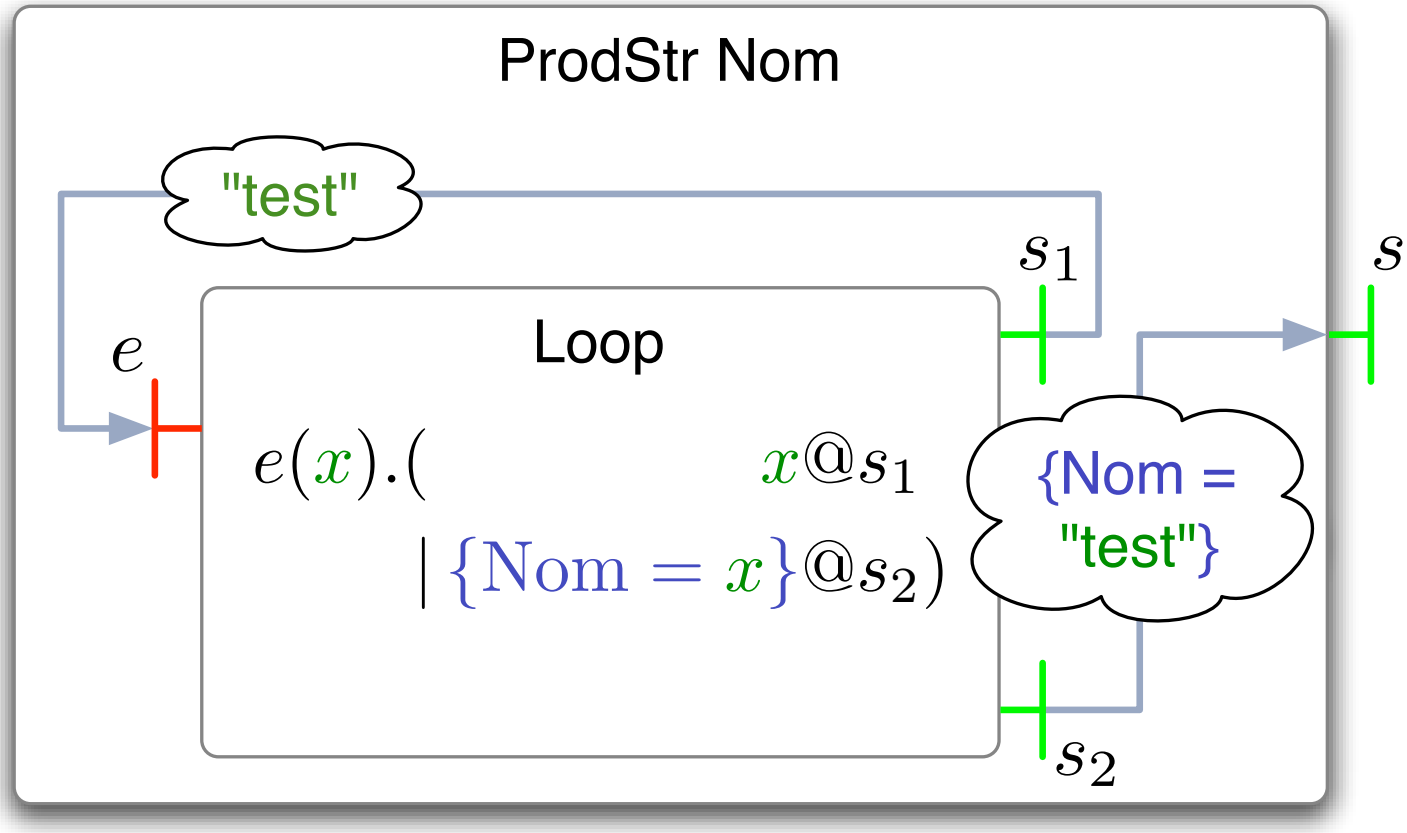


Production par des boucles



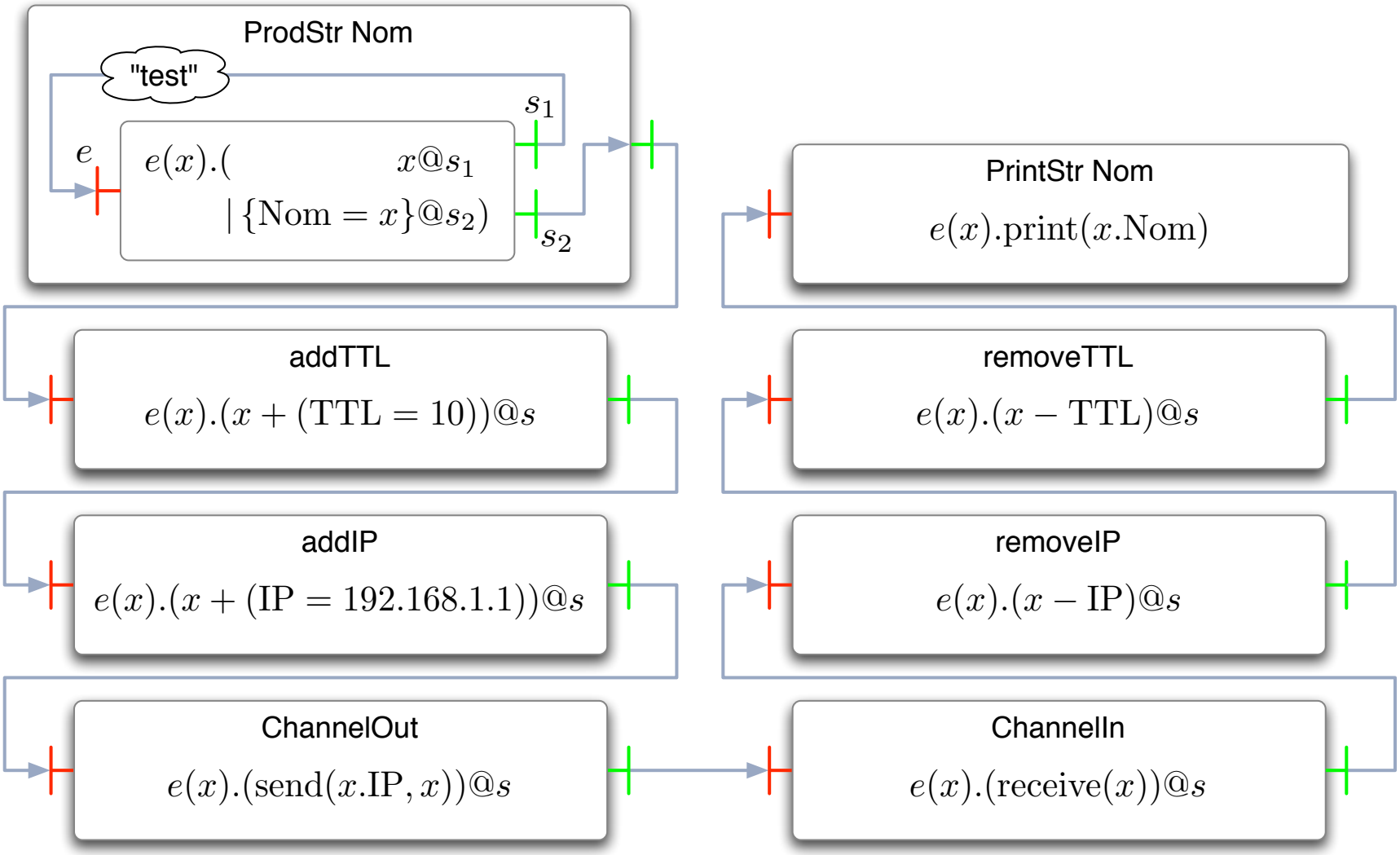
$$\text{ProdStr}\left[\text{Loop}\left[\begin{array}{l} e(x).(x@s_1 \\ \quad | \{\text{Nom}{=}\text{x}\}\,@s_2) \end{array}\right]\;\begin{array}{l} |\,\text{"test"}\,@e \\ |\,s_1(x).x@e \\ |\,s_2(x).x@s \end{array}\right]$$

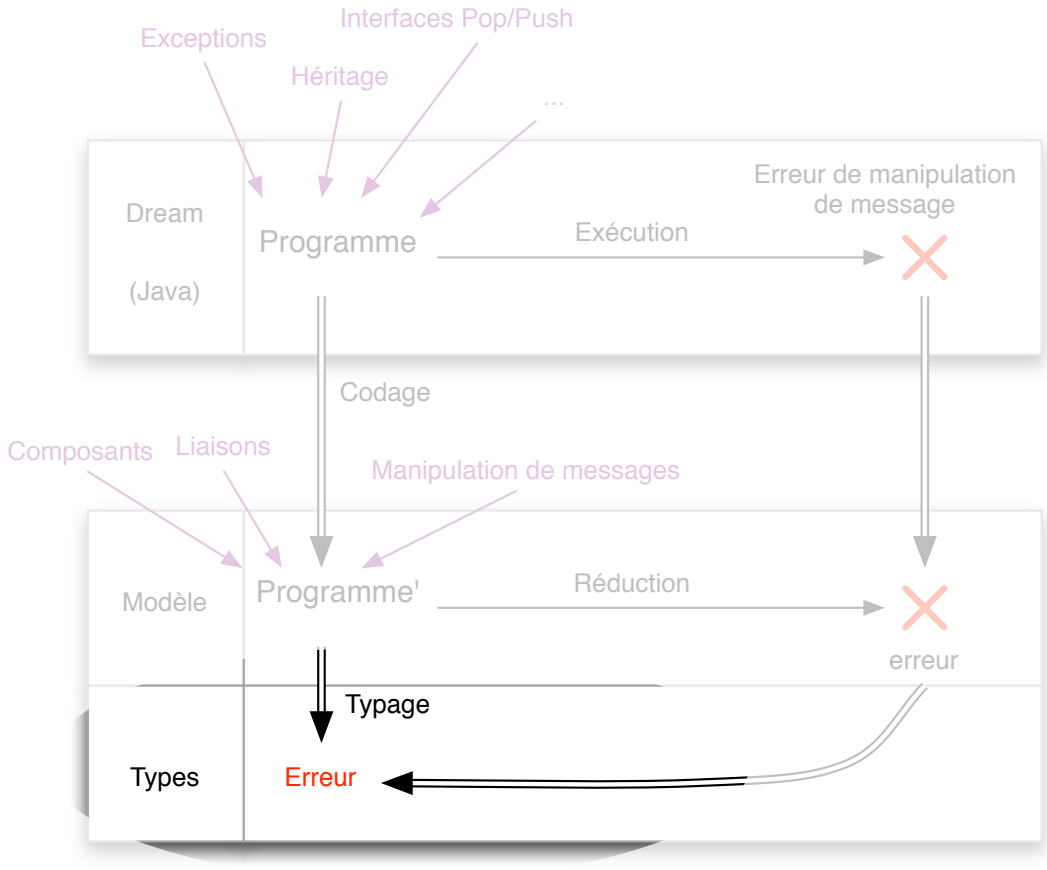
Production par des boucles



$$\text{ProdStr}\left[\text{Loop}\left[\begin{array}{l} e(x).(x@s_1 \\ \quad | \ \{\text{Nom}{=}\text{x}\}\ @s_2) \end{array}\right]\ \begin{array}{l} |\ \text{"test"}\ @e \\ |\ s_1(x).x@e \\ |\ s_2(x).x@s \end{array}\right]$$

Notation hybride





Testing can show the presence of errors, but not their absence.

E. Dijkstra

Typer des messages

Objectif: identifier des opérations erronées sur des messages

- $\{a = 6.9\} + (a = 4)$
- $\{b = 3\}\,.a$
- $\{b = 3\} - a$

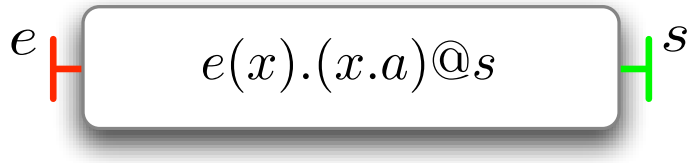
$\Longrightarrow$ suivre la présence des blocs

$$\{\text{Nom} = \text{"test"};\qquad \text{TTL} = 10;\qquad \text{IP} = 192.168.1.1\}$$

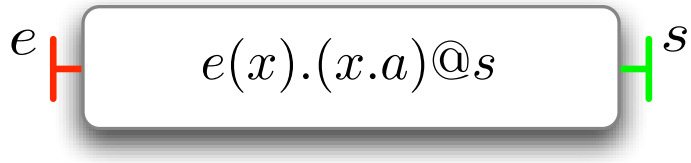
a le type

$$\{\text{Nom} : \mathrm{Pre}(\text{string});\quad \text{TTL} : \mathrm{Pre}(\text{int});\quad \text{IP} : \mathrm{Pre}(\text{ip})\}$$

Vers le polymorphisme de rangées



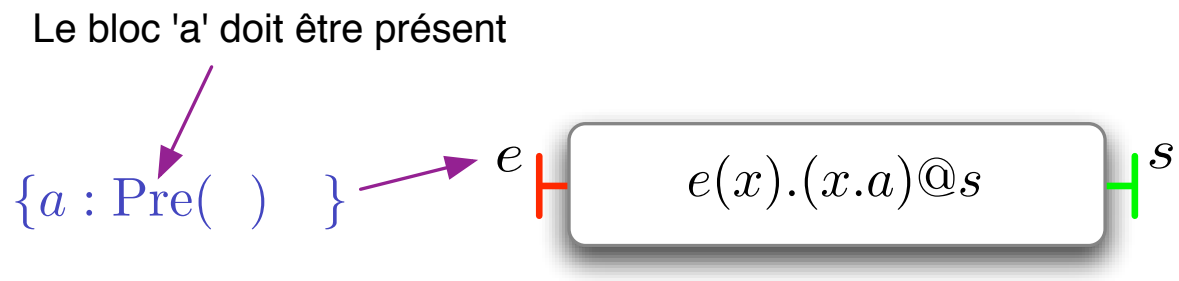
Vers le polymorphisme de rangées



Type fonctionnel

$@e \longrightarrow \quad @s$

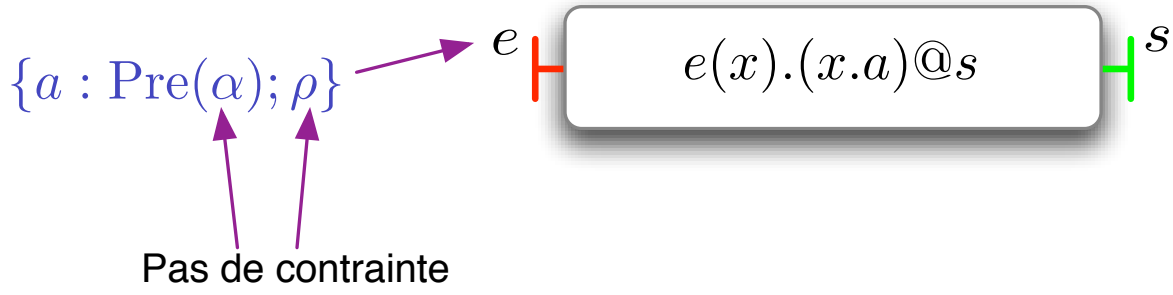
Vers le polymorphisme de rangées



Type fonctionnel

$$\{a : \mathrm{Pre}(\ )\ \}\,@e \rightarrow\ \ @s$$

Vers le polymorphisme de rangées

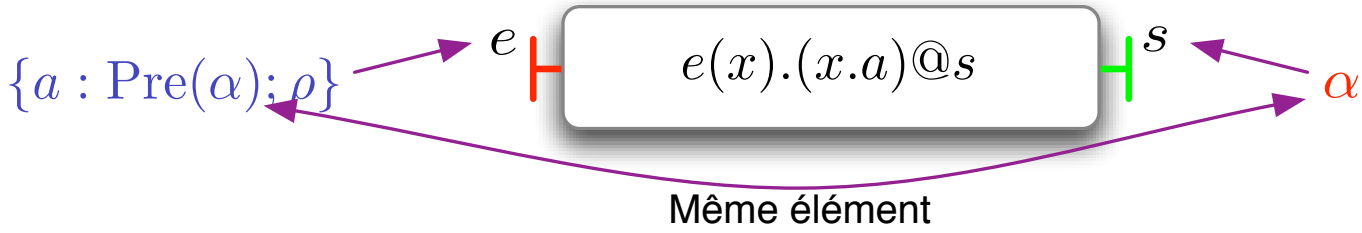


Type fonctionnel

$$\{a : \mathrm{Pre}(\alpha); \rho\}\,@e \rightarrow \quad @s$$

- α variable de type (peut être n'importe quelle type)
- ρ variable de rangée (peut être n'importe quel ensemble de blocs)

Vers le polymorphisme de rangées

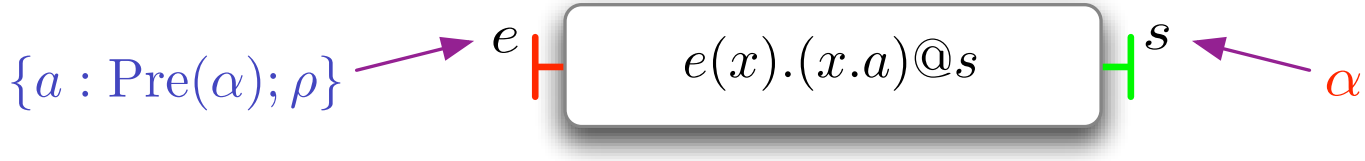


Type fonctionnel avec contraintes (sur α)

$$\{a : \mathrm{Pre}(\alpha); \rho\}\,@e \rightarrow \alpha@s$$

- α variable de type (peut être n'importe quelle type)
- ρ variable de rangée (peut être n'importe quel ensemble de blocs)

Vers le polymorphisme de rangées

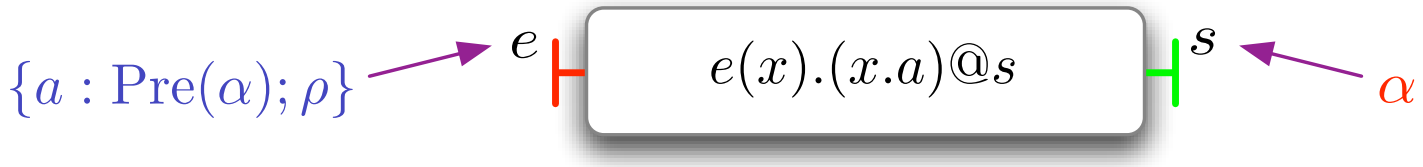


Type fonctionnel avec contraintes (sur α) et polymorphisme

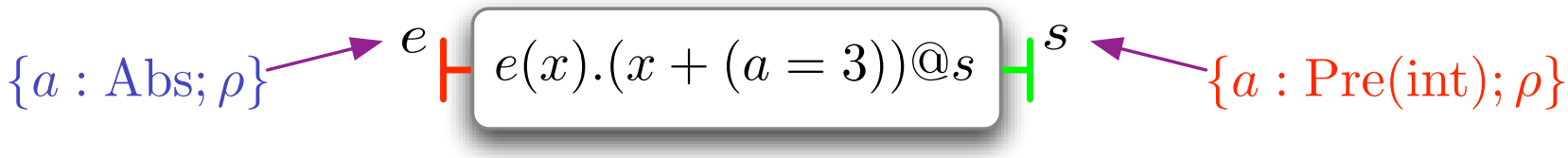
$$\forall \alpha, \rho. \{a : \mathrm{Pre}(\alpha); \rho\}\,@e \rightarrow \alpha@s$$

- α variable de type (peut être n'importe quelle type)
- ρ variable de rangée (peut être n'importe quel ensemble de blocs)

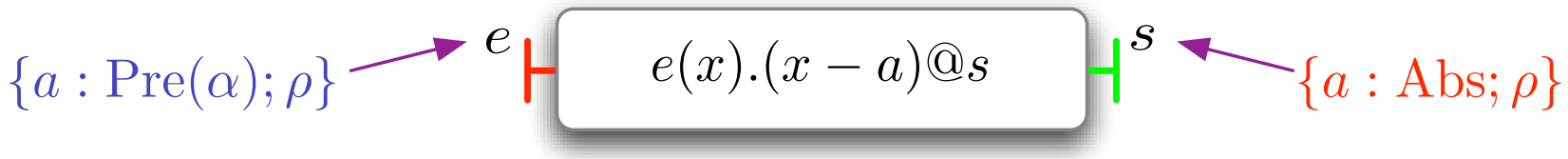
Quelques types



$$\forall \alpha, \rho. \{a : \mathrm{Pre}(\alpha); \rho\}\ @e \rightarrow \alpha @s$$

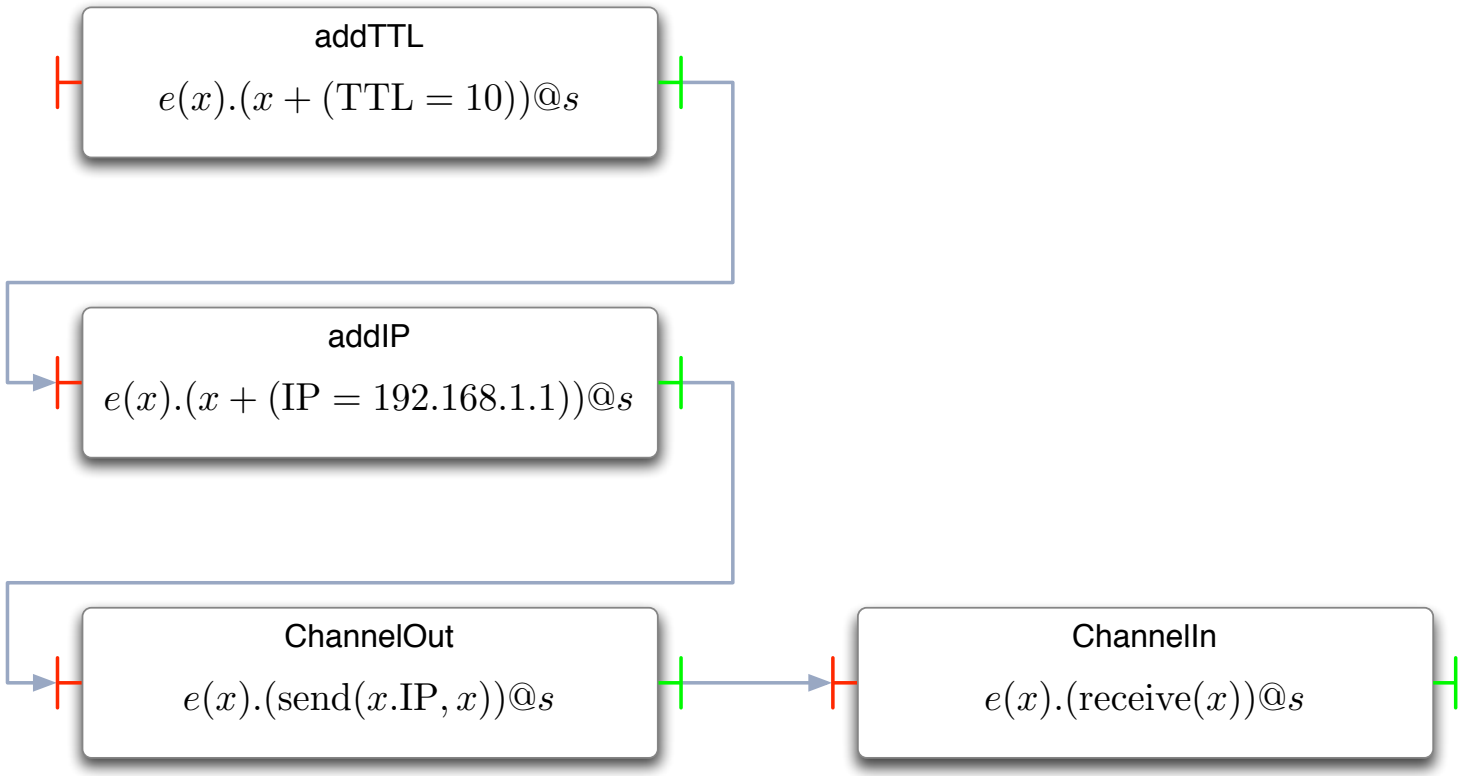


$$\forall \rho. \{a : \mathrm{Abs}; \rho\}\ @e \rightarrow \{a : \mathrm{Pre(int)}; \rho\}\ @s$$

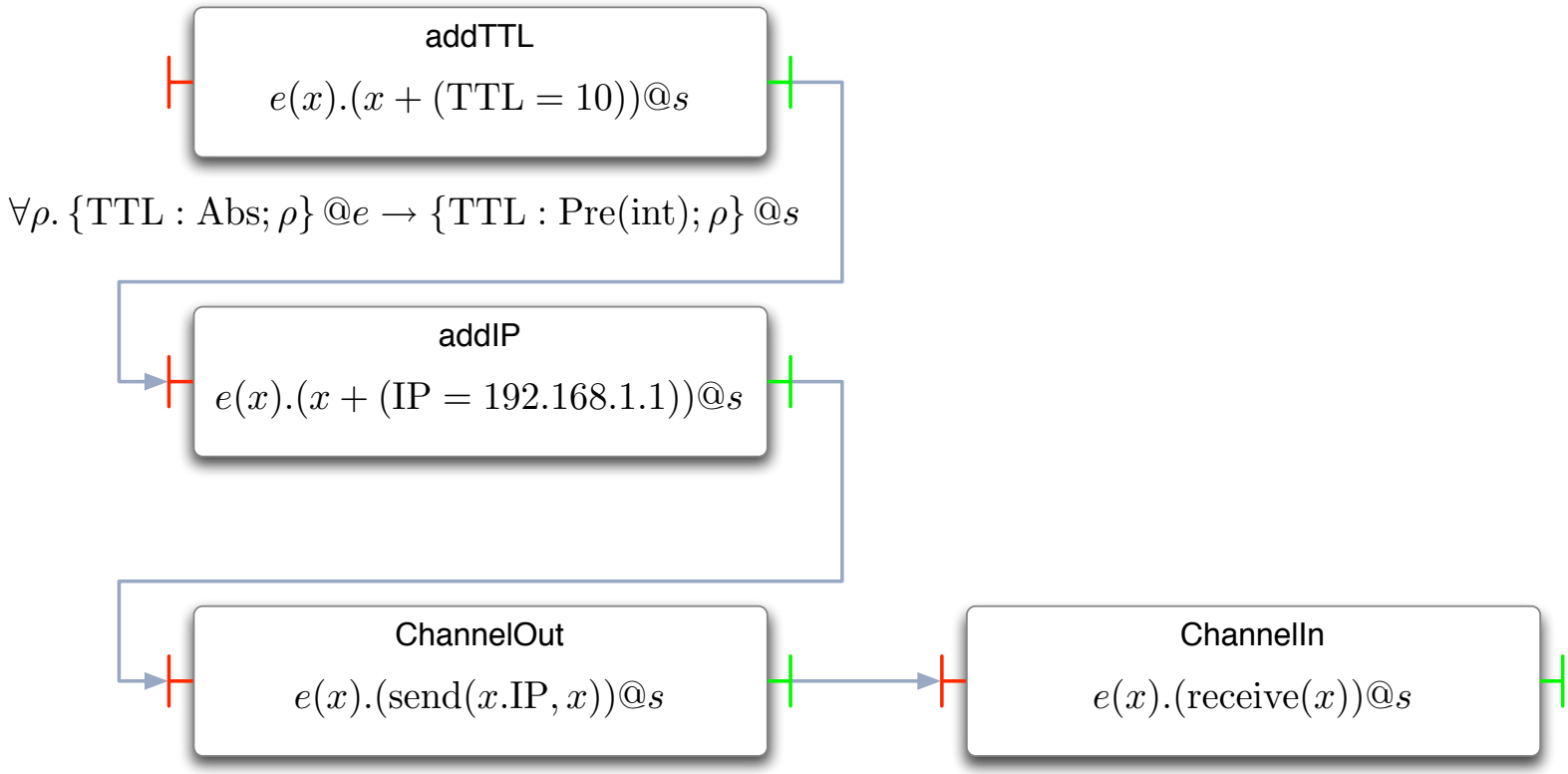


$$\forall \alpha, \rho. \{a : \mathrm{Pre}(\alpha); \rho\}\ @e \rightarrow \{a : \mathrm{Abs}; \rho\}\ @s$$

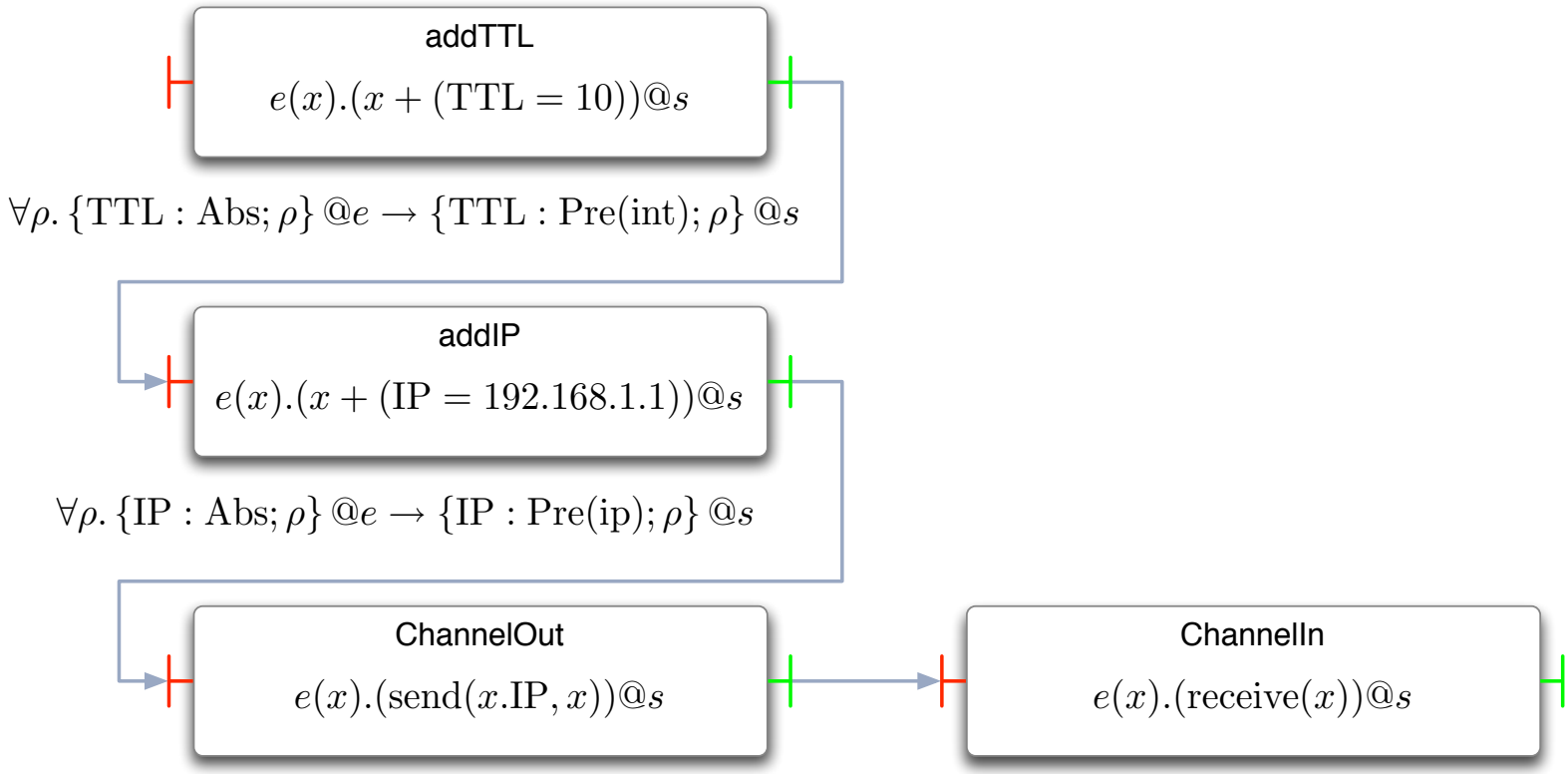
Typage d'un assemblage



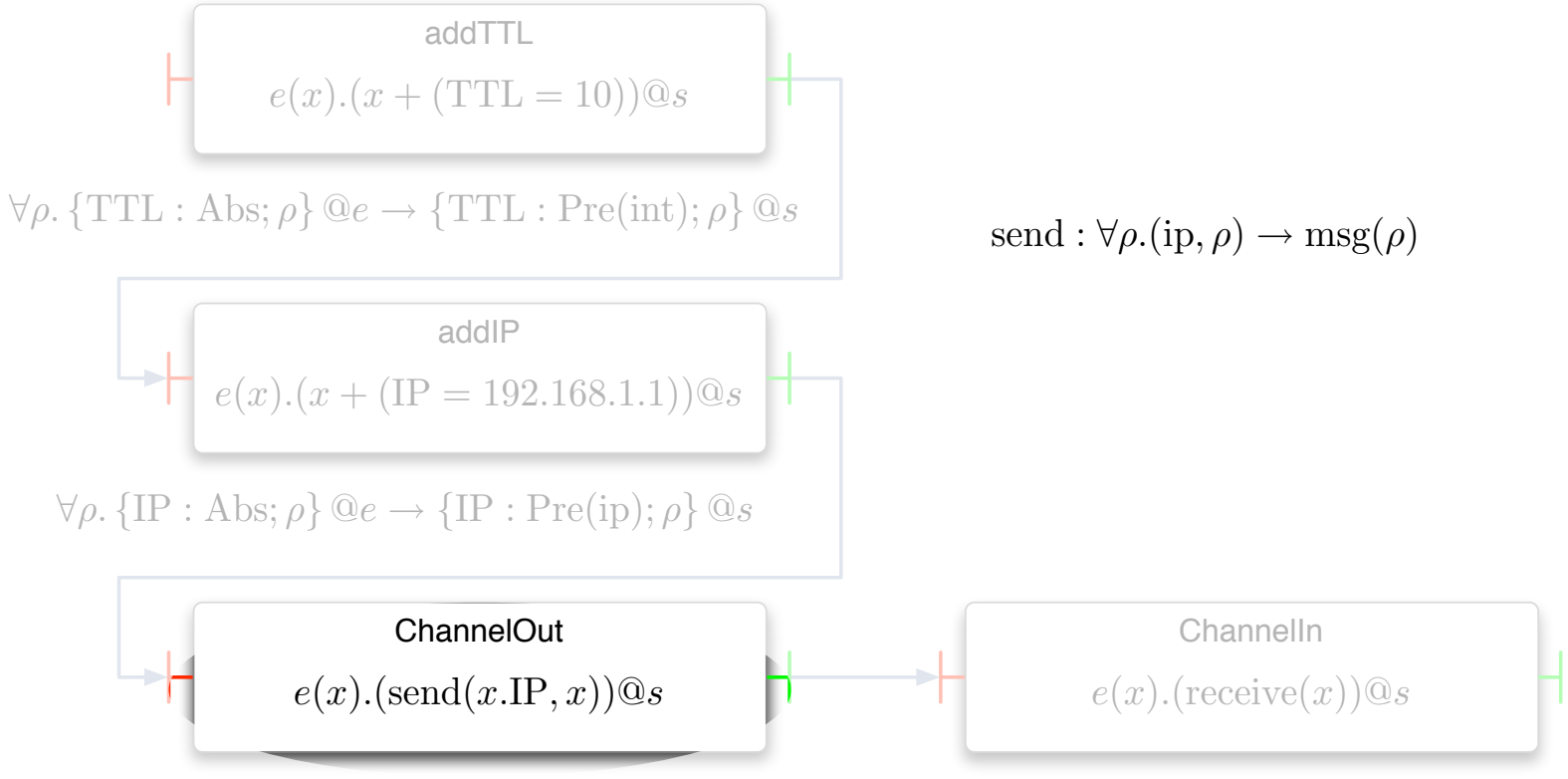
Typage d'un assemblage



Typage d'un assemblage

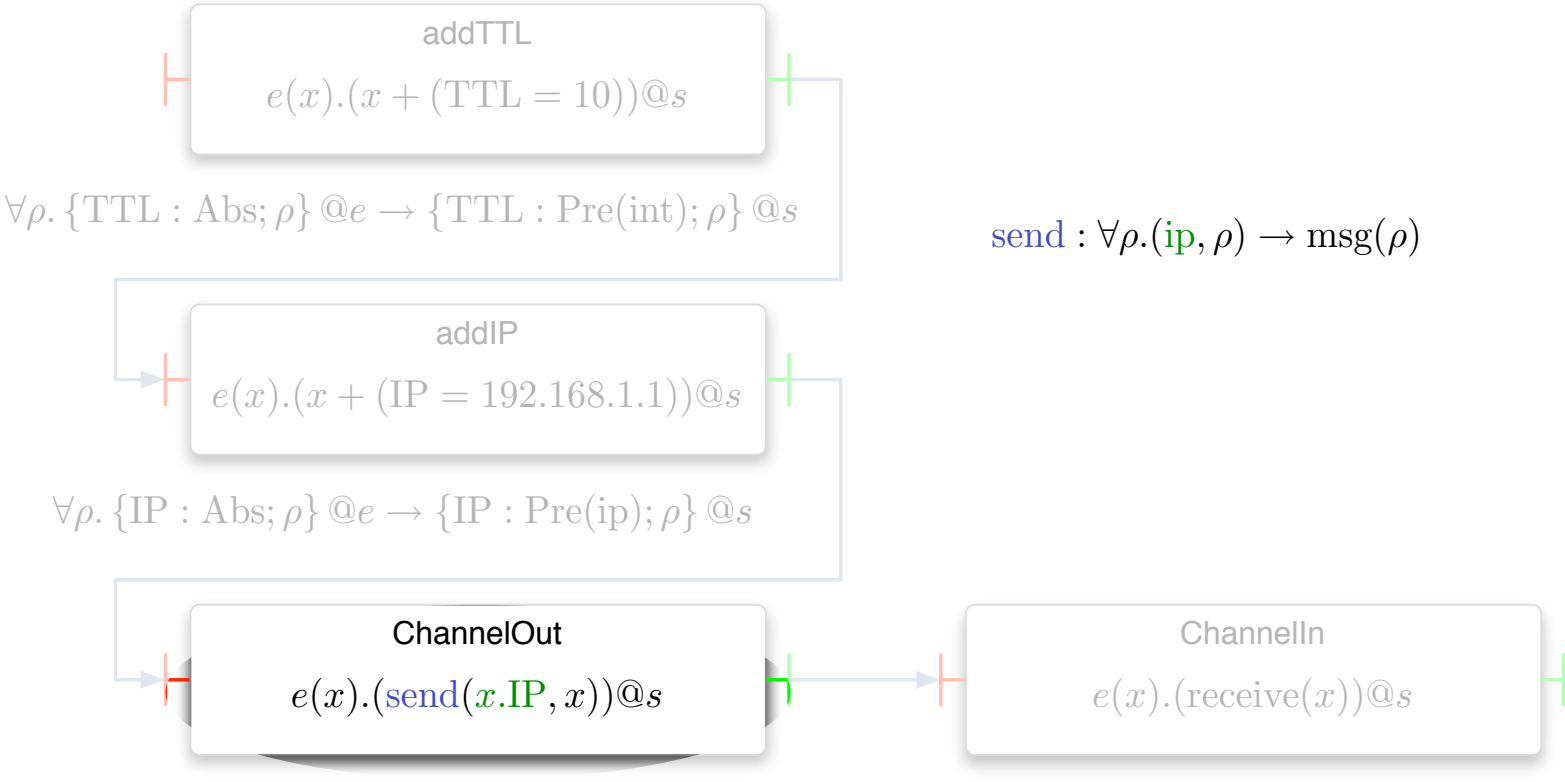


Typage d'un assemblage



$\mathrm{send} : \forall\rho.(\mathrm{ip}, \rho) \to \mathrm{msg}(\rho)$

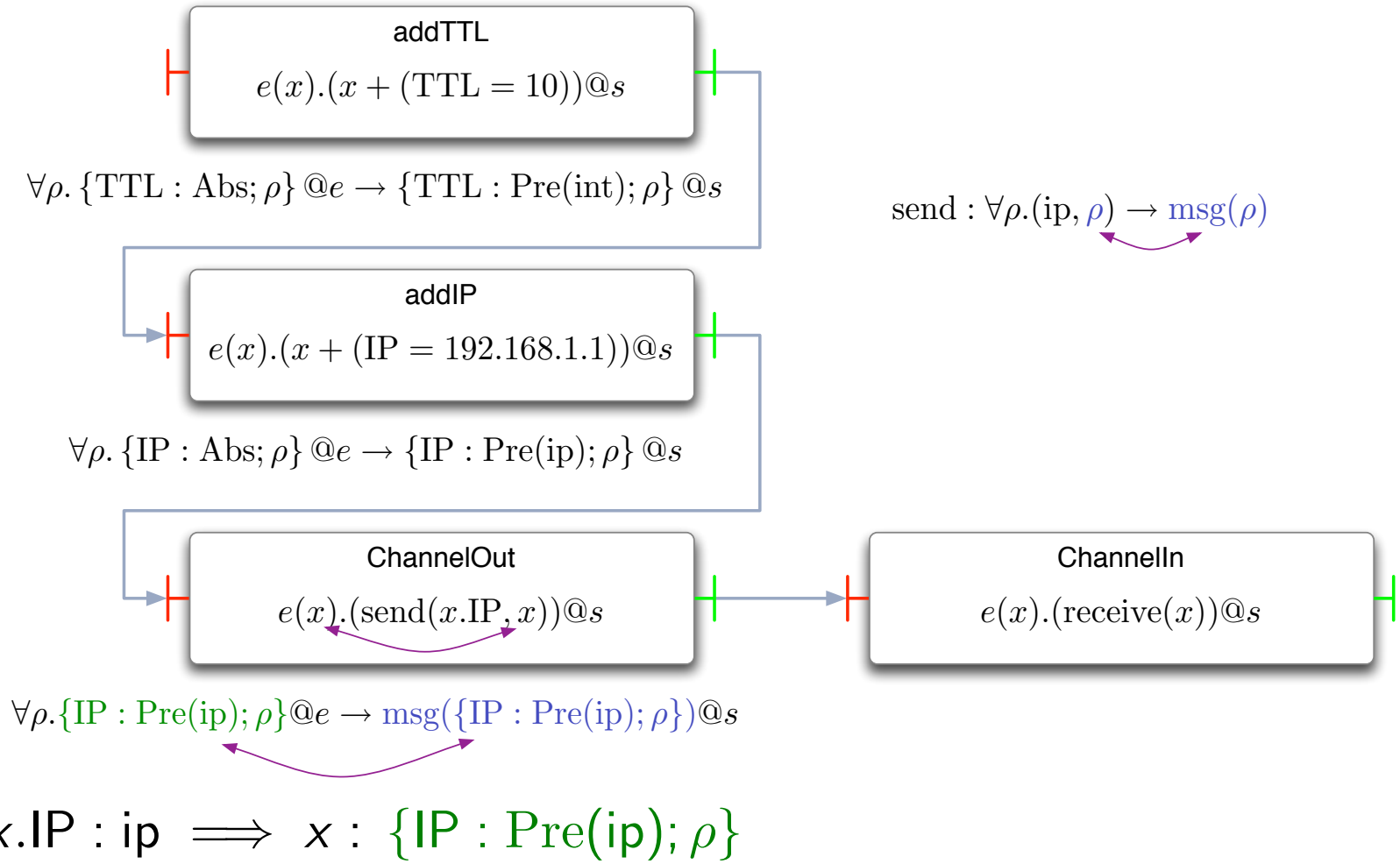
Typage d'un assemblage



$\mathrm{send} : \forall\rho.(\mathrm{ip}, \rho) \to \mathrm{msg}(\rho)$

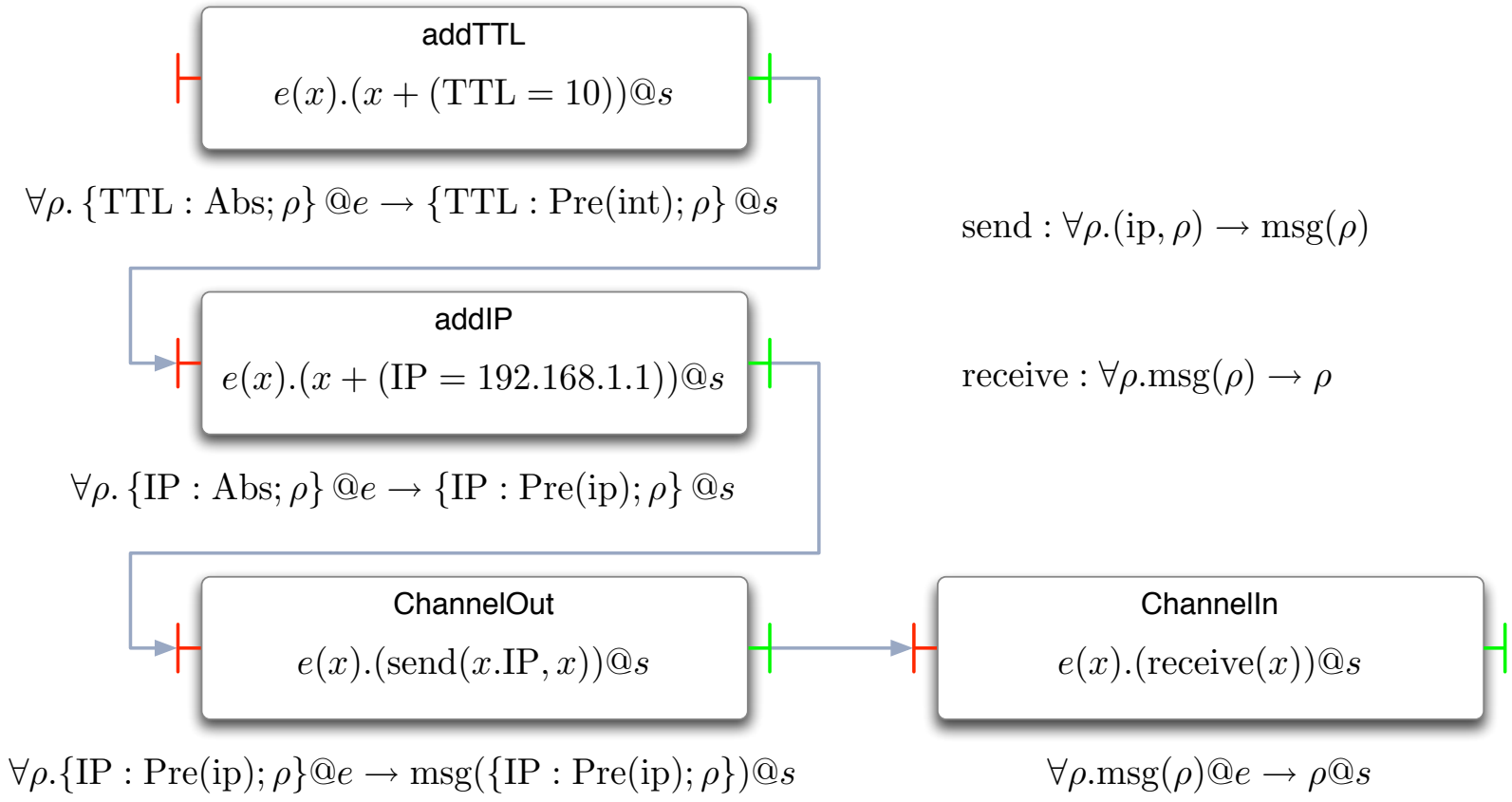
$$x.\mathrm{IP} : \mathrm{ip} \implies x : \{\mathrm{IP} : \mathrm{Pre}(\mathrm{ip}); \rho\}$$

Typage d'un assemblage

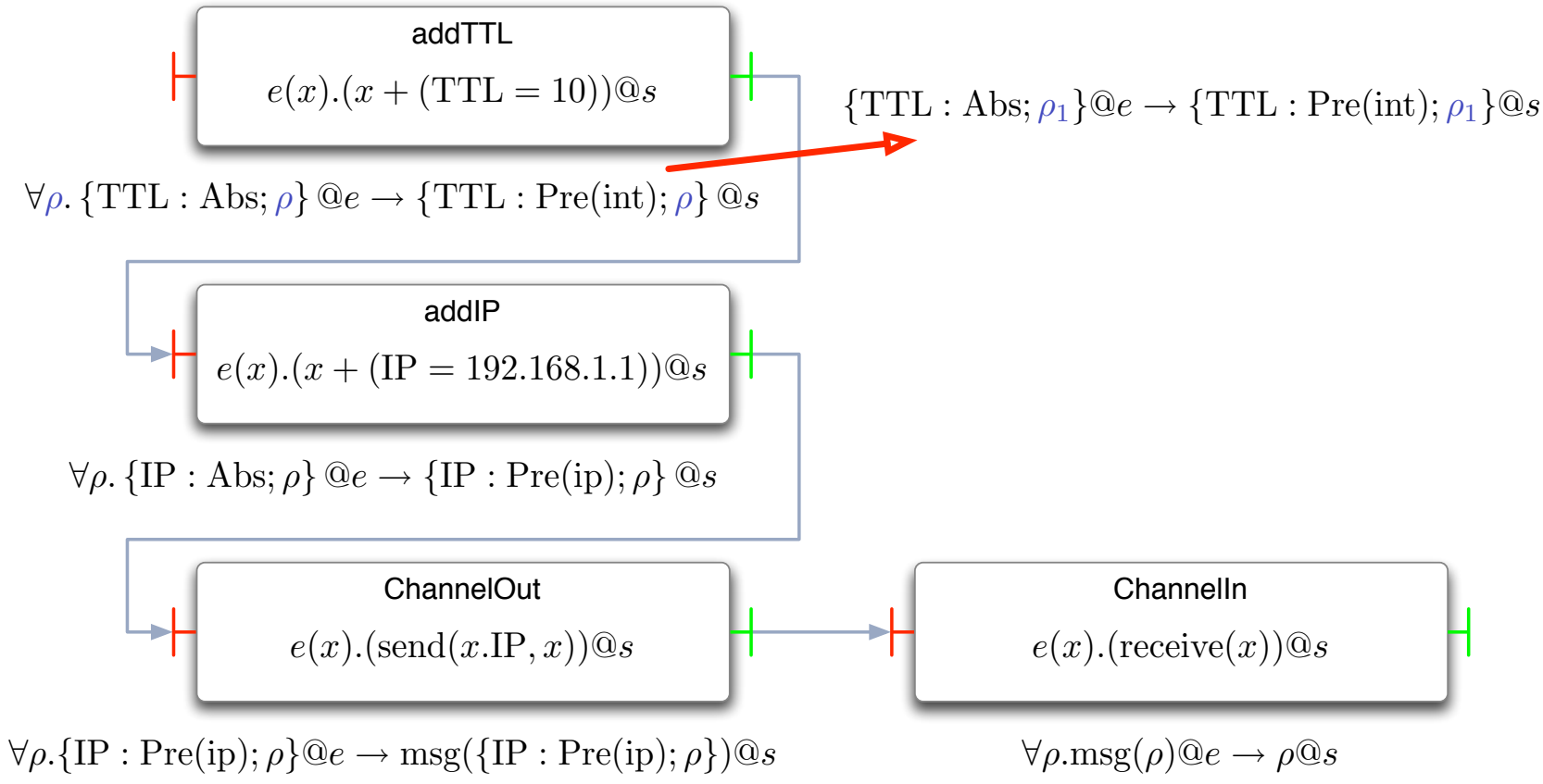


$$x.\mathsf{IP} : \mathsf{ip} \implies x : \{\mathrm{IP} : \mathrm{Pre}(\mathrm{ip}); \rho\}$$

Typage d'un assemblage

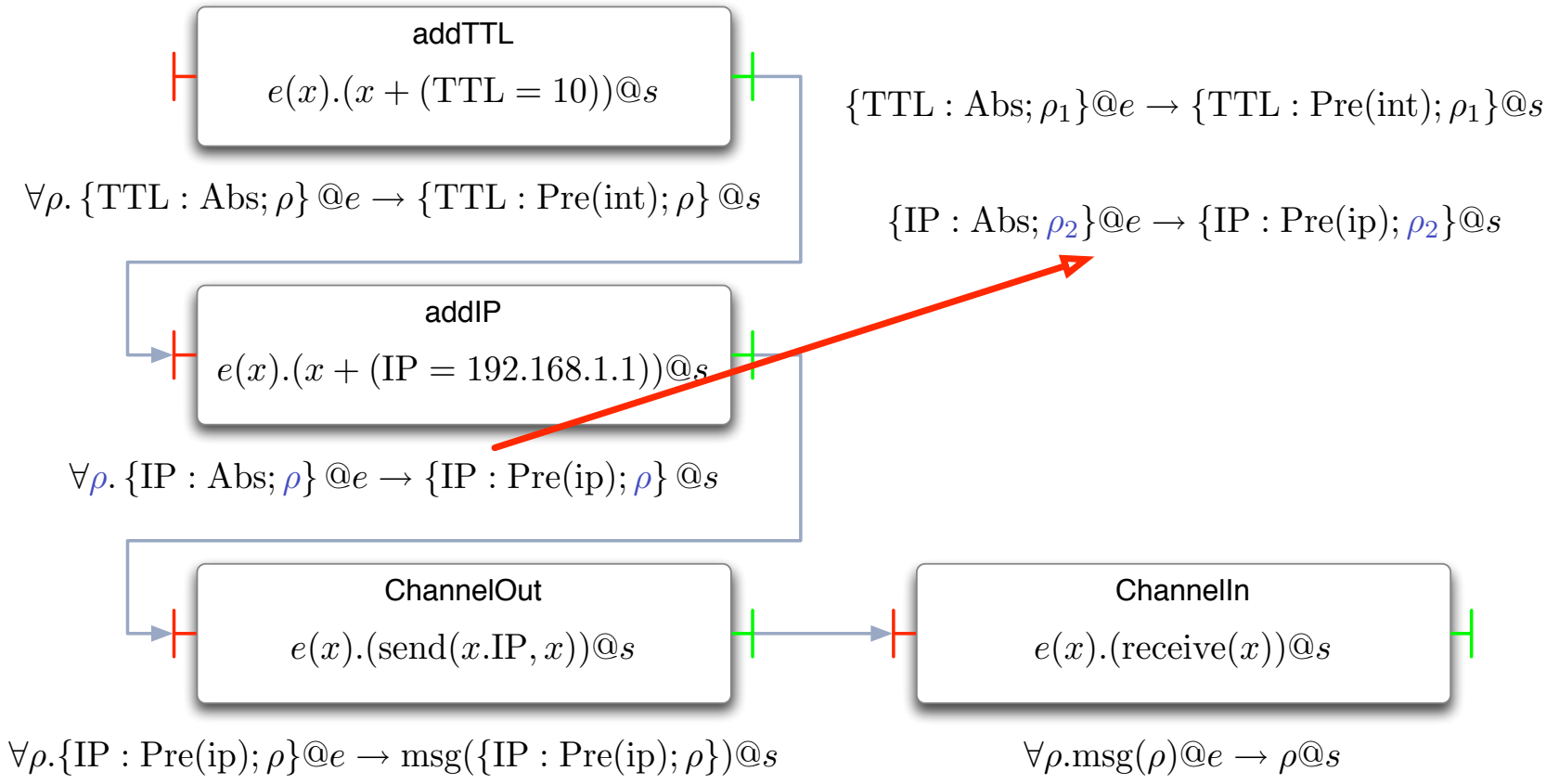


Typage d'un assemblage



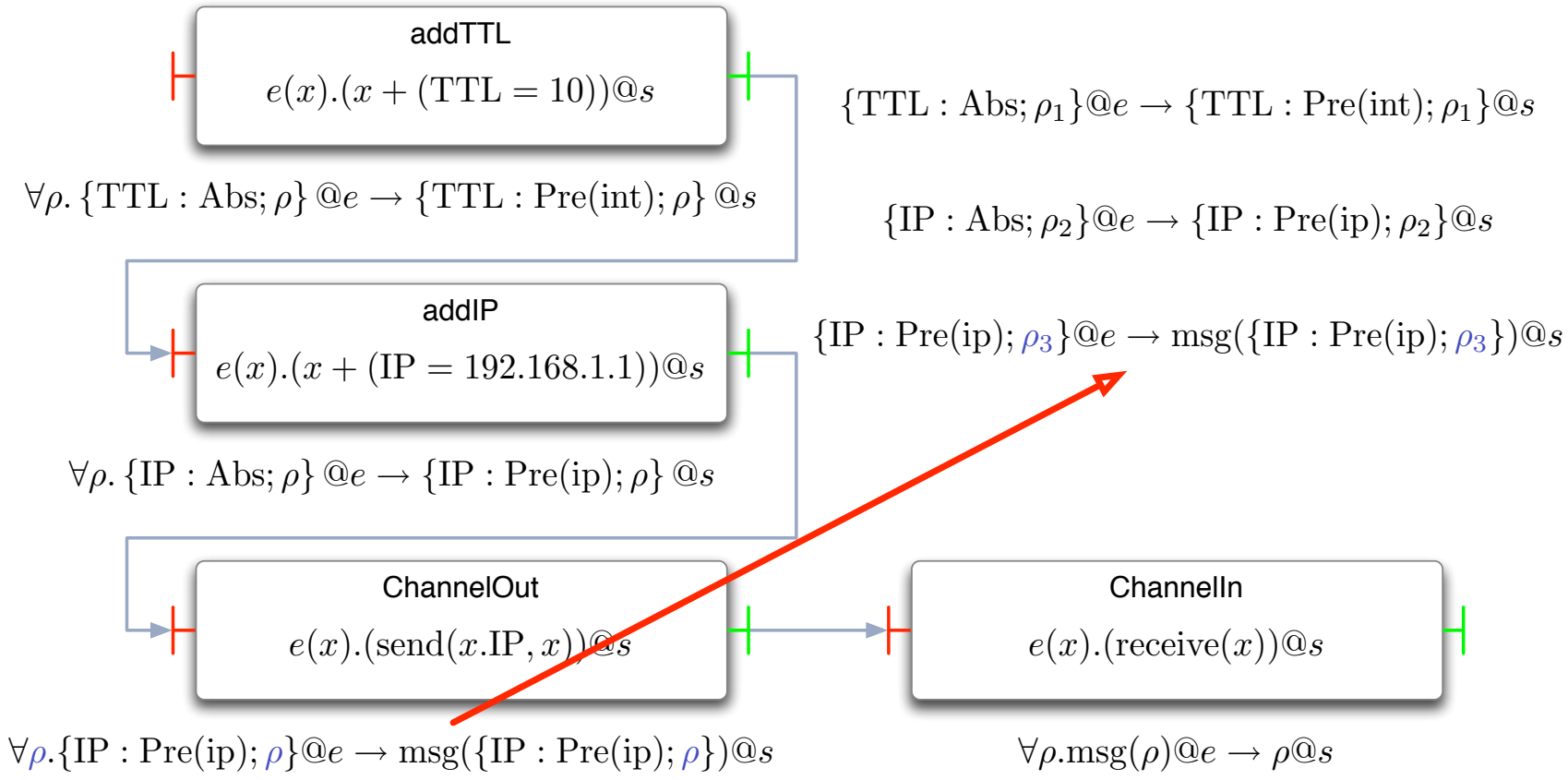
Instanciation de variables

Typage d'un assemblage



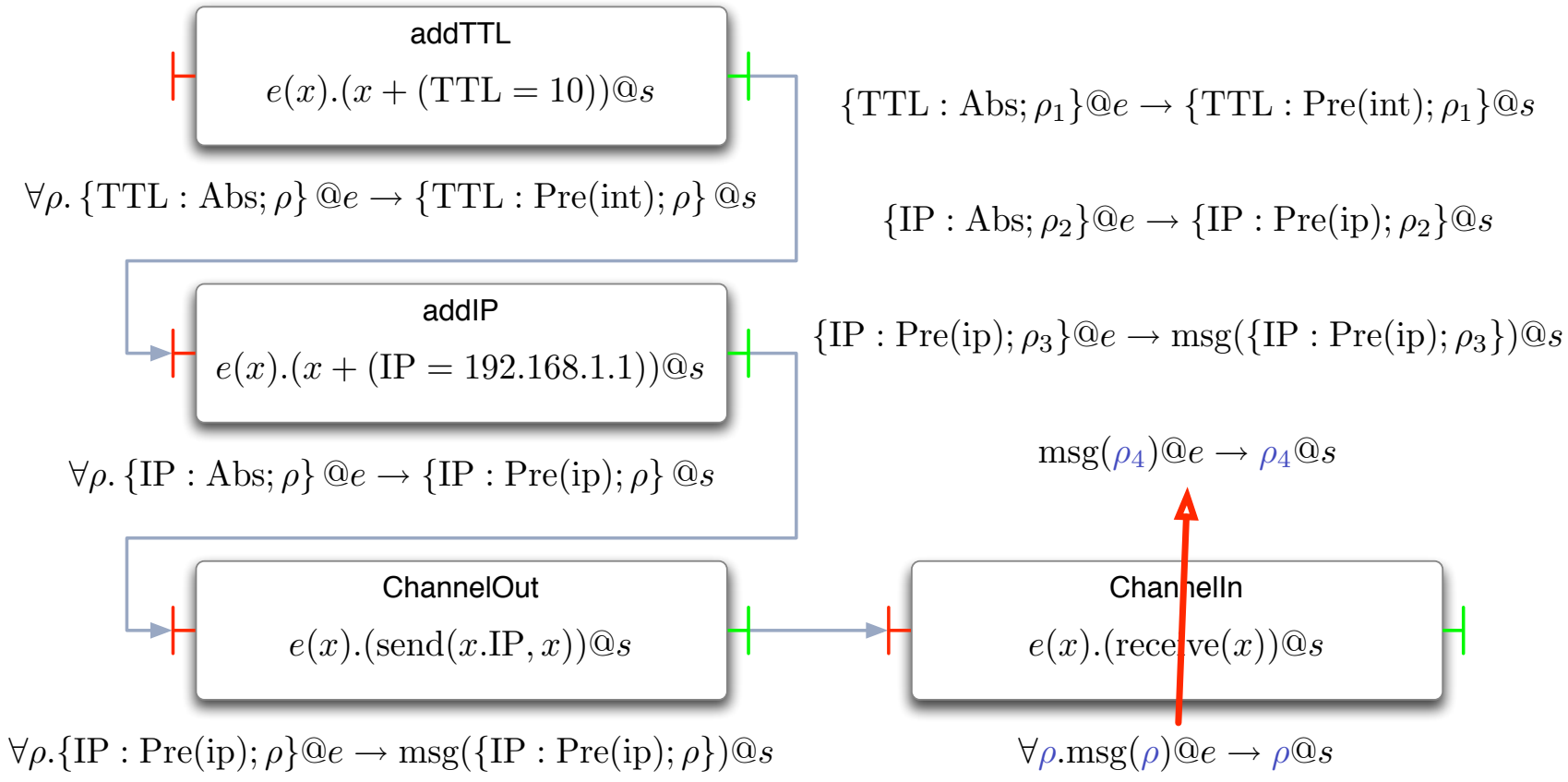
Instanciation de variables

Typage d'un assemblage



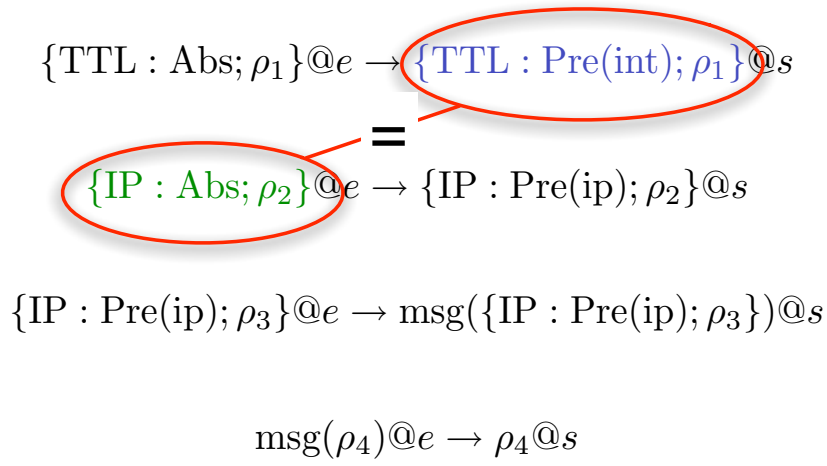
Instanciation de variables

Typage d'un assemblage



Instanciation de variables

Typage d'un assemblage



$$\rho_1 = \{\text{IP} : \text{Abs}; \rho'\}\,, \quad \rho_2 = \{\text{TTL} : \text{Pre(int)}; \rho'\}$$

Typage d'un assemblage

$\{\text{TTL} : \text{Abs}; \text{IP} : \text{Abs}; \rho'\}@e \rightarrow$

$\{\text{TTL} : \text{Pre(int)}; \text{IP} : \text{Pre(ip)}; \rho'\}@s$

$\{\text{TTL} : \text{Abs}; \rho_1\}@e \rightarrow \{\text{TTL} : \text{Pre(int)}; \rho_1\}@s$

$=$

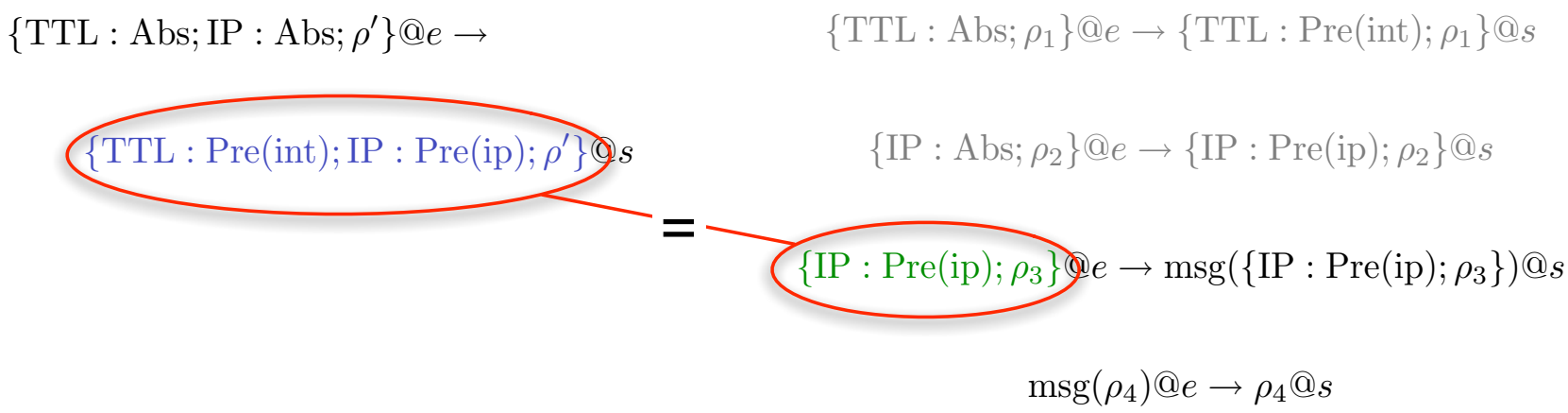
$\{\text{IP} : \text{Abs}; \rho_2\}@e \rightarrow \{\text{IP} : \text{Pre(ip)}; \rho_2\}@s$

$\{\text{IP} : \text{Pre(ip)}; \rho_3\}@e \rightarrow \text{msg}(\{\text{IP} : \text{Pre(ip)}; \rho_3\})@s$

$\text{msg}(\rho_4)@e \rightarrow \rho_4@s$

$$\rho_1 = \{\text{IP} : \text{Abs}; \rho'\}, \quad \rho_2 = \{\text{TTL} : \text{Pre(int)}; \rho'\}$$

Typage d'un assemblage



$\rho_3 = \{\text{TTL} : \text{Pre(int)}; \rho'\}$

Typage d'un assemblage

$\{\text{TTL} : \text{Abs}; \text{IP} : \text{Abs}; \rho'\}@e \rightarrow$

$\{\text{TTL} : \text{Abs}; \rho_1\}@e \rightarrow \{\text{TTL} : \text{Pre(int)}; \rho_1\}@s$

$\{\text{IP} : \text{Abs}; \rho_2\}@e \rightarrow \{\text{IP} : \text{Pre(ip)}; \rho_2\}@s$

$\text{msg}(\{\text{TTL} : \text{Pre(int)}; \text{IP} : \text{Pre(ip)}; \rho'\})@s$ $\{\text{IP} : \text{Pre(ip)}; \rho_3\}@e \rightarrow \text{msg}(\{\text{IP} : \text{Pre(ip)}; \rho_3\})@s$

$\text{msg}(\rho_4)@e \rightarrow \rho_4@s$

$\rho_3 = \{\text{TTL} : \text{Pre(int)}; \rho'\}$

Typage d'un assemblage

$\{\text{TTL} : \text{Abs}; \text{IP} : \text{Abs}; \rho'\}@e \rightarrow$

$\{\text{TTL} : \text{Abs}; \rho_1\}@e \rightarrow \{\text{TTL} : \text{Pre(int)}; \rho_1\}@s$

$\{\text{IP} : \text{Abs}; \rho_2\}@e \rightarrow \{\text{IP} : \text{Pre(ip)}; \rho_2\}@s$

$\text{msg}(\{\text{TTL} : \text{Pre(int)}; \text{IP} : \text{Pre(ip)}; \rho'\})@s$ $\{\text{IP} : \text{Pre(ip)}; \rho_3\}@e \rightarrow \text{msg}(\{\text{IP} : \text{Pre(ip)}; \rho_3\})@s$

$=$

$\text{msg}(\rho_4)@e \rightarrow \rho_4@s$

$\rho_4 = \{\text{TTL} : \text{Pre(int)}; \text{IP} : \text{Pre(ip)}; \rho'\}$

Typage d'un assemblage

$\{\text{TTL} : \text{Abs}; \text{IP} : \text{Abs}; \rho'\}@e \rightarrow$

$\{\text{TTL} : \text{Abs}; \rho_1\}@e \rightarrow \{\text{TTL} : \text{Pre(int)}; \rho_1\}@s$

$\{\text{IP} : \text{Abs}; \rho_2\}@e \rightarrow \{\text{IP} : \text{Pre(ip)}; \rho_2\}@s$

$\{\text{IP} : \text{Pre(ip)}; \rho_3\}@e \rightarrow \text{msg}(\{\text{IP} : \text{Pre(ip)}; \rho_3\})@s$

$\{\text{TTL} : \text{Pre(int)}; \text{IP} : \text{Pre(ip)}; \rho'\}@s$

$\text{msg}(\rho_4)@e \rightarrow \rho_4@s$

$$\rho_4 = \{\text{TTL} : \text{Pre(int)}; \text{IP} : \text{Pre(ip)}; \rho'\}$$

Typage d'un assemblage

$\{\text{TTL} : \text{Abs}; \text{IP} : \text{Abs}; \rho'\}@e \rightarrow$

$\{\text{TTL} : \text{Abs}; \rho_1\}@e \rightarrow \{\text{TTL} : \text{Pre(int)}; \rho_1\}@s$

$\{\text{IP} : \text{Abs}; \rho_2\}@e \rightarrow \{\text{IP} : \text{Pre(ip)}; \rho_2\}@s$

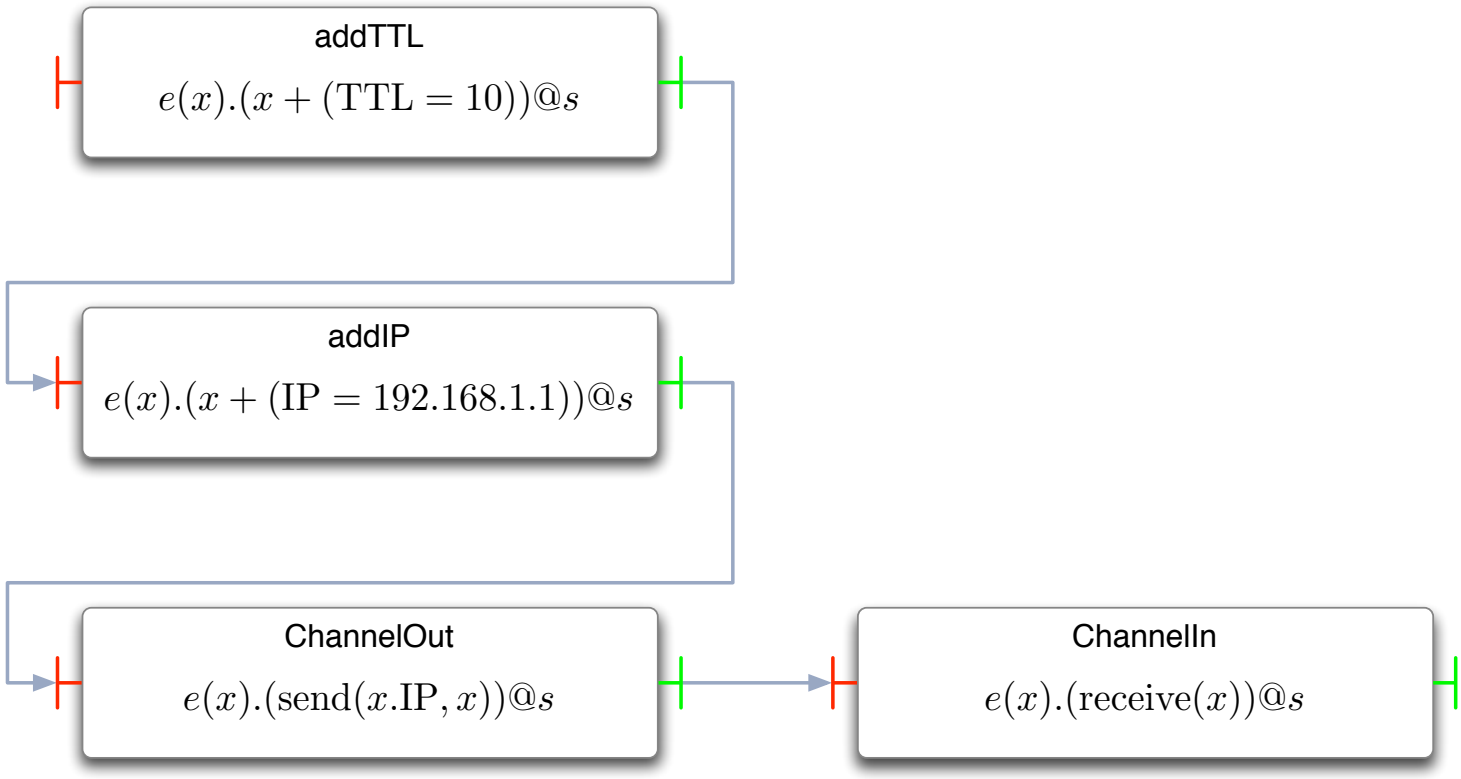
$\{\text{IP} : \text{Pre(ip)}; \rho_3\}@e \rightarrow \text{msg}(\{\text{IP} : \text{Pre(ip)}; \rho_3\})@s$

$\{\text{TTL} : \text{Pre(int)}; \text{IP} : \text{Pre(ip)}; \rho'\}@s$

$\text{msg}(\rho_4)@e \rightarrow \rho_4@s$

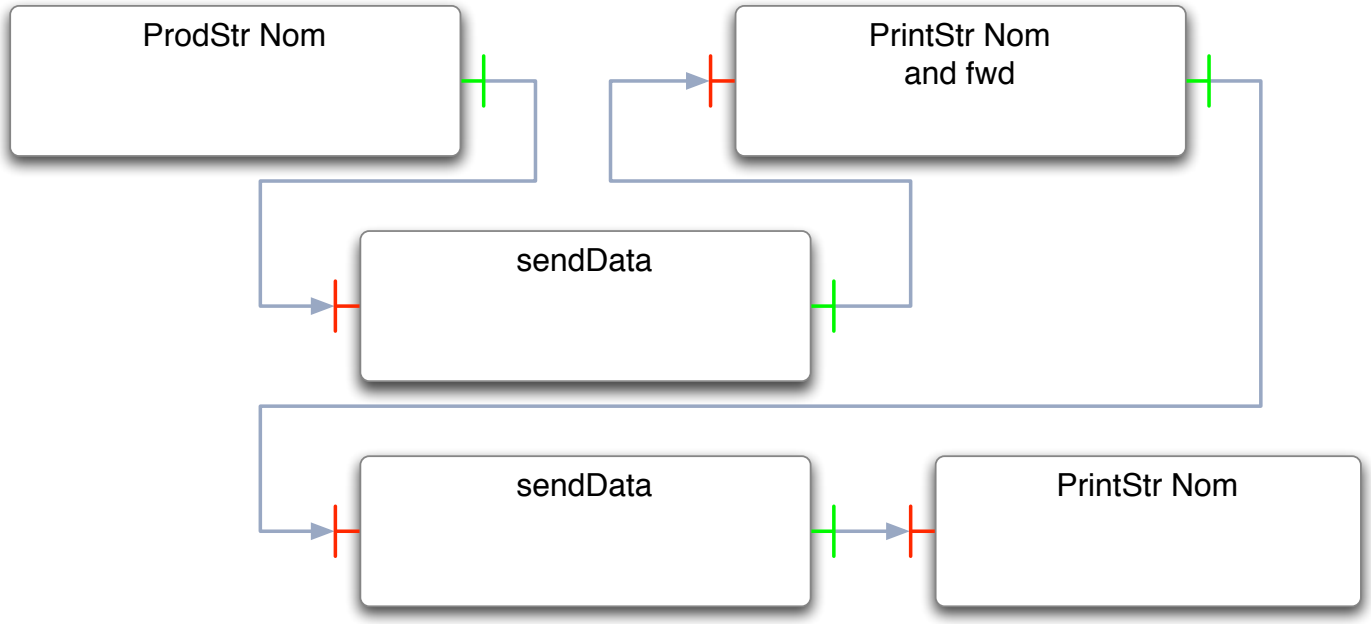
$$\forall \rho'. \{\text{TTL} : \text{Abs}; \text{IP} : \text{Abs}; \rho'\}\,@e \rightarrow \{\text{TTL} : \text{Pre(int)}; \text{IP} : \text{Pre(ip)}; \rho'\}@s$$

Typage d'un assemblage

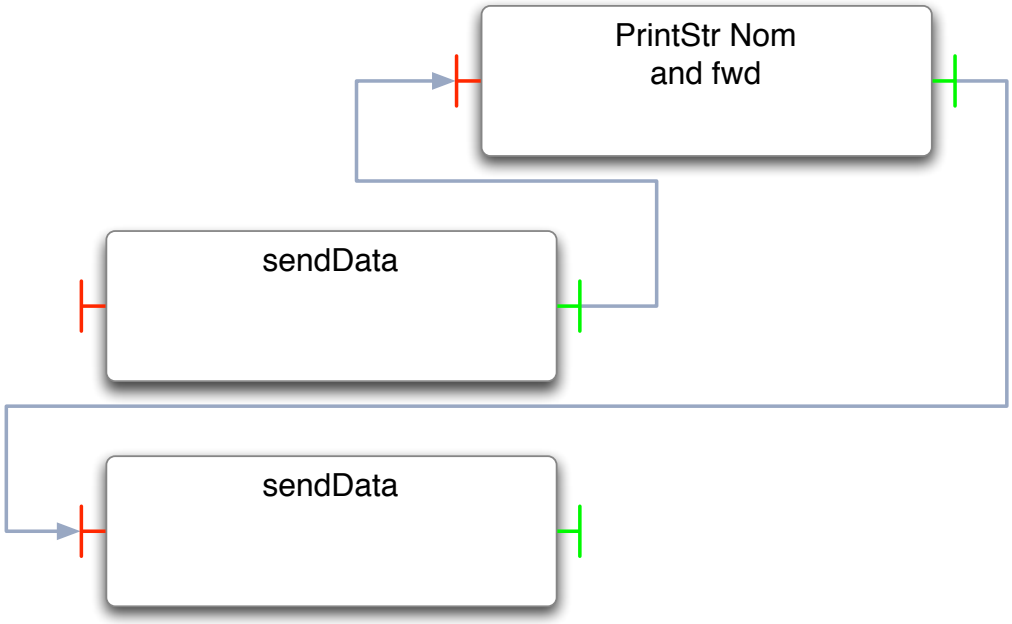


$$\forall \rho'.\, \{\mathsf{TTL} : \mathrm{Abs}; \mathsf{IP} : \mathrm{Abs}; \rho'\}\, @e \rightarrow \{\mathsf{TTL} : \mathrm{Pre}(\mathsf{int}); \mathsf{IP} : \mathrm{Pre}(\mathsf{ip}); \rho'\} @s$$

Détection d'erreur

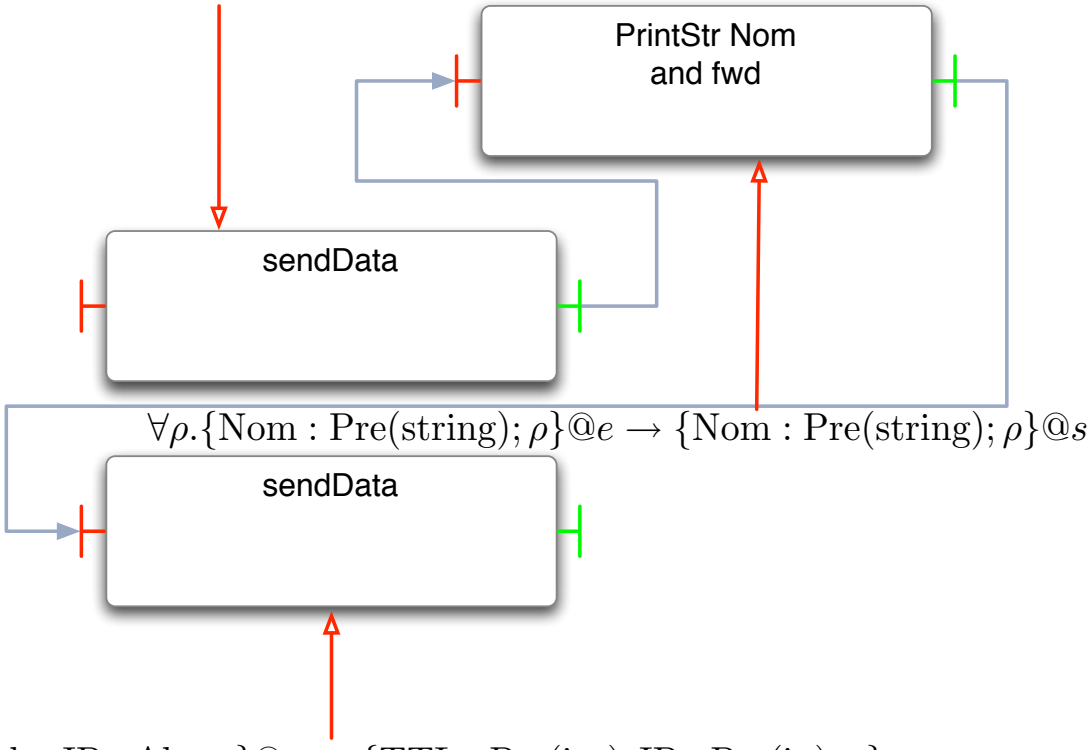


Détection d'erreur



Détection d'erreur

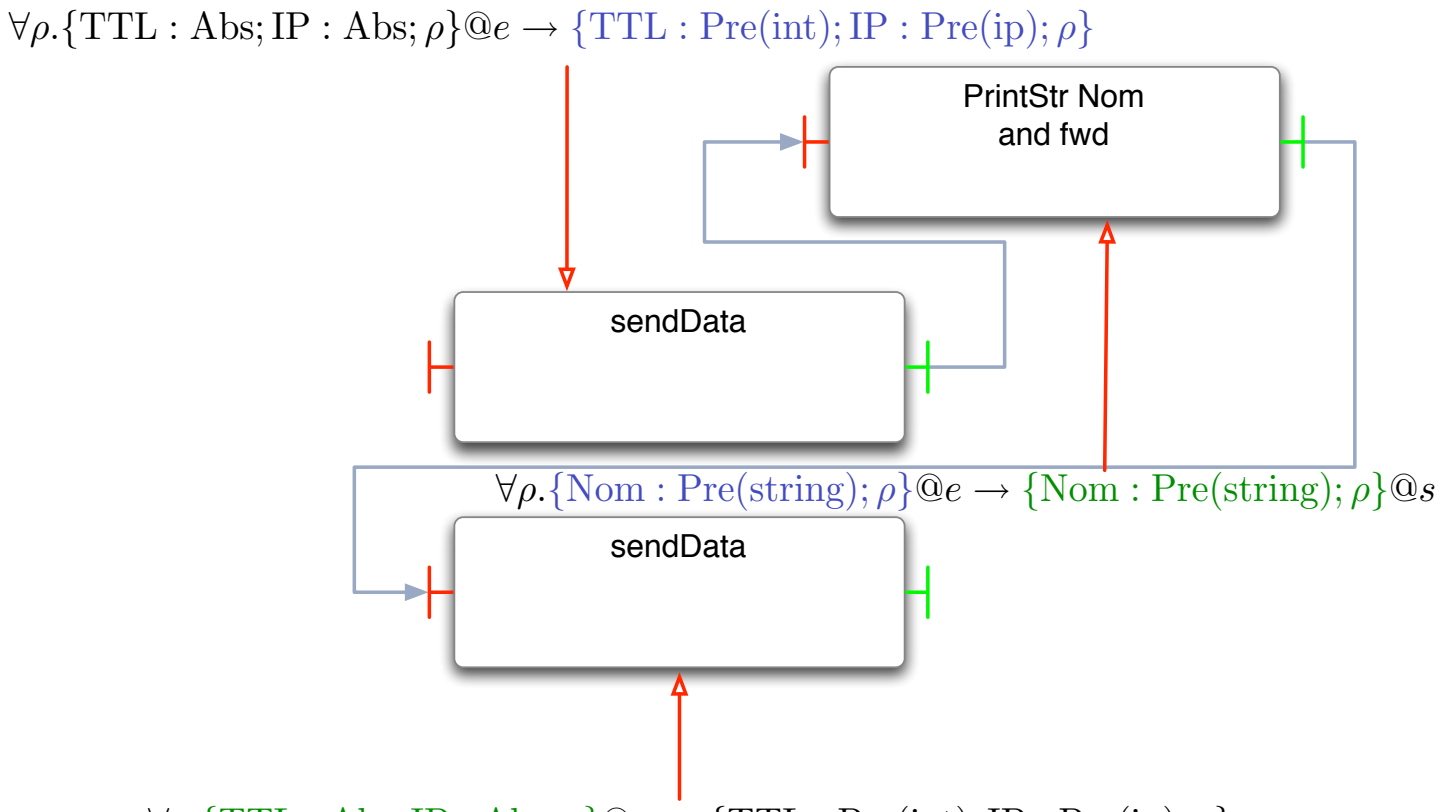
$\forall\rho.\{\text{TTL} : \text{Abs}; \text{IP} : \text{Abs}; \rho\}@e \rightarrow \{\text{TTL} : \text{Pre}(\text{int}); \text{IP} : \text{Pre}(\text{ip}); \rho\}$



$\forall\rho.\{\text{Nom} : \text{Pre}(\text{string}); \rho\}@e \rightarrow \{\text{Nom} : \text{Pre}(\text{string}); \rho\}@s$

$\forall\rho.\{\text{TTL} : \text{Abs}; \text{IP} : \text{Abs}; \rho\}@e \rightarrow \{\text{TTL} : \text{Pre}(\text{int}); \text{IP} : \text{Pre}(\text{ip}); \rho\}$

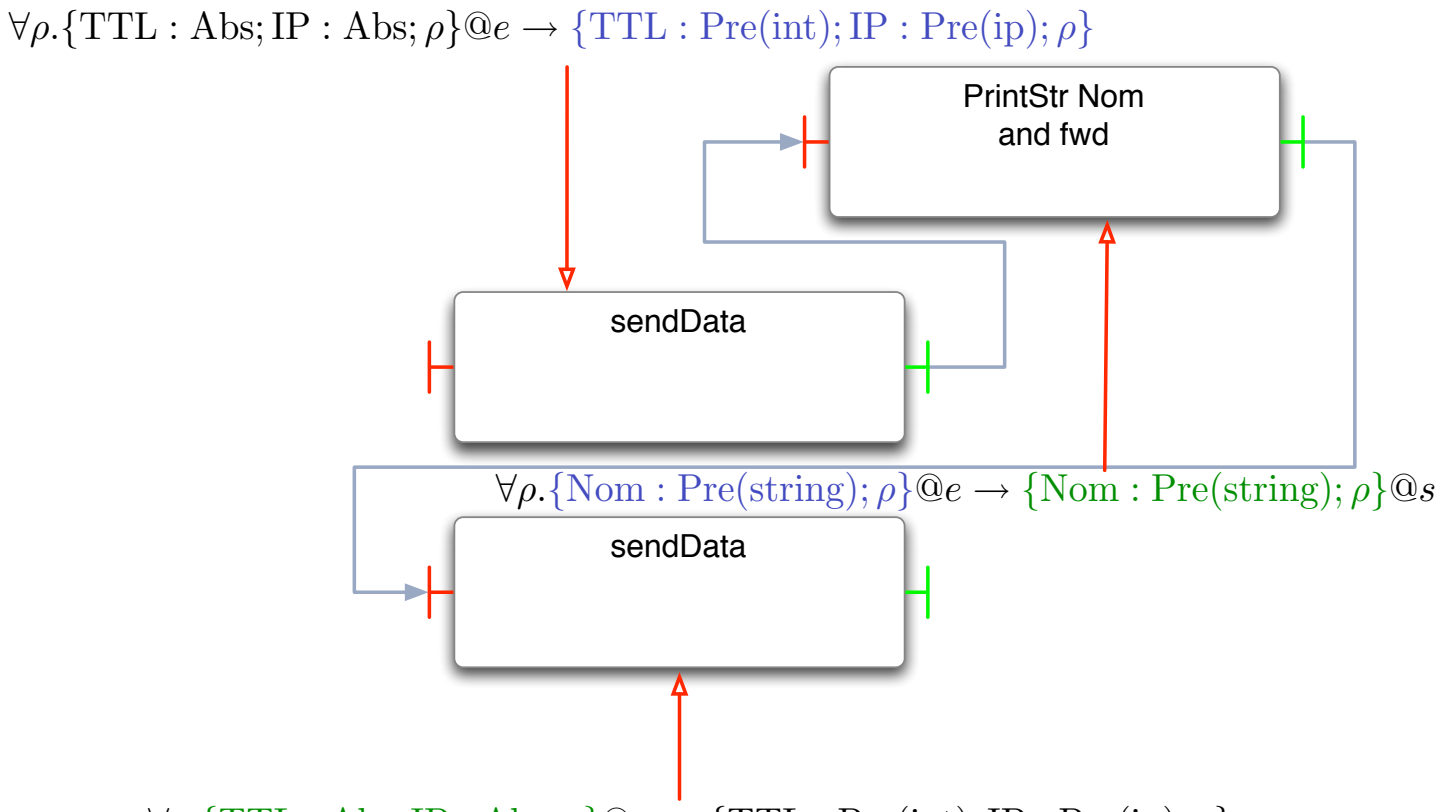
Détection d'erreur



$$\{\text{TTL} : \text{Pre(int)}; \text{IP} : \text{Pre(ip)}; \rho_1\}$$

$$\stackrel{?}{=} \{\text{Nom} : \text{Pre(string)}; \rho_2\} \stackrel{?}{=} \{\text{TTL} : \text{Abs}; \text{IP} : \text{Abs}; \rho_3\}$$

Détection d'erreur



$$\{\text{TTL} : \text{Pre(int)}; \text{IP} : \text{Pre(ip)}; \rho_1\}$$

$$\stackrel{?}{=} \{\text{Nom} : \text{Pre(string)}; \rho_2\} \neq \{\text{TTL} : \text{Abs}; \text{IP} : \text{Abs}; \rho_3\}$$

Correction

Définition

Une *erreur* survient dans un programme Dream quand un opérateur est appliqué à un mauvais argument.

Définition

Un *'bon' système de types* est correct :

1. Aucun programme typé n'a d'erreur.
2. Tout programme typé se réduit en un programme typé.

Théorème

Notre système de type est correct.

[*Type Inference for Records in a Natural Extension of ML*, Didier Rémy, 1993]

Inférence de types

Inférence de Types

Un algorithme *d'inférence* est une fonction qui prend un programme en argument et retourne:

- soit "intypable",
- soit un programme annoté par des types.

Correction et complétude

Un algorithme d'inférence devrait être:

correct : les annotations correspondent à un typage,

complet : l'algorithme trouve des annotations pour un programme typable.

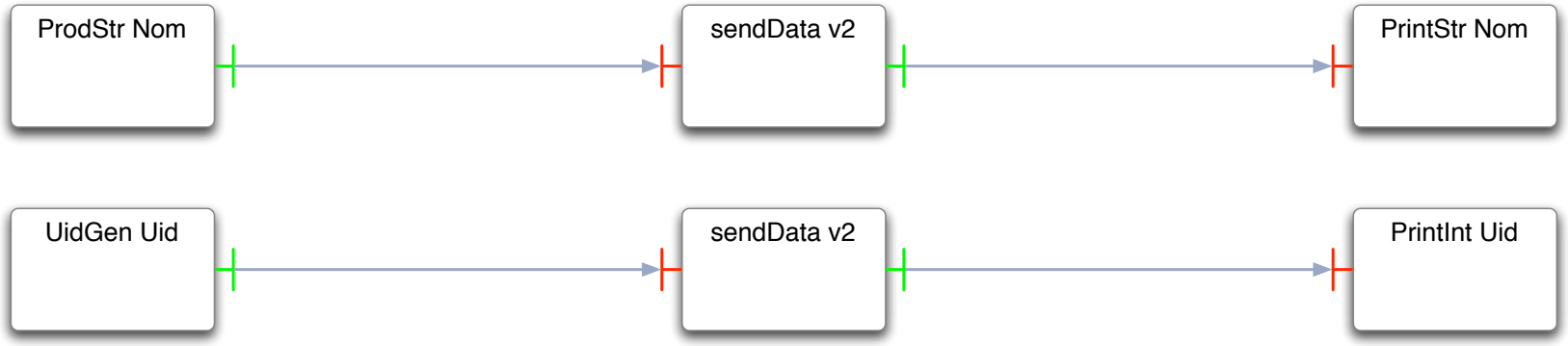
[*Type Inference for Records in a Natural Extension of ML*, Didier Rémy, 1993]

In mathematics you don't understand things. You just get used to them.

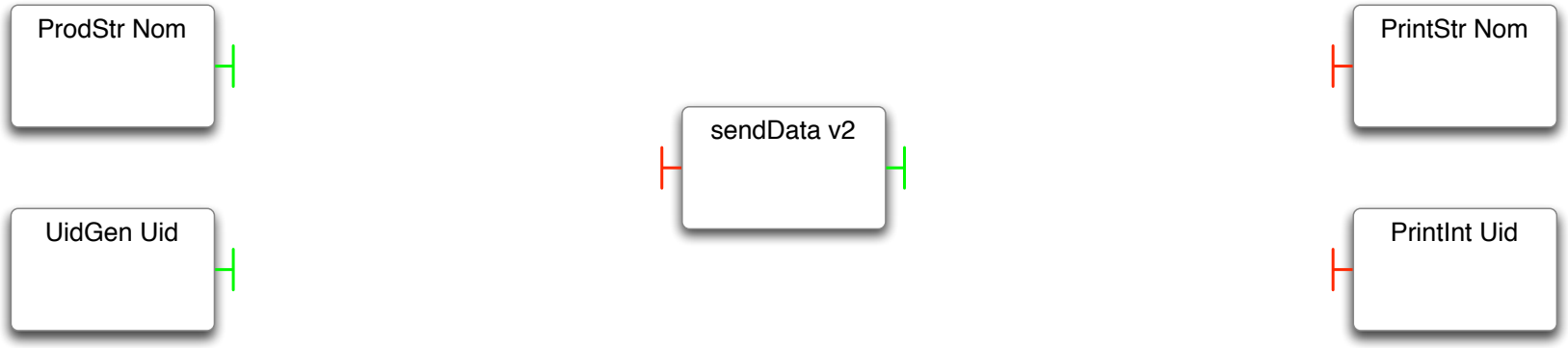
J. von Neumann

Multiplexer pour plus de modularité

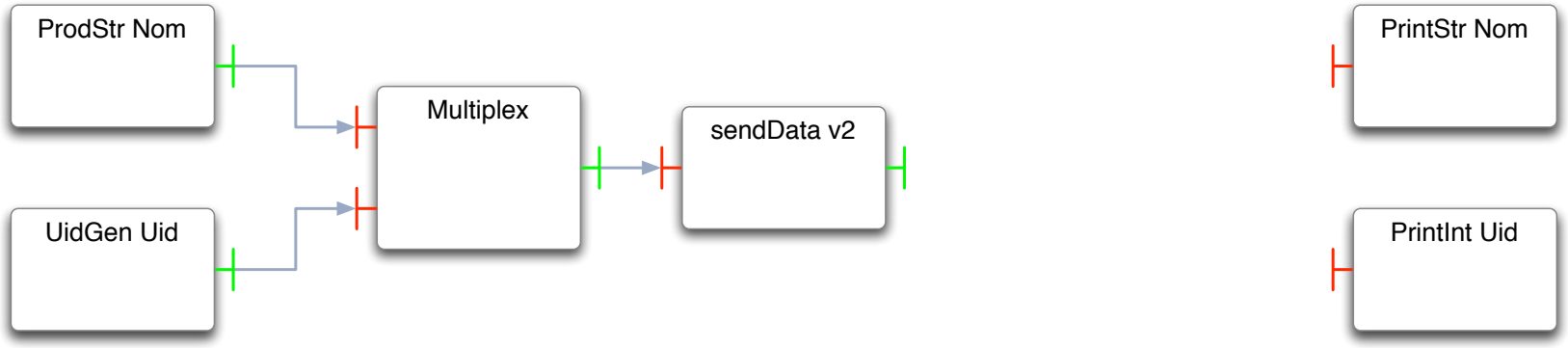
Multiplexer pour plus de modularité

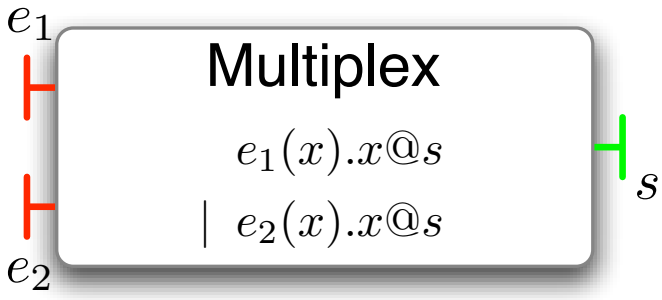


Multiplexer pour plus de modularité

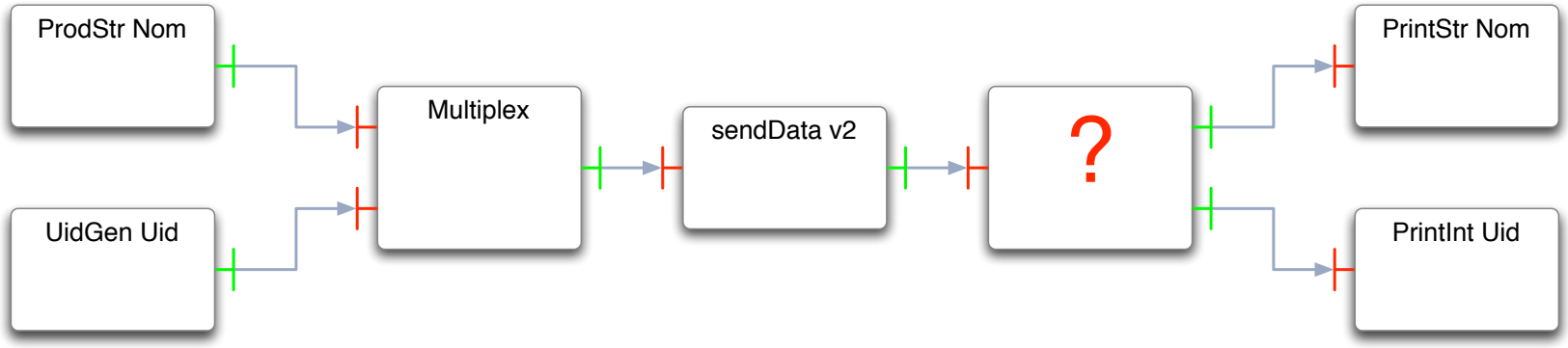


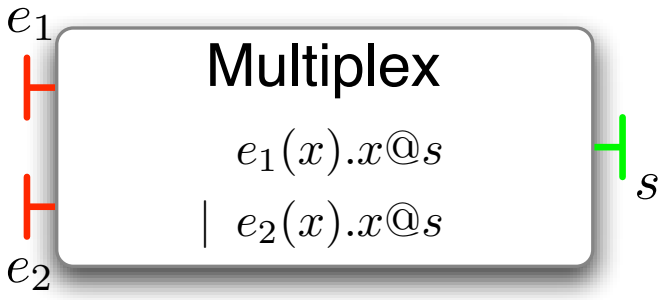
Multiplexer pour plus de modularité



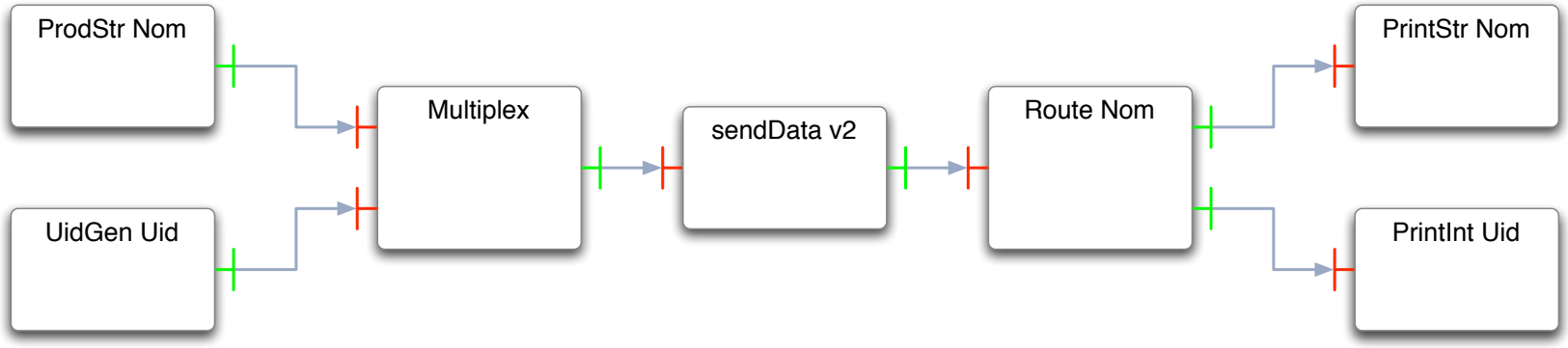


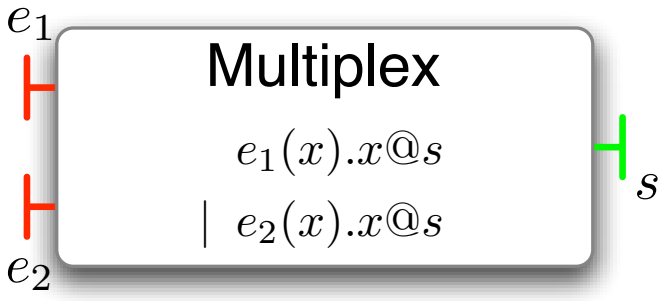
Multiplexer pour plus de modularité

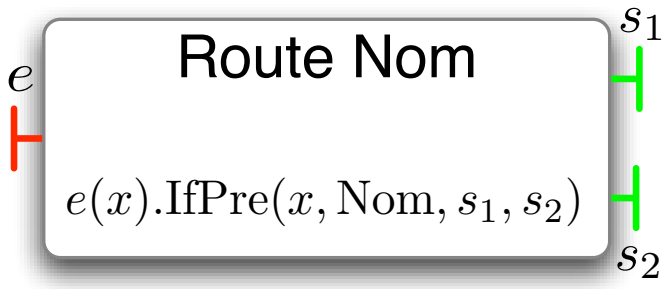


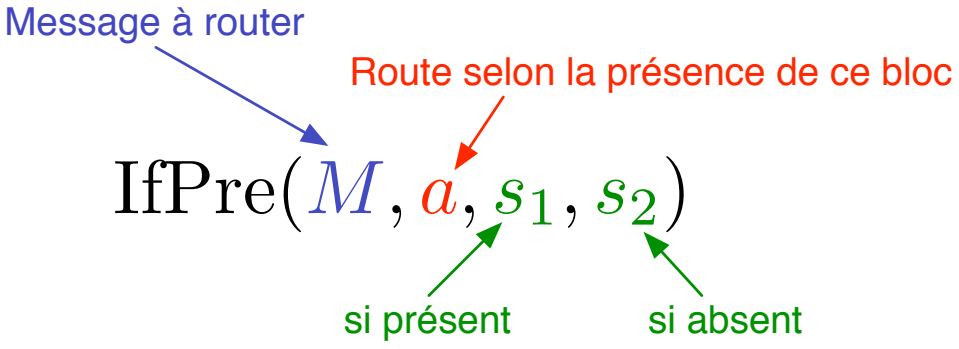


Multiplexer pour plus de modularité

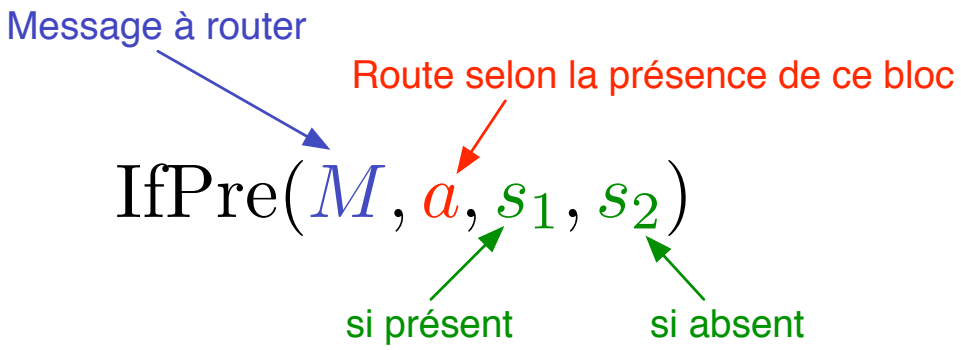








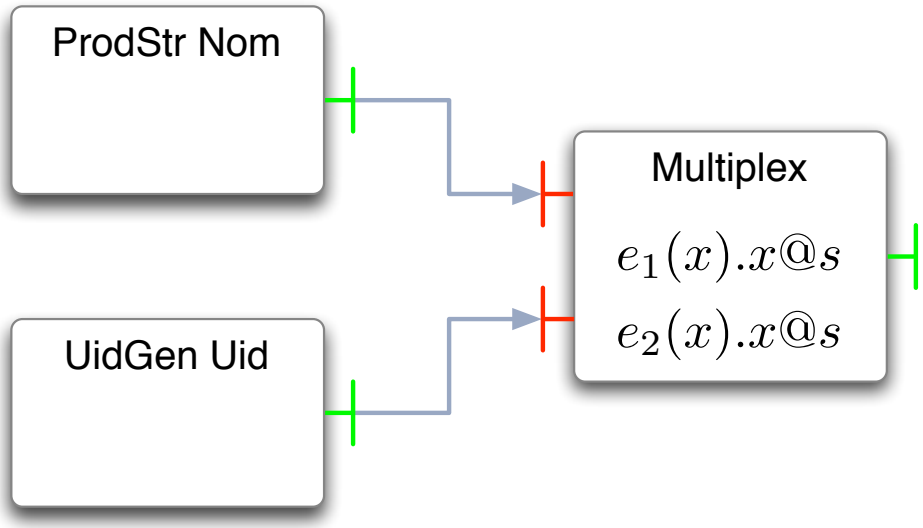
Modélisation du routeur



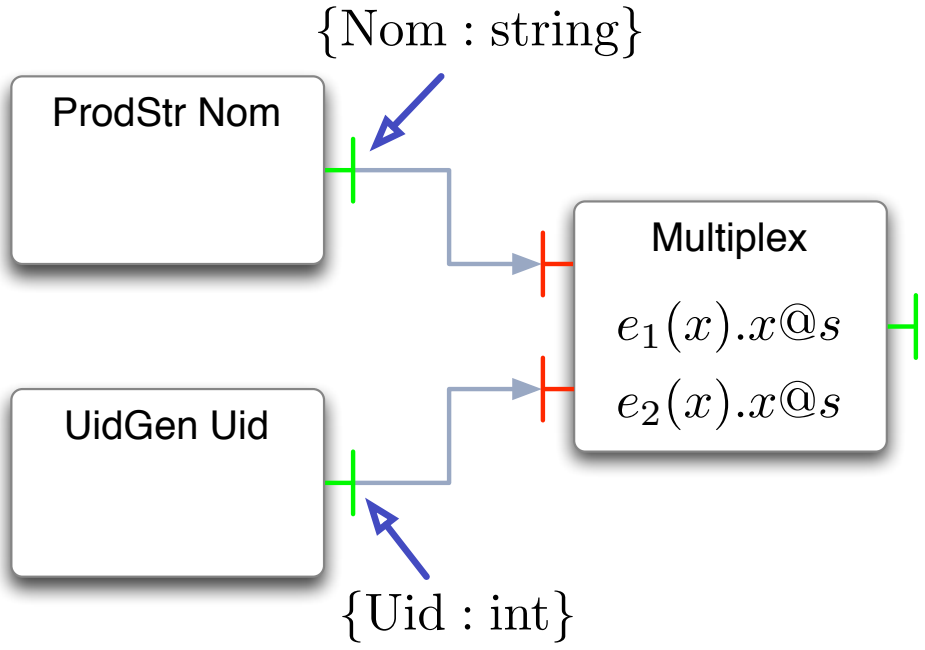
$$\mathrm{IfPre}(\{a_1 = v_1; \ldots; a_i = v_i; \ldots; a_n = v_n\}, a_i, s_1, s_2)$$
$$\longrightarrow \{a_1 = v_1; \ldots; a_i = v_i; \ldots; a_n = v_n\} @ s_1$$

$$\mathrm{IfPre}(\{a_1 = v_1; \ldots; a_n = v_n\}, a, s_1, s_2)$$
$$\longrightarrow \{a_1 = v_1; \ldots; a_n = v_n\} @ s_2 \qquad \text{si } a \notin \{a_1, \ldots, a_n\}$$

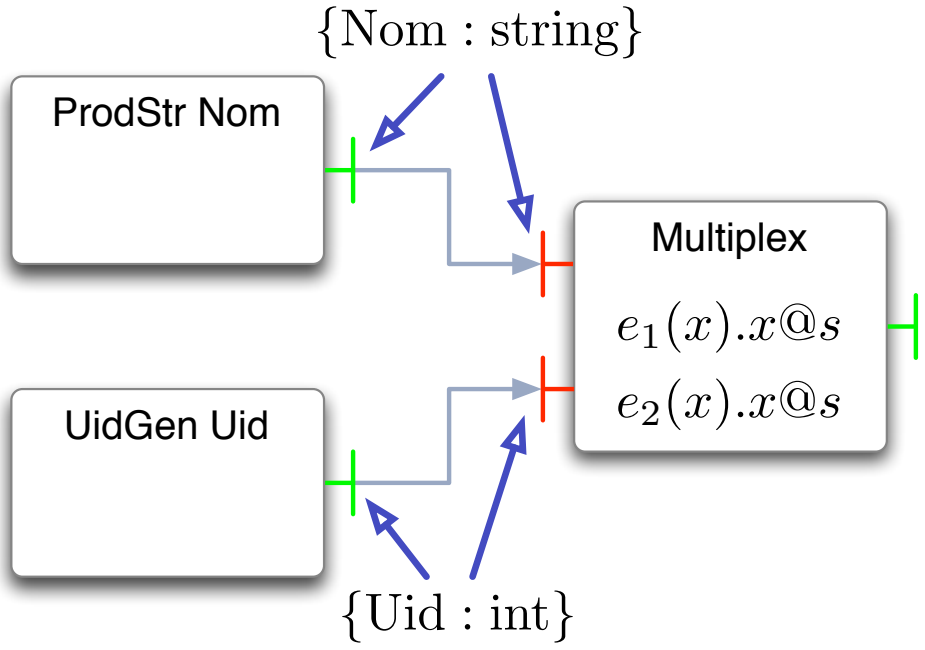
Typage du multiplexeur



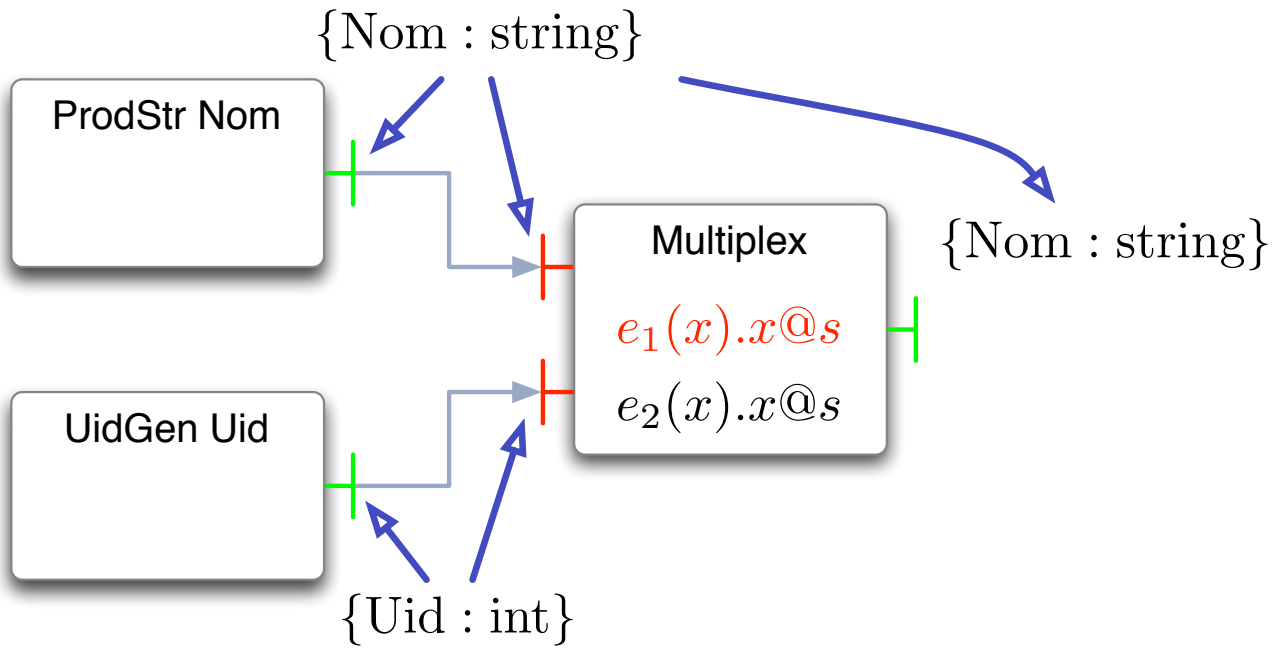
Typage du multiplexeur



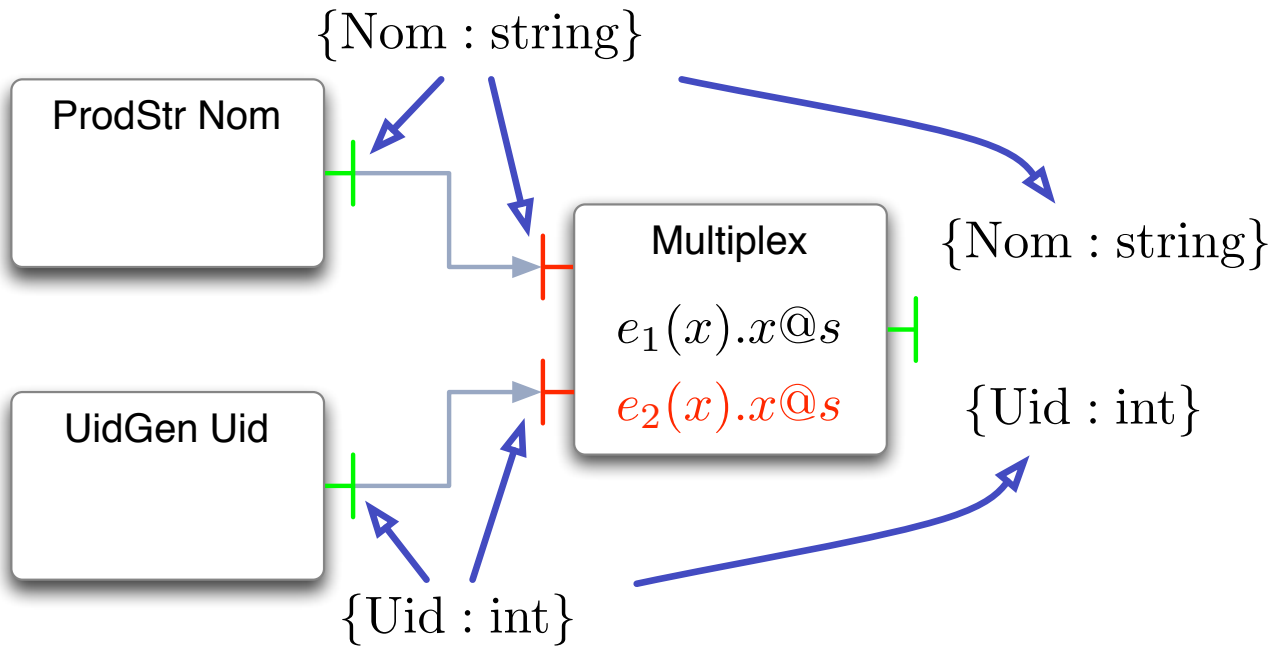
Typage du multiplexeur



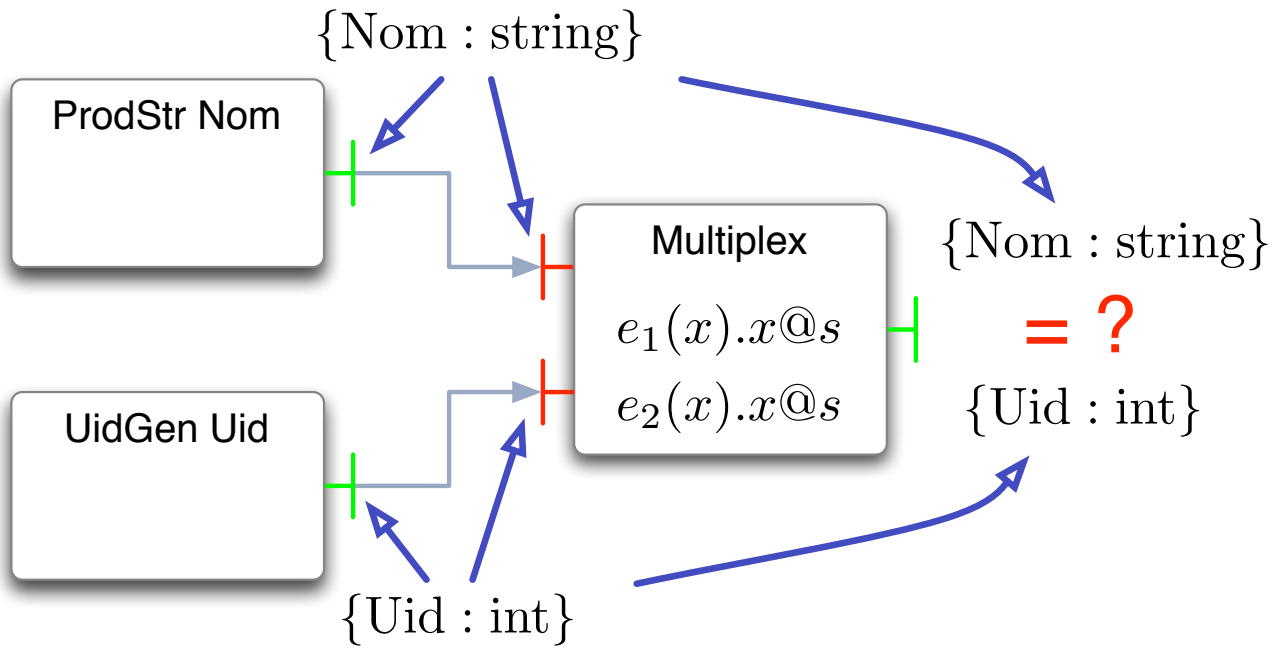
Typage du multiplexeur



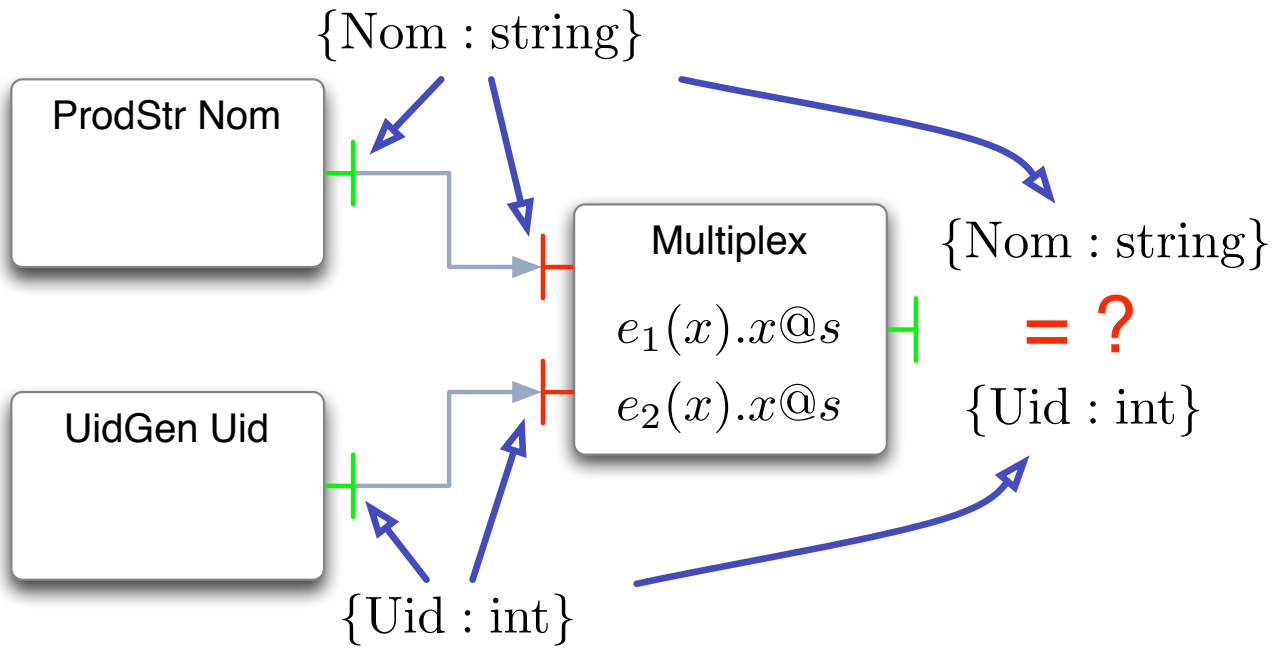
Typage du multiplexeur



Typage du multiplexeur



Typage du multiplexeur



$\Longrightarrow$ notion d'union dans les types: $\{\text{Nom} : \text{string}\} \cup \{\text{Uid} : \text{int}\}$

les types sont des ensembles

Vérifier des types avec des ensembles

$$e(x \qquad).((x - a)@s_1 \,|\, (x.a)@s_2)$$

Vérifier des types avec des ensembles

$$e(x : \{a : \text{int}; c : \text{string}\}).((x - a)@s_1 \mid (x.a)@s_2) : (s_1 : \{c : \text{string}\}) \cup (s_2 : \text{int})$$

Types donnés par l'utilisateur vérifiés par le système

$$x : \{a : \text{int}; c : \text{string}\}$$

Vérifier des types avec des ensembles

$$e(x : \{a : \text{int}; c : \text{string}\}).((x - a)@s_1 \mid (x.a)@s_2) : (s_1 : \{c : \text{string}\}) \cup (s_2 : \text{int})$$

Types donnés par l'utilisateur vérifiés par le système

$$x : \{a : \text{int}; c : \text{string}\}$$

$$x - a : \{c : \text{string}\}$$

Vérifier des types avec des ensembles

$$e(x : \{a : \text{int}; c : \text{string}\}).((x - a)@s_1 \mid (x.a)@s_2) : (s_1 : \{c : \text{string}\}) \cup (s_2 : \text{int})$$

Types donnés par l'utilisateur vérifiés par le système

$$x : \{a : \text{int}; c : \text{string}\}$$
$$x - a : \{c : \text{string}\}$$
$$(x - a)@s_1 : (s_1 : \{c : \text{string}\})$$

Vérifier des types avec des ensembles

$$e(x : \{a : \text{int}; c : \text{string}\}).((x - a)@s_1 \mid (x.a)@s_2) : (s_1 : \{c : \text{string}\}) \cup (s_2 : \text{int})$$

Types donnés par l'utilisateur vérifiés par le système

$$\begin{array}{c} x : \{a : \text{int}; c : \text{string}\} \\ x - a : \{c : \text{string}\} \qquad x.a : \text{int} \\ (x - a)@s_1 : (s_1 : \{c : \text{string}\}) \quad (x.a)@s_2 : (s_2 : \text{int}) \end{array}$$

Vérifier des types avec des ensembles

$$e(x : \{a : \text{int}; c : \text{string}\}).((x - a)@s_1 \mid (x.a)@s_2) : (s_1 : \{c : \text{string}\}) \cup (s_2 : \text{int})$$

Types donnés par l'utilisateur vérifiés par le système

$$x : \{a : \text{int}; c : \text{string}\}$$

$$x - a : \{c : \text{string}\} \qquad x.a : \text{int}$$

$$(x - a)@s_1 : (s_1 : \{c : \text{string}\}) \qquad (x.a)@s_2 : (s_2 : \text{int})$$

$$(s_1 : \{c : \text{string}\}) \cup (s_2 : \text{int}) \subseteq (s_1 : \{c : \text{string}\}) \cup (s_2 : \text{int})$$

Bien Typé

Typer IfPre

$$e(x : \{a : \text{int}; c : \text{string}\} \cup \{b : \text{float}\} \cup \{a : \text{int}\}).($$
$$\text{IfPre}(x, a, s_1, s_2)$$
$$) : (s_1 : \{a : \text{int}; c : \text{string}\} \cup s_2 : \{b : \text{float}\})$$

Typer IfPre

$$e(x : \{a : \text{int}; c : \text{string}\} \cup \{b : \text{float}\} \cup \{a : \text{int}\}).($$
$$\text{IfPre}(x, a, s_1, s_2)$$
$$) : (s_1 : \{a : \text{int}; c : \text{string}\} \cup s_2 : \{b : \text{float}\})$$

$$s_1 : \{a : \text{int}; c : \text{string}\}$$

Typer IfPre

$$e(x : \{a : \text{int}; c : \text{string}\} \cup \{b : \text{float}\} \cup \{a : \text{int}\}).(\\ \text{IfPre}(x, a, s_1, s_2)\\ ) : (s_1 : \{a : \text{int}; c : \text{string}\} \cup s_2 : \{b : \text{float}\})$$

$$s_1 : \{a : \text{int}; c : \text{string}\}$$

$$s_2 : \{b : \text{float}\}$$

Typer IfPre

$$e(x : \{a : \text{int}; c : \text{string}\} \cup \{b : \text{float}\} \cup \{a : \text{int}\}).(\\ \text{IfPre}(x, a, s_1, s_2)\\ ) : (s_1 : \{a : \text{int}; c : \text{string}\} \cup s_2 : \{b : \text{float}\})$$

$$s_1 : \{a : \text{int}; c : \text{string}\} \cup \{a : \text{int}\}$$

$$s_2 : \{b : \text{float}\}$$

Typer IfPre

$$e(x : \{a : \text{int}; c : \text{string}\} \cup \{b : \text{float}\} \cup \{a : \text{int}\}).($$
$$\text{IfPre}(x, a, s_1, s_2)$$
$$) : (s_1 : \{a : \text{int}; c : \text{string}\} \cup s_2 : \{b : \text{float}\})$$

$$s_1 : \{a : \text{int}; c : \text{string}\} \cup \{a : \text{int}\}$$
$$s_2 : \{b : \text{float}\}$$

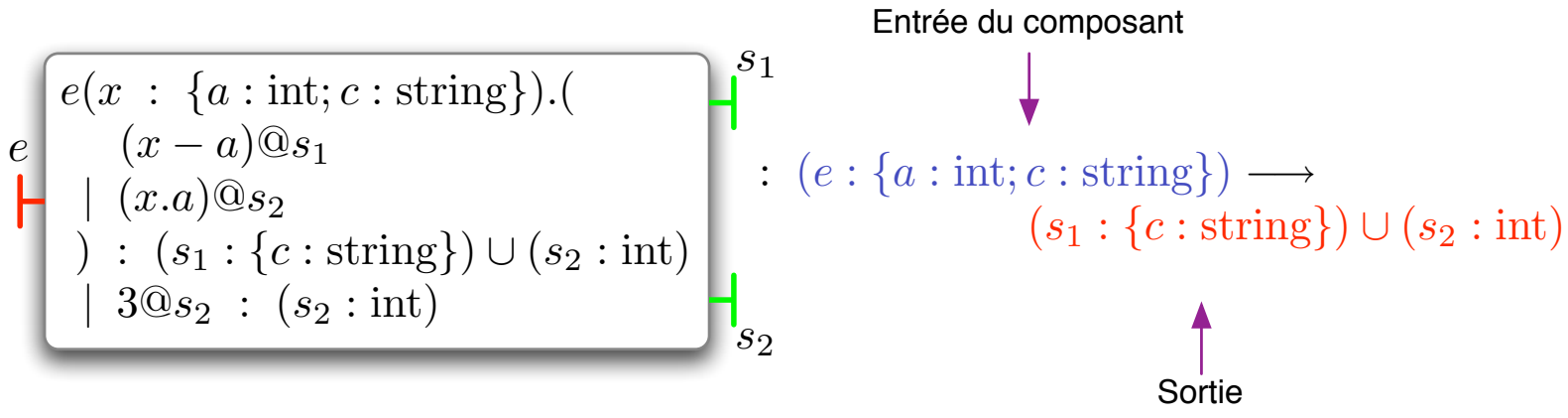
$$s_1 : (\{a : \text{int}; c : \text{string}\} \cup \{a : \text{int}\}) \cup s_2 : \{b : \text{float}\}$$
$$\not\subseteq s_1 : \{a : \text{int}; c : \text{string}\} \cup s_2 : \{b : \text{float}\}$$

Mal Typé

Vérification des annotations

Entrées et sorties annotées par les types

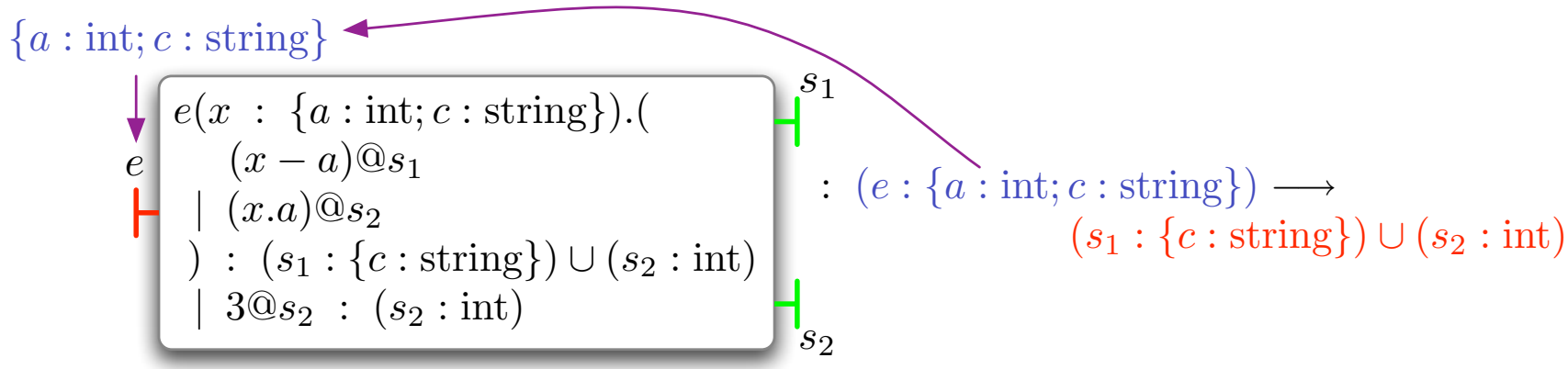
Le système vérifie la cohérence (inclusion) des annotations



Vérification des annotations

Entrées et sorties annotées par les types

Le système vérifie la cohérence (inclusion) des annotations

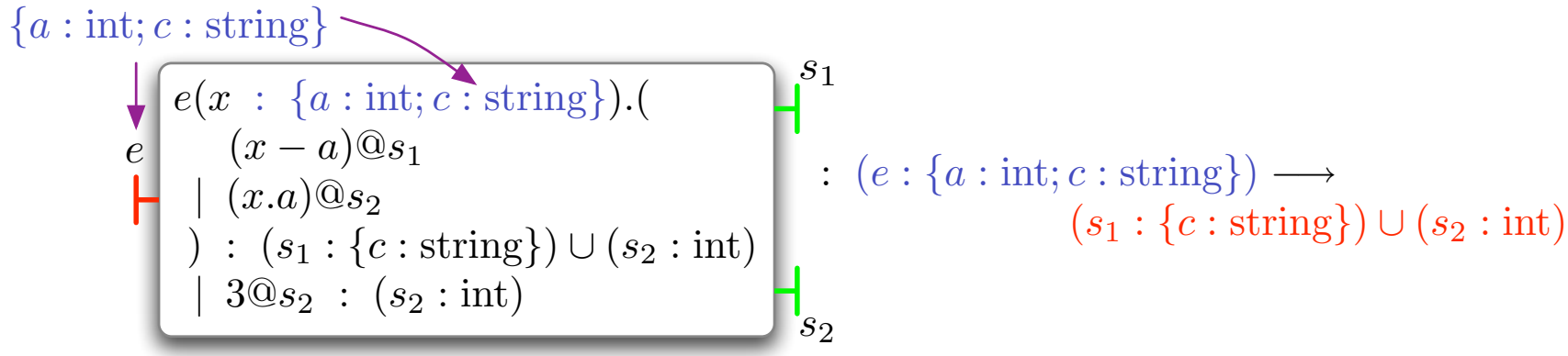


Le type de la boîte définit son type d'entrée

Vérification des annotations

Entrées et sorties annotées par les types

Le système vérifie la cohérence (inclusion) des annotations

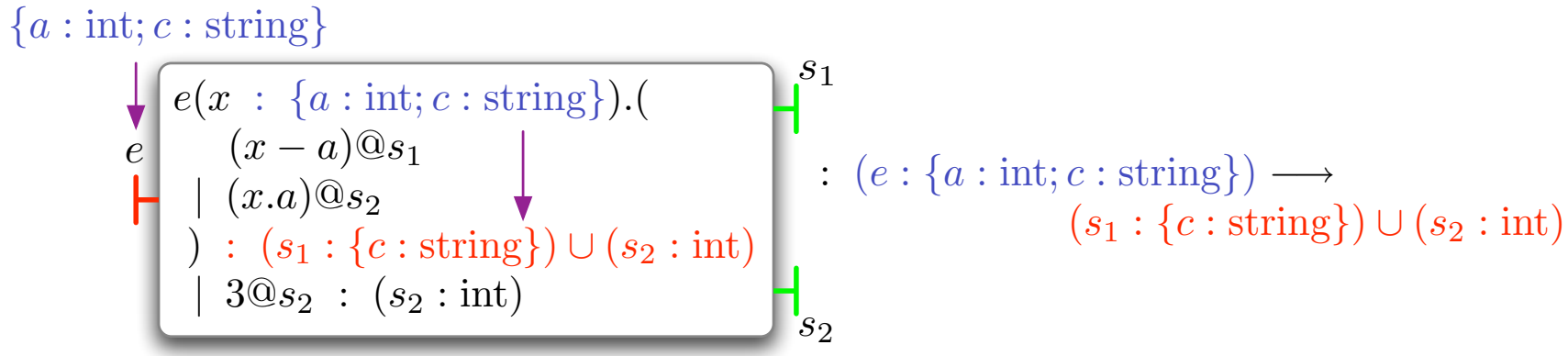


Le type d'entrée définie l'entrée du programme

Vérification des annotations

Entrées et sorties annotées par les types

Le système vérifie la cohérence (inclusion) des annotations

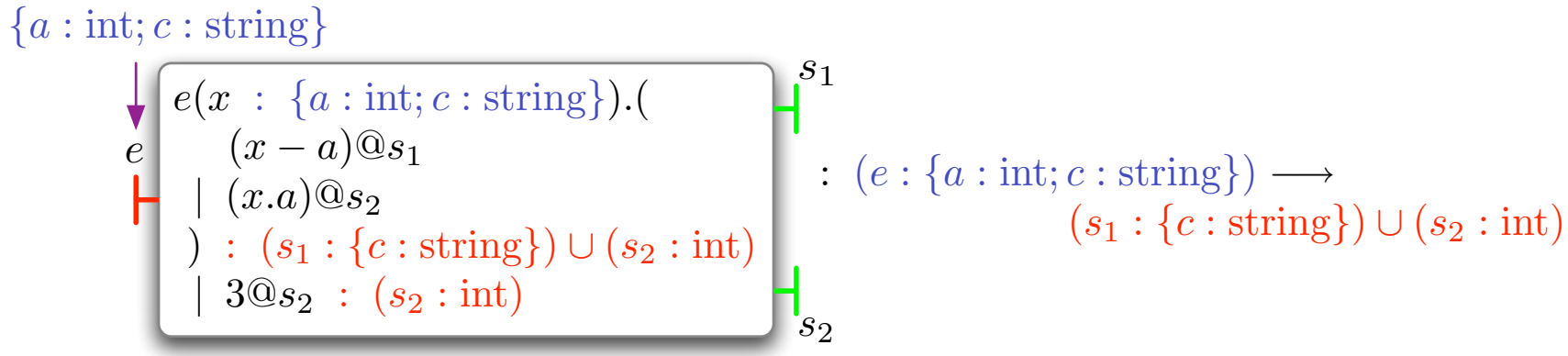


Le type de sortie du programme est calculé

Vérification des annotations

Entrées et sorties annotées par les types

Le système vérifie la cohérence (inclusion) des annotations

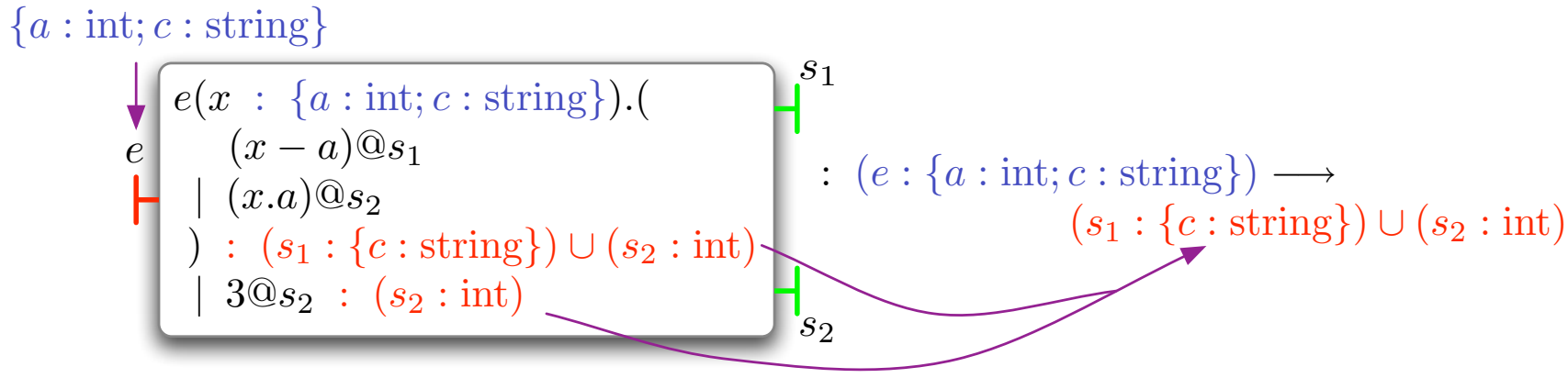


Les messages existants ont aussi un type

Vérification des annotations

Entrées et sorties annotées par les types

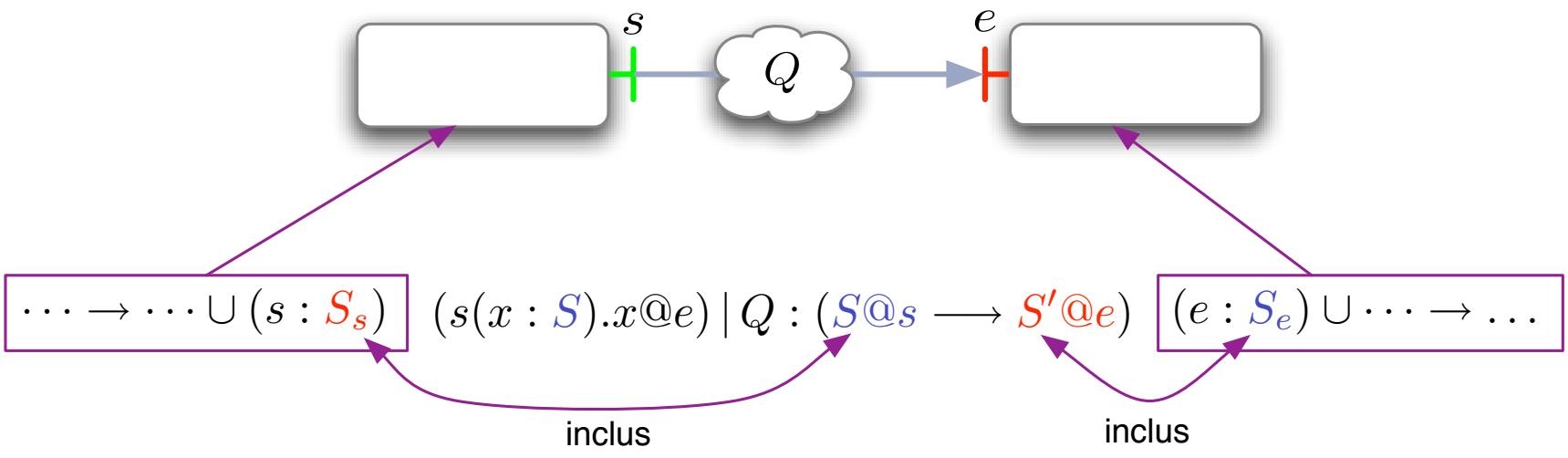
Le système vérifie la cohérence (inclusion) des annotations



L'inclusion dans le type sortie de la boîte est vérifiée

Typage d'une liaison

$$(s(x : \mathsf{int}).x@e) \mid \{a = 3.9\}@e \mid \{c = `t'\}@e :$$
$$\mathsf{int}@s \longrightarrow (\mathsf{int} \cup \{a : \mathsf{float}\} \cup \{c : \mathsf{char}\})@e$$



Correction

Définition

Une *erreur* survient dans un programme Dream quand un opérateur est appliqué à un mauvais argument.

Définition

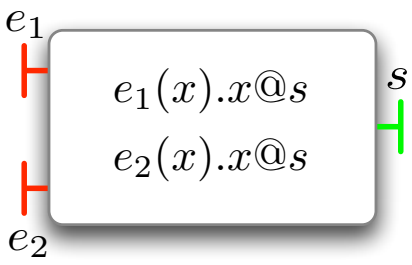
Un *'bon' système de types* est correct :

1. Aucun programme typé n'a d'erreur.
2. Tout programme typé se réduit en un programme typé.

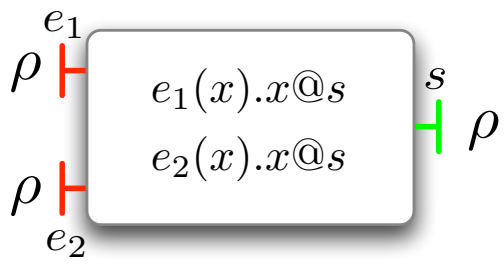
Théorème

Notre système de type est correct.

Vers l'inférence ?



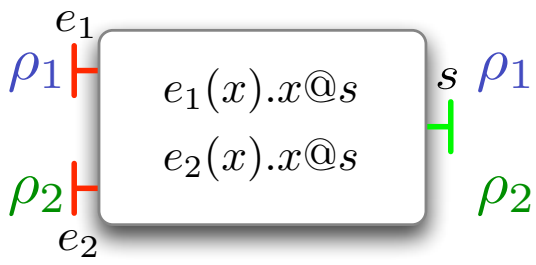
Vers l'inférence ?



Unification des types des entrées

$$\forall \rho.(\rho @ e_1 \rightarrow \rho @ s) \,|\, (\rho @ e_2 \rightarrow \rho @ s)$$

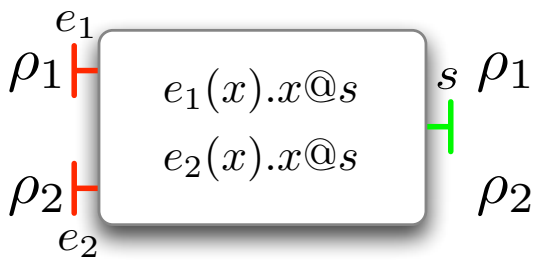
Vers l'inférence ?

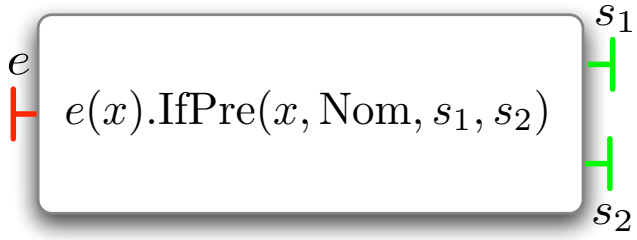


Séparation des flots

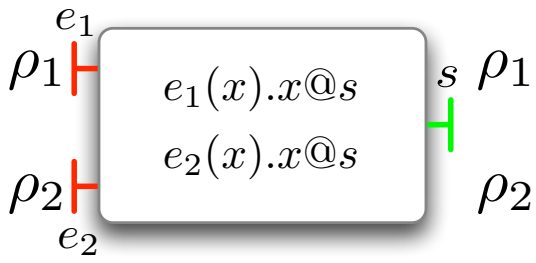
$$\forall \rho_1, \rho_2.(\rho_1 @ e_1 \longrightarrow \rho_1 @ s) \mid (\rho_2 @ e_2 \longrightarrow \rho_2 @ s)$$

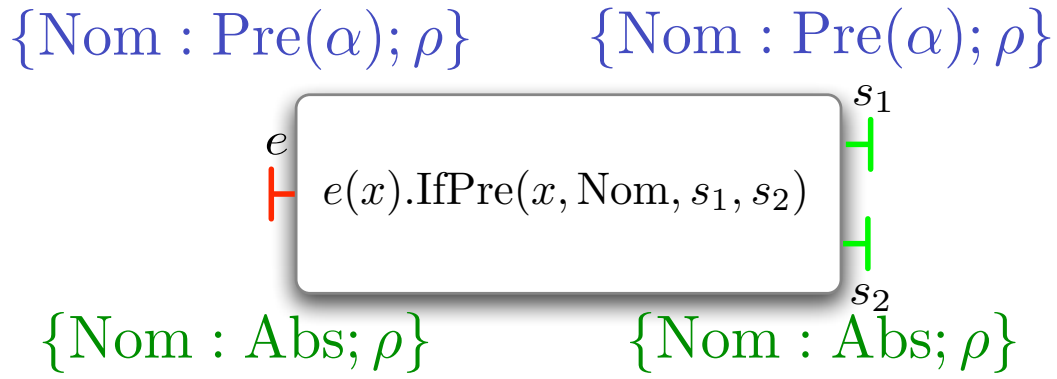
Vers l'inférence ?





Vers l'inférence ?

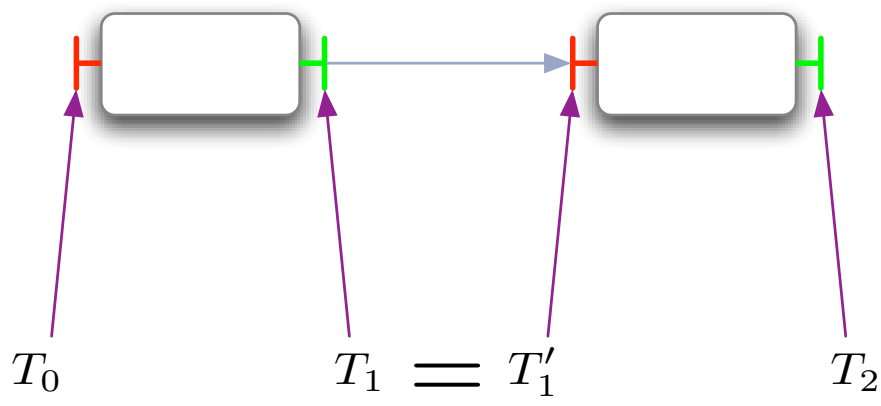




$$\forall \alpha, \rho.(\ \{\text{Nom} : \text{Pre}(\alpha); \rho\}\ @e \rightarrow \{\text{Nom} : \text{Pre}(\alpha); \rho\}\ @s_1)$$
$$|\quad (\{\text{Nom} : \text{Abs}; \rho\} @e \rightarrow \{\text{Nom} : \text{Abs}; \rho\} @s_2)$$

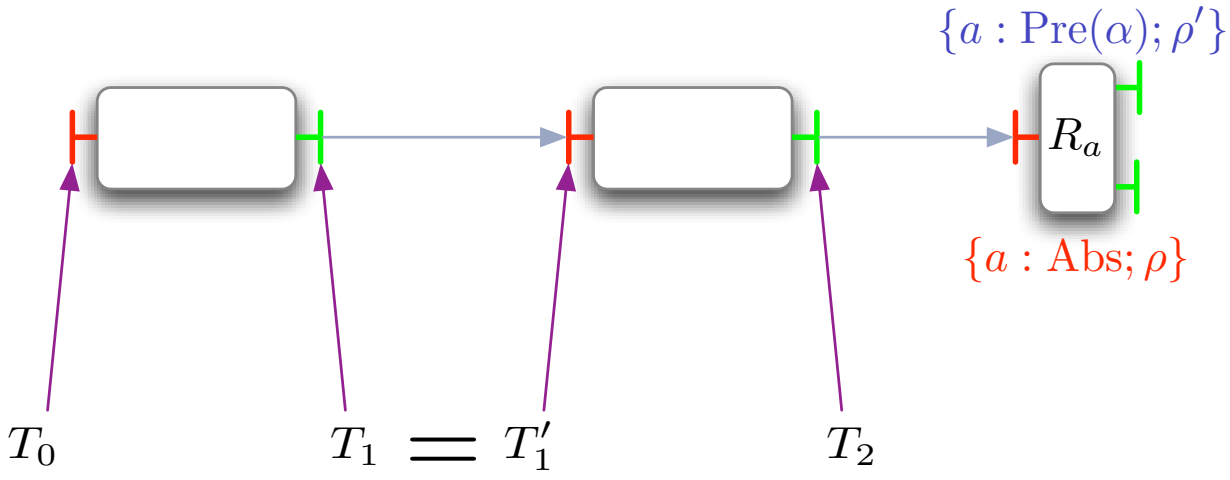
Problématique de l'inférence

Une approche par flots ?



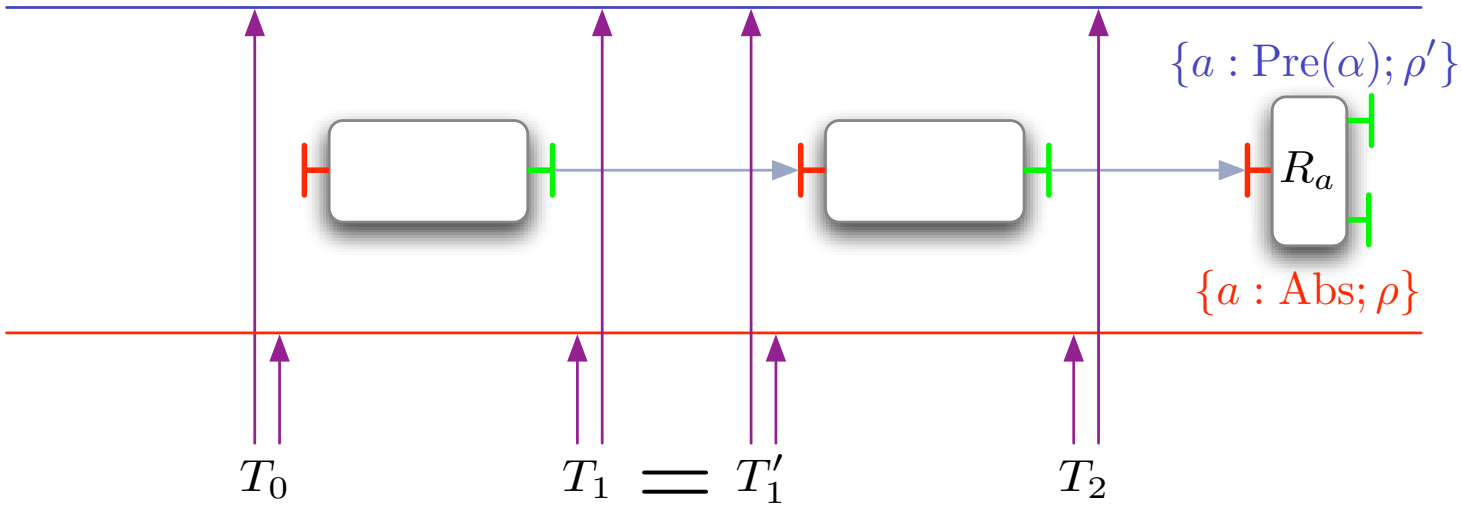
Problématique de l'inférence

Une approche par flots ?



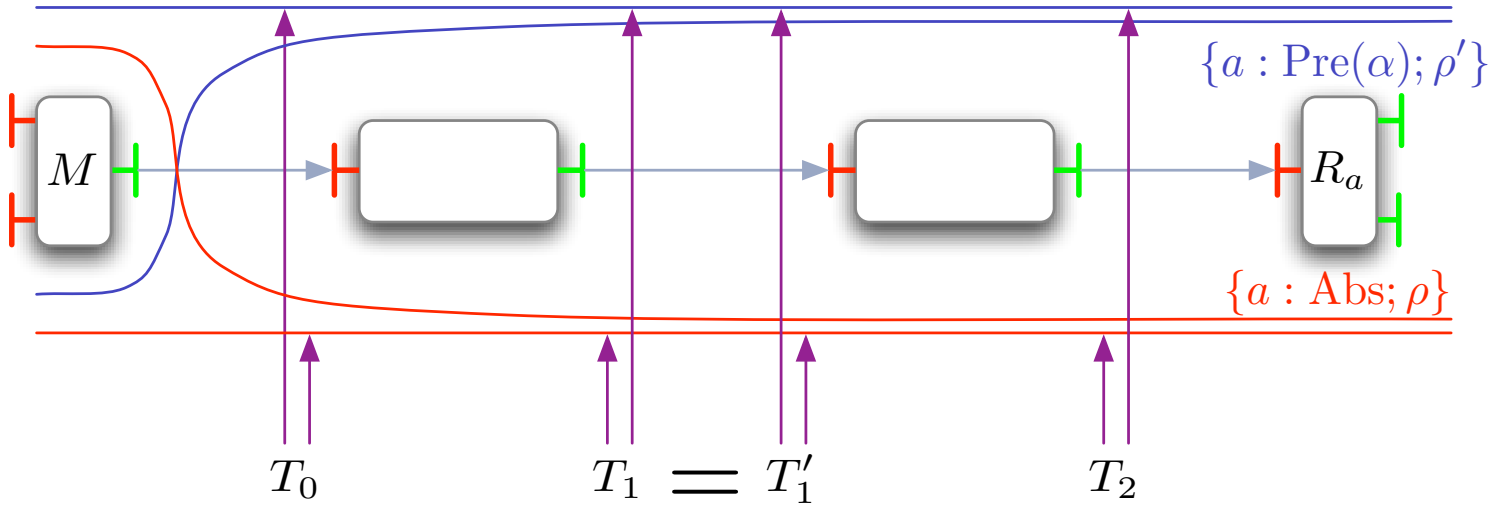
Problématique de l'inférence

Une approche par flots ?



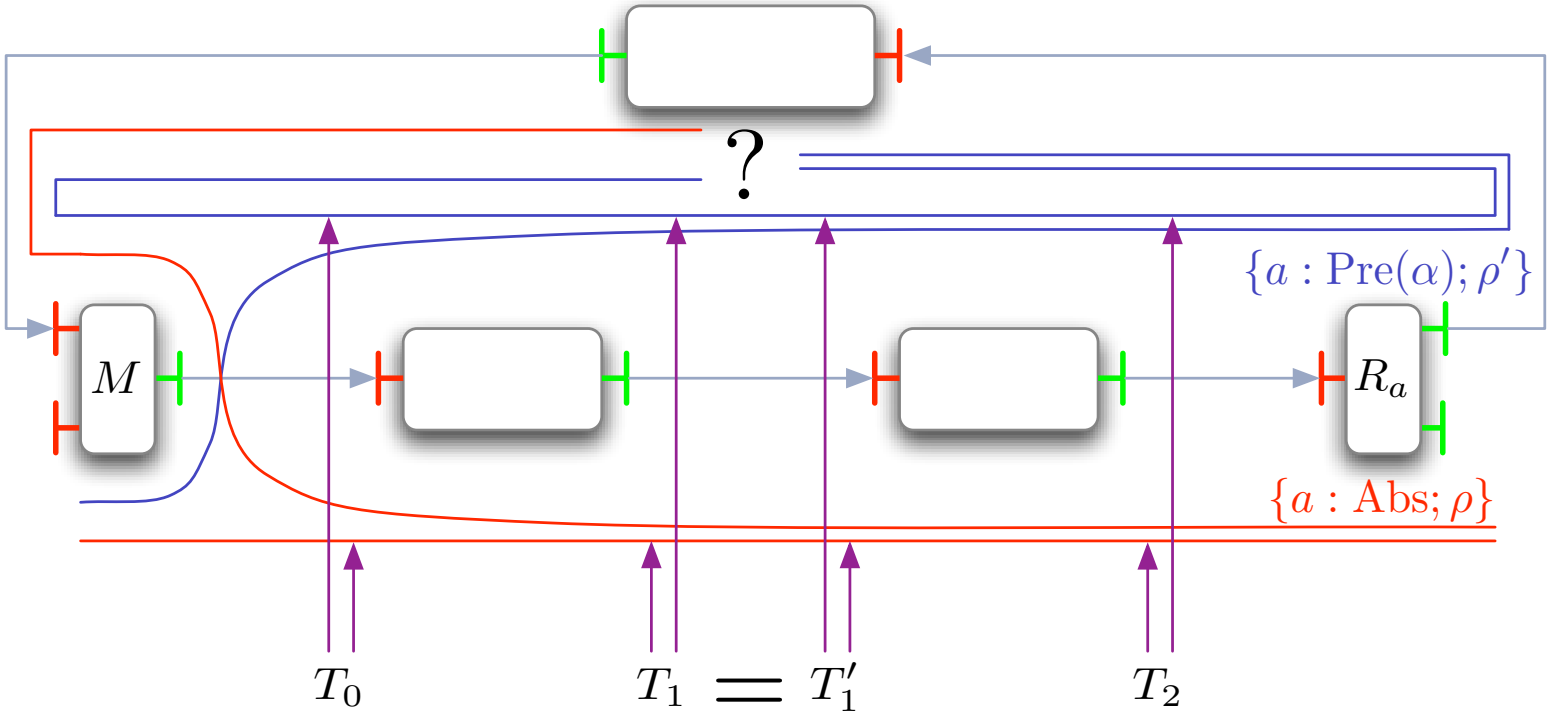
Problématique de l'inférence

Une approche par flots ?



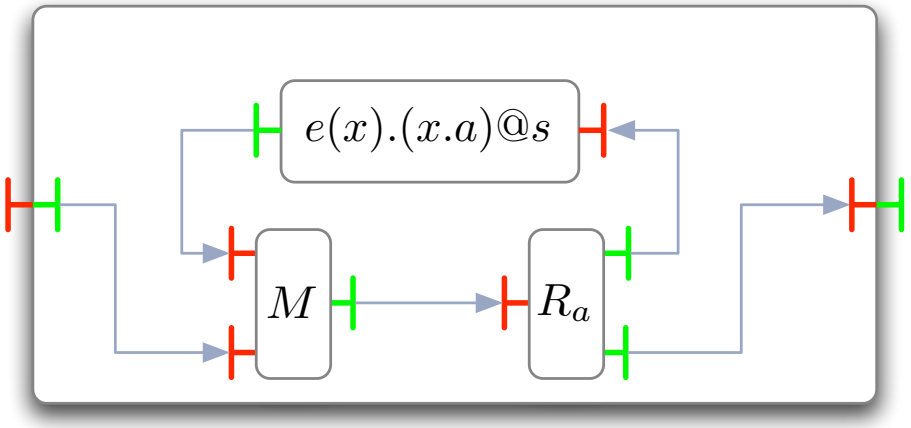
Problématique de l'inférence

Une approche par flots ?

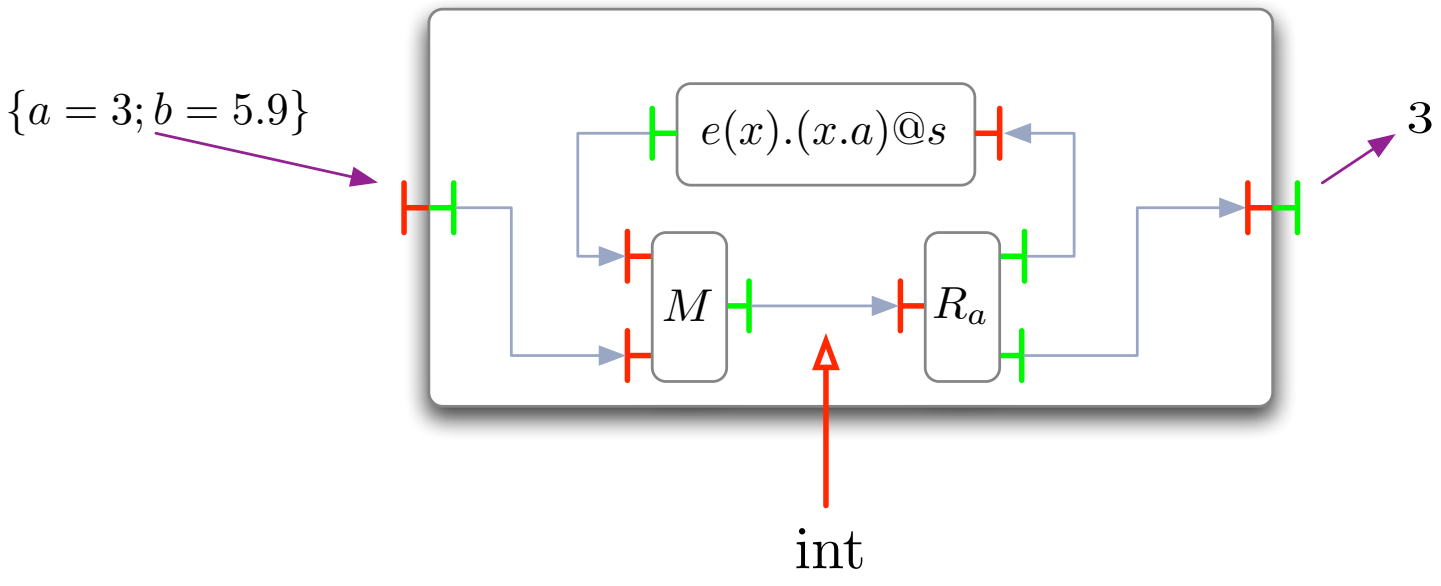


problème de la récursion polymorphe ?

Boucles et Types

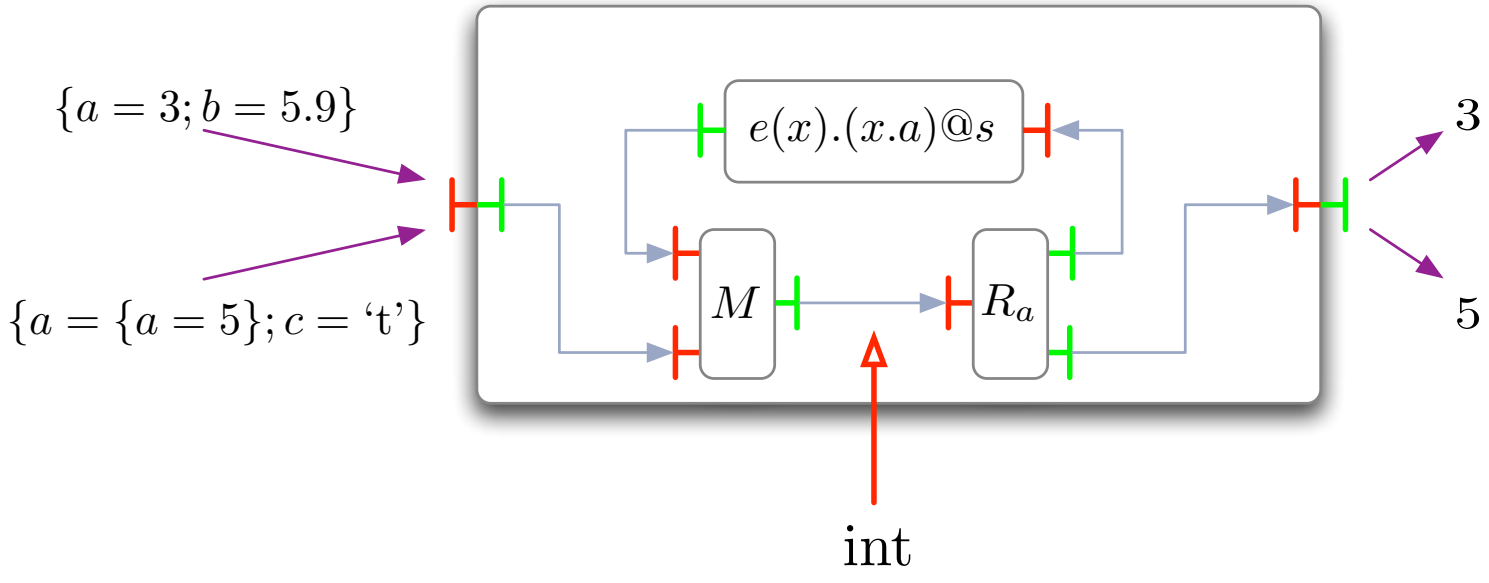


Boucles et Types



$\{a : \mathrm{Pre}(\mathrm{int}); b : \mathrm{Pre}(\mathrm{float})\}$

Boucles et Types



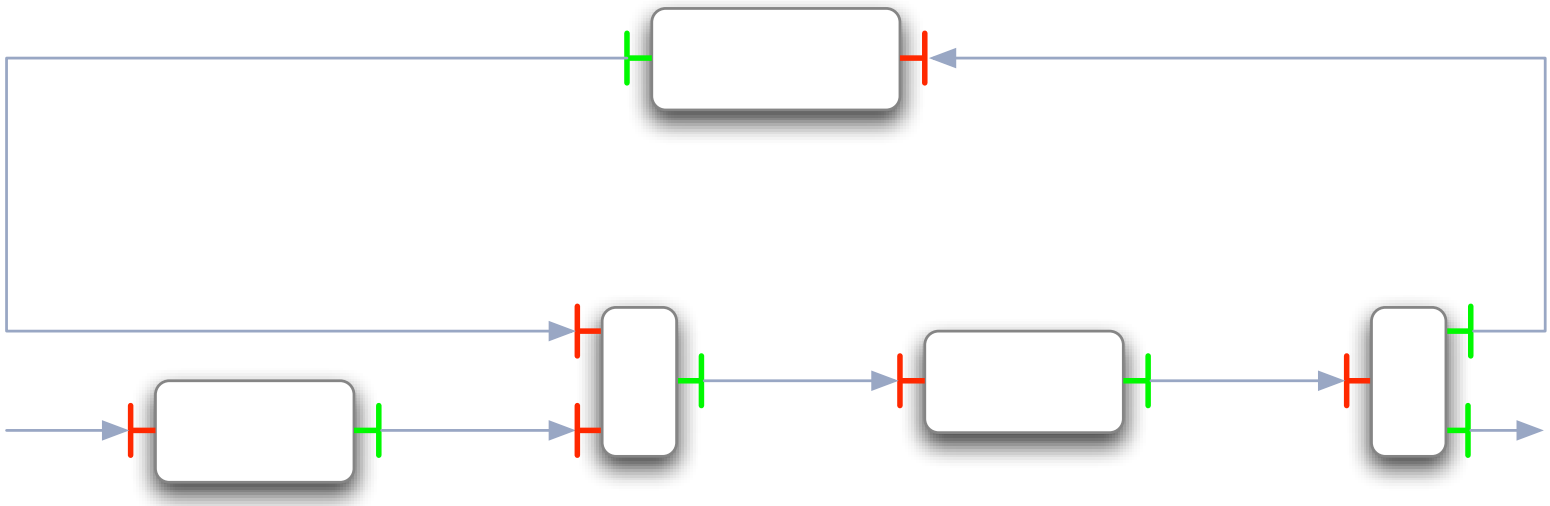
$$\{a : \text{Pre}(\text{int}); b : \text{Pre}(\text{float})\}$$

$$\{a : \text{Pre}(\{a : \text{Pre}(\text{int})\}); c : \text{Pre}(\text{char})\}$$

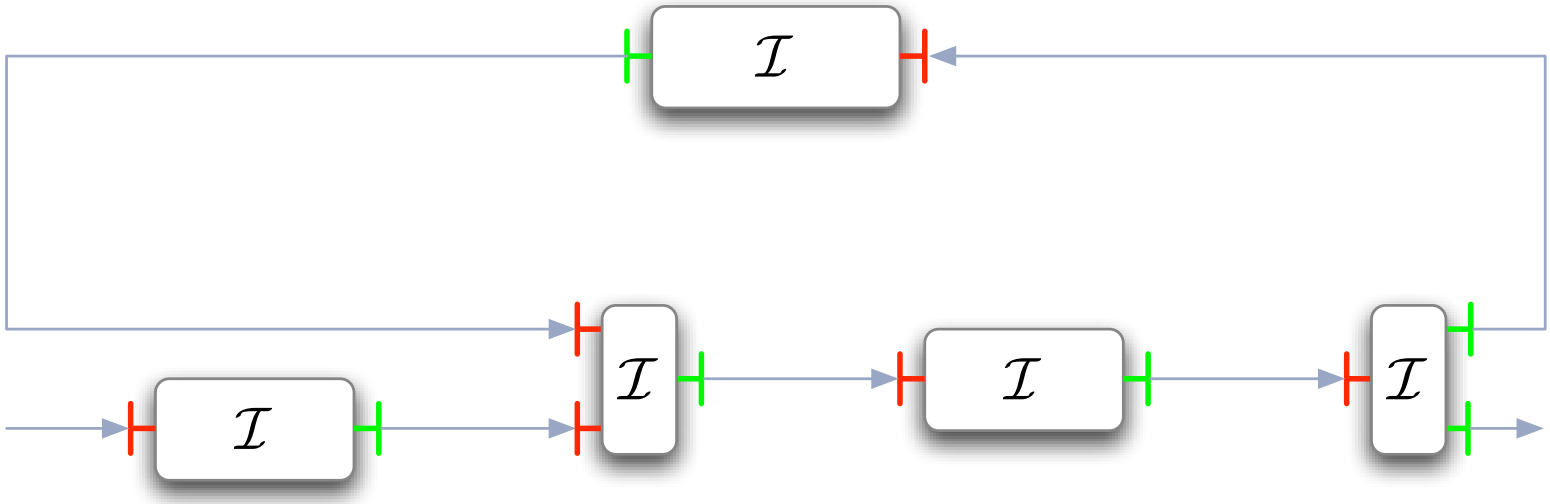
$$\forall \alpha, \rho_1, \ldots, \rho_n.$$

$$\{a : \text{Pre}(\{a : \text{Pre}(\ldots\{a : \text{Pre}(\alpha); \rho_n\}\ldots); \rho_2\}); \rho_1\} \rightarrow \alpha$$

Vers une approche concrète

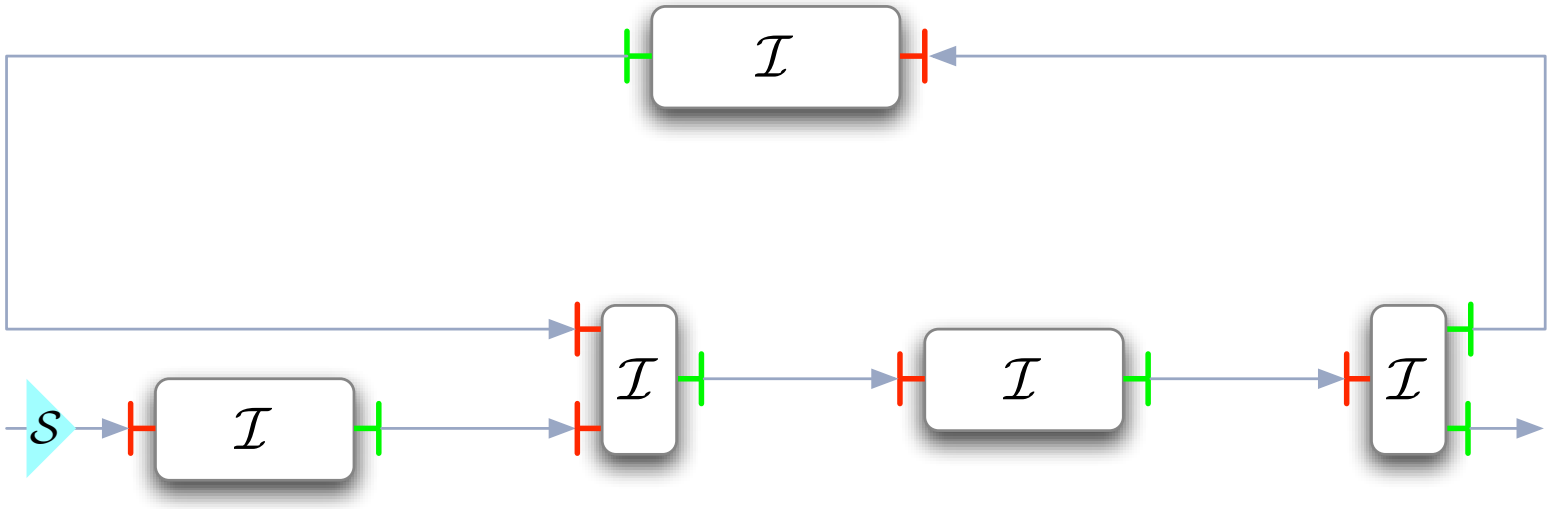


Vers une approche concrète



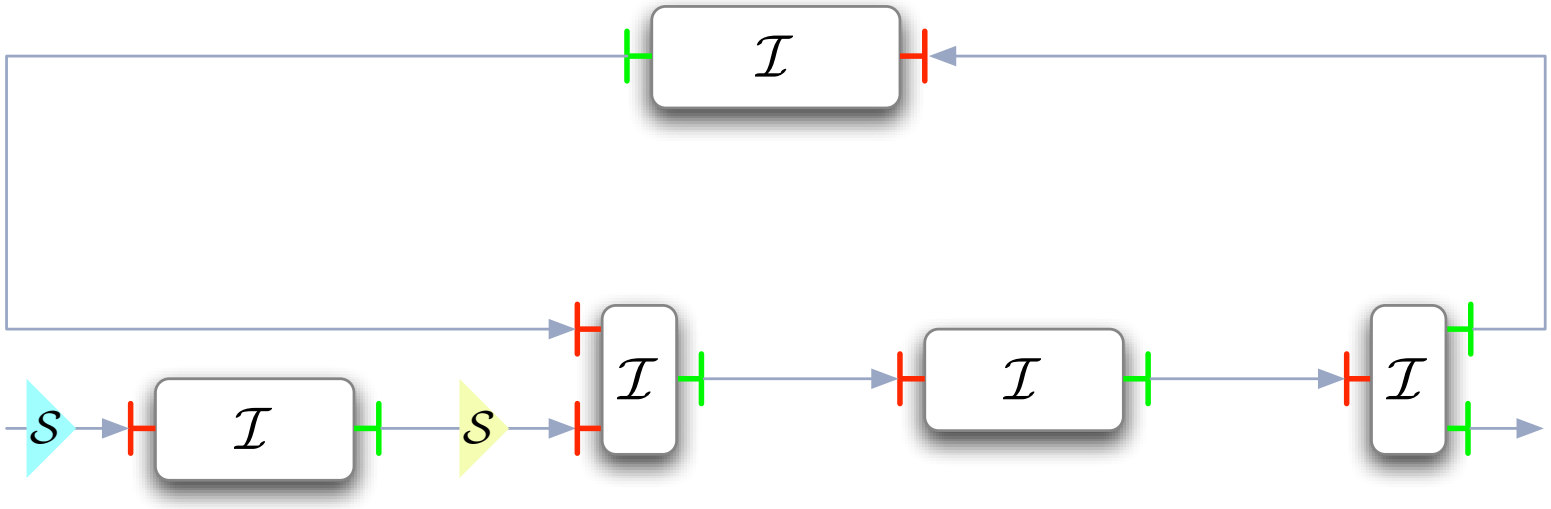
- Types des boîtes de base inférés $\mathcal{I}$

Vers une approche concrète



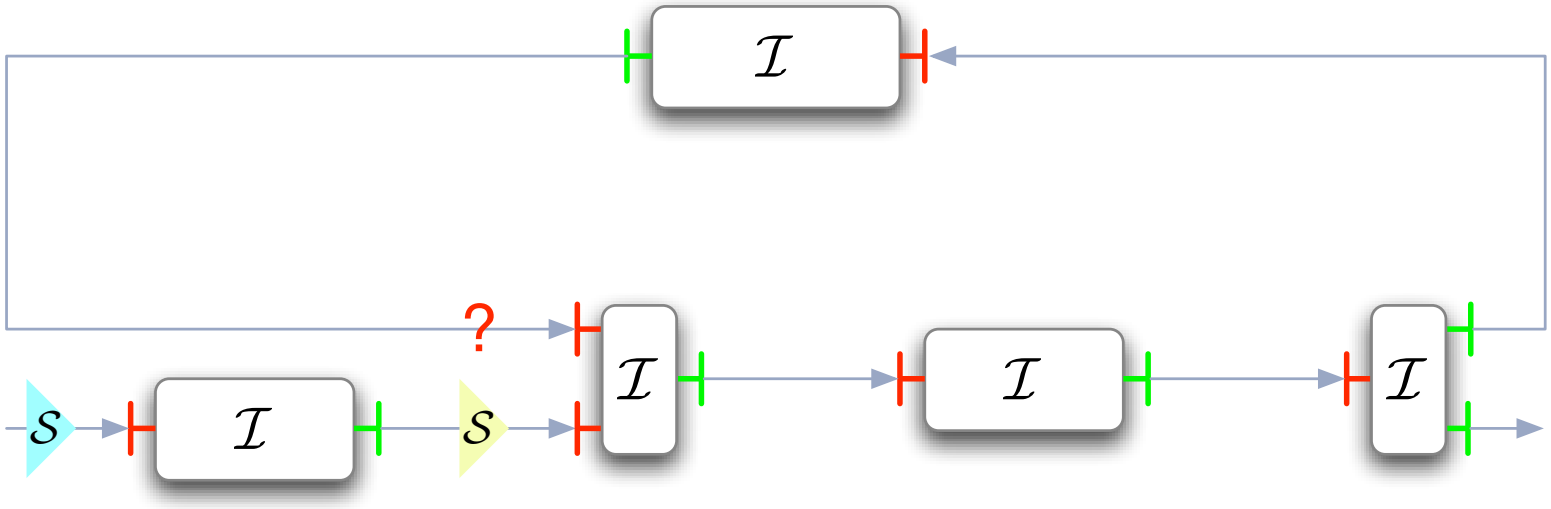
- Types des boîtes de base inférés $\mathcal{I}$
- Annotations $\mathcal{S}$ (types) en entrée

Vers une approche concrète



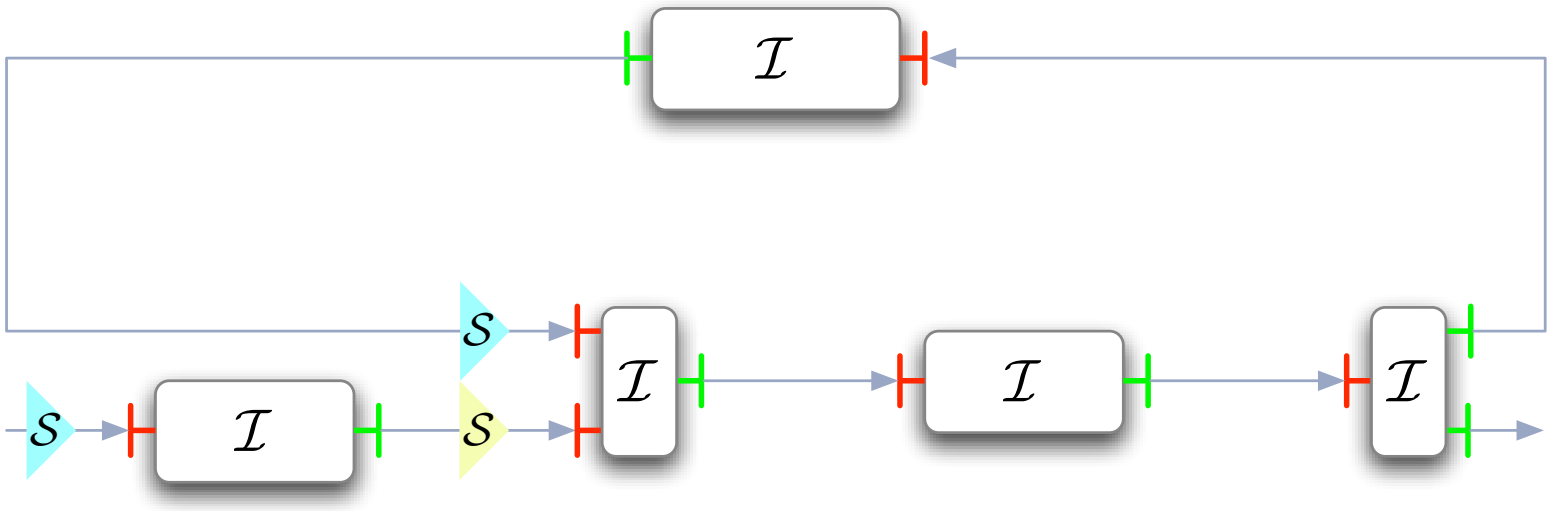
- Types des boîtes de base inférés $\mathcal{I}$
- Annotations $\mathcal{S}$ (types) en entrée
- Propagation (inférence) des annotations

Vers une approche concrète



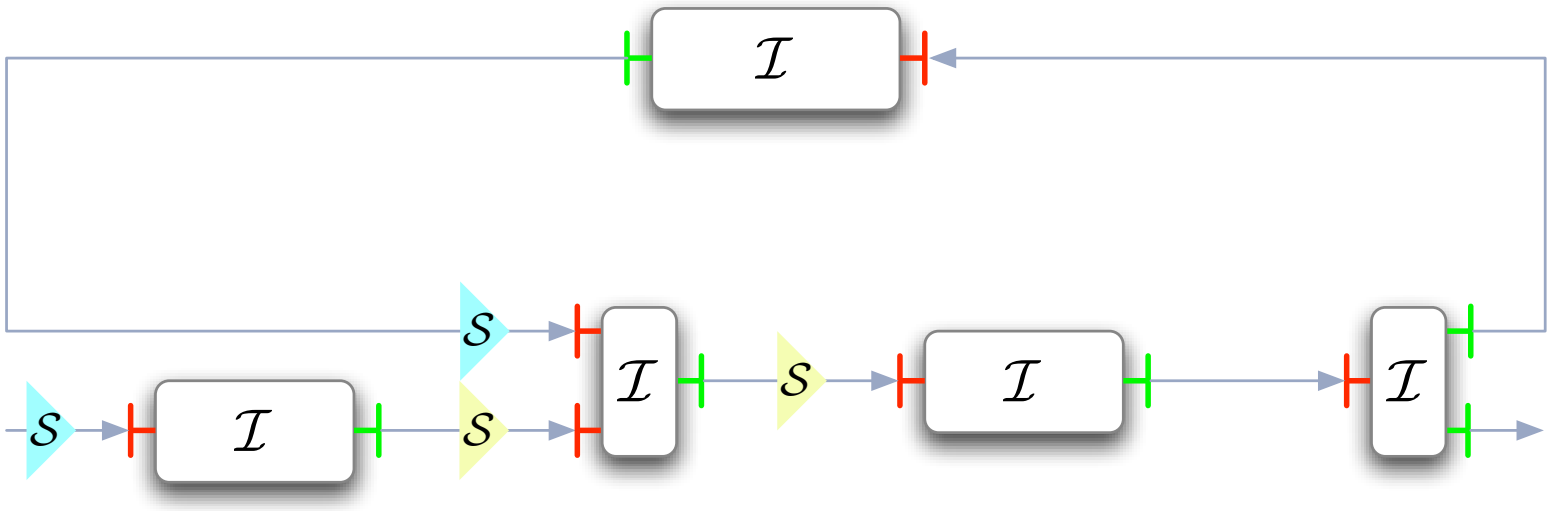
- Types des boîtes de base inférés $\mathcal{I}$
- Annotations $\mathcal{S}$ (types) en entrée
- Propagation (inférence) des annotations
- Détection du manque d'annotations

Vers une approche concrète



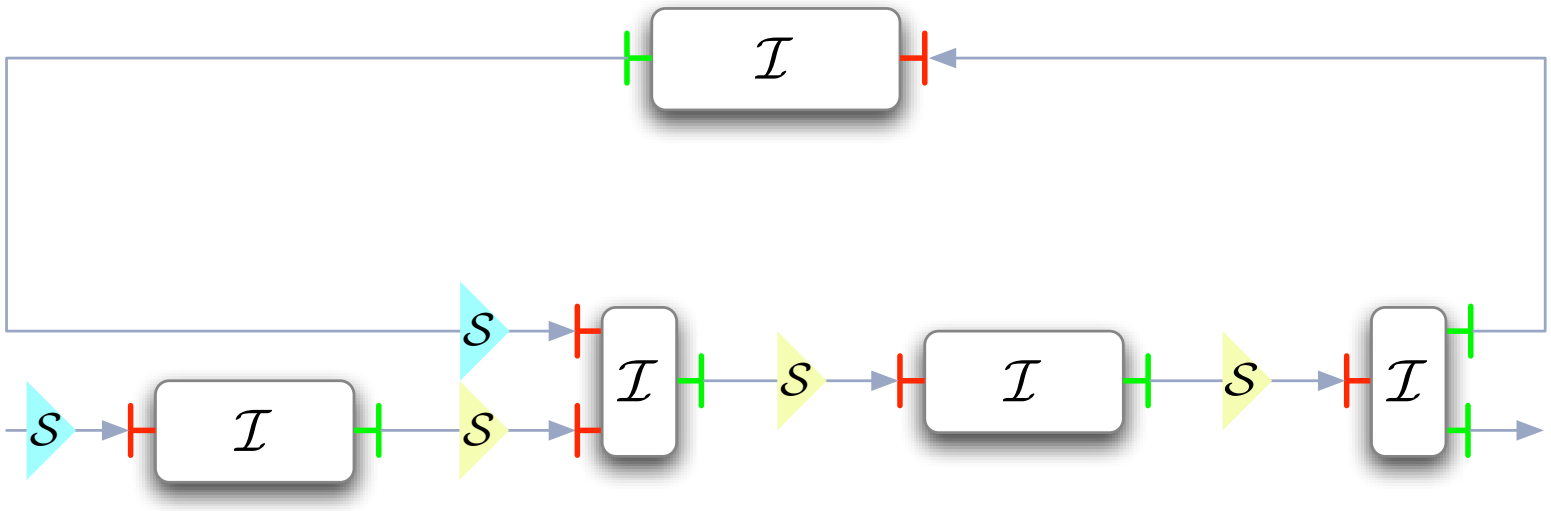
- Types des boîtes de base inférés $\mathcal{I}$
- Annotations $\mathcal{S}$ (types) en entrée et sur les boucles
- Propagation (inférence) des annotations
- Détection du manque d'annotations

Vers une approche concrète



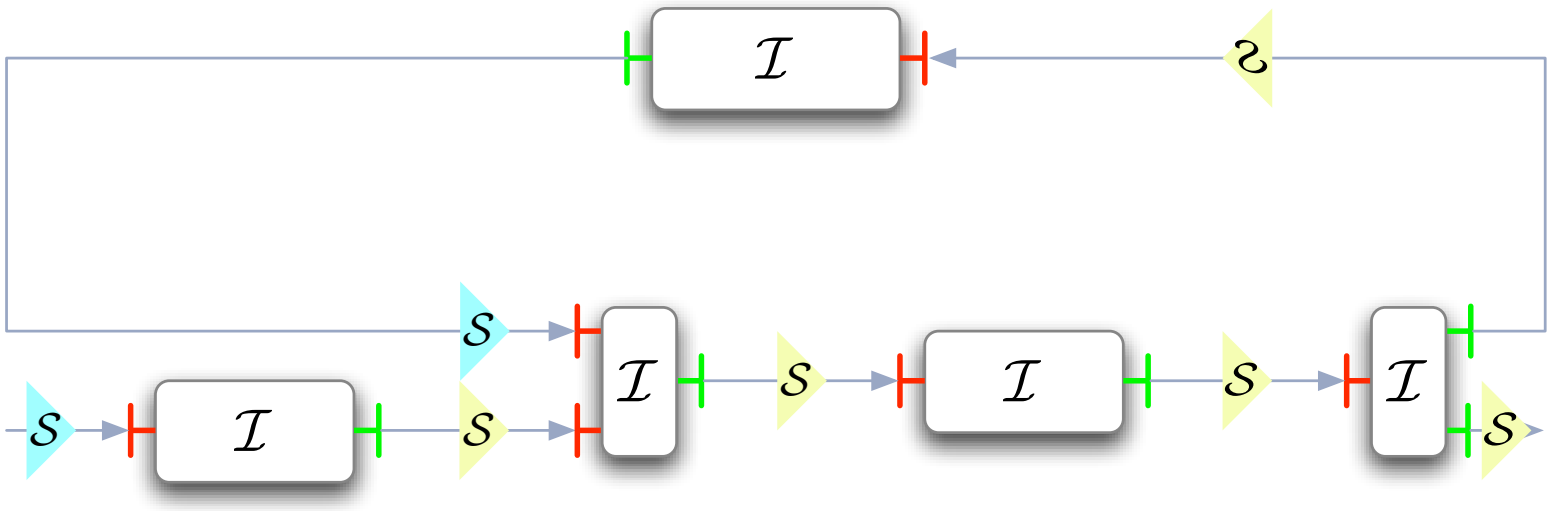
- Types des boîtes de base inférés $\mathcal{I}$
- Annotations $\mathcal{S}$ (types) en entrée et sur les boucles
- Propagation (inférence) des annotations
- Détection du manque d'annotations

Vers une approche concrète



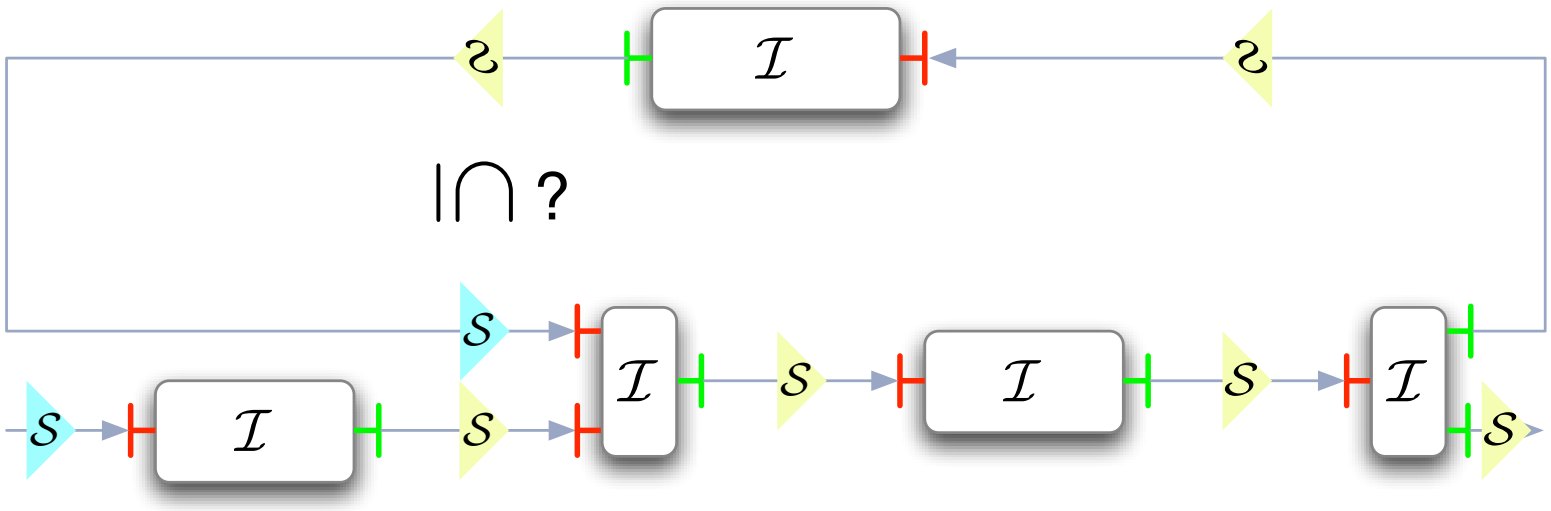
- Types des boîtes de base inférés $\mathcal{I}$
- Annotations $\mathcal{S}$ (types) en entrée et sur les boucles
- Propagation (inférence) des annotations
- Détection du manque d'annotations

Vers une approche concrète



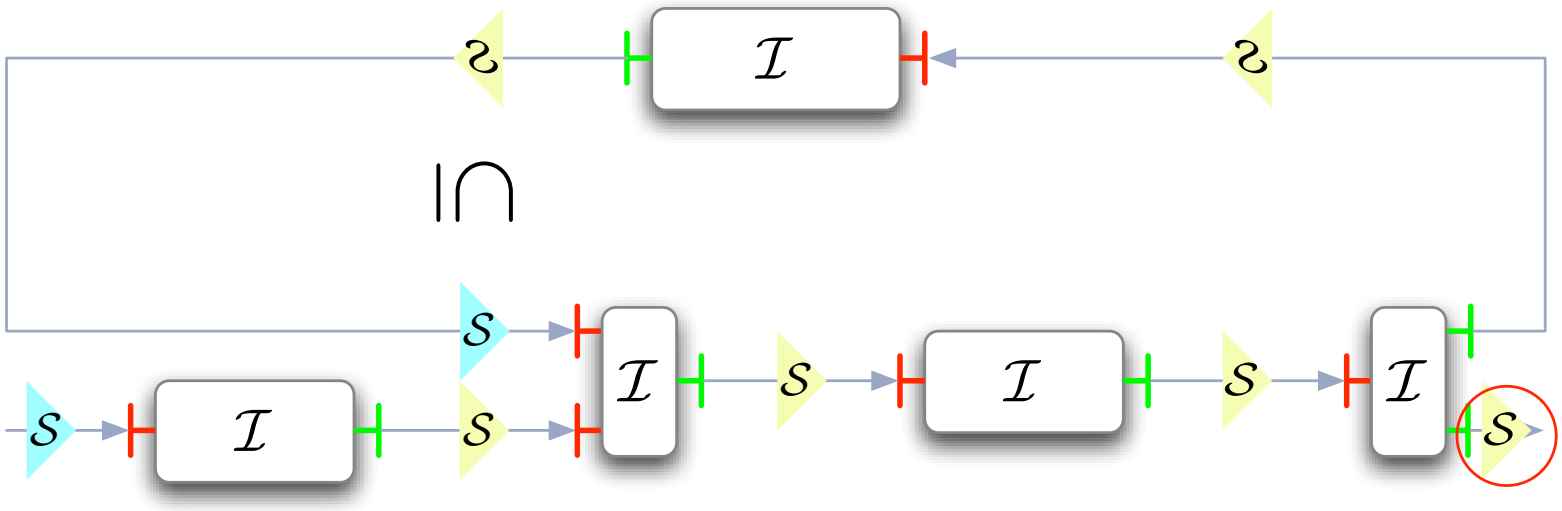
- Types des boîtes de base inférés $\mathcal{I}$
- Annotations $\mathcal{S}$ (types) en entrée et sur les boucles
- Propagation (inférence) des annotations
- Détection du manque d'annotations

Vers une approche concrète



- Types des boîtes de base inférés $\mathcal{I}$
- Annotations $\mathcal{S}$ (types) en entrée et sur les boucles
- Propagation (inférence) des annotations
- Détection du manque d'annotations
- Vérification de l'inclusion des annotations

Vers une approche concrète



- Types des boîtes de base inférés $\mathcal{I}$
- Annotations $\mathcal{S}$ (types) en entrée et sur les boucles
- Propagation (inférence) des annotations
- Détection du manque d'annotations
- Vérification de l'inclusion des annotations
- Inférence du type résultat

Systèmes de types en projet

Jade: cohérence des protocoles

- Simples noms
- Organisés en hiérarchie (cf classes de Java)
- Types comportementaux

Liaisons entre composants

- présence des liaisons obligatoires
- liaisons et cycle de vie

Utilisations de ressources

[*Recollecting resources in the π calculus*, David Teller, 2004]

A retenir

Les systèmes de types sont vos amis

A retenir

Les systèmes de types sont vos amis

Modéliser pour régner

A retenir

Les systèmes de types sont vos amis

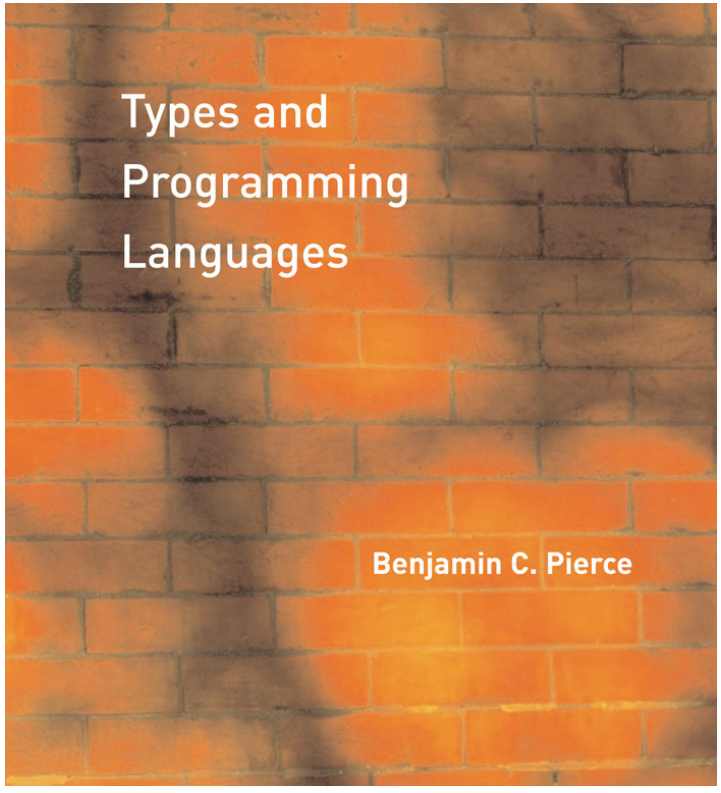
Modéliser pour régner

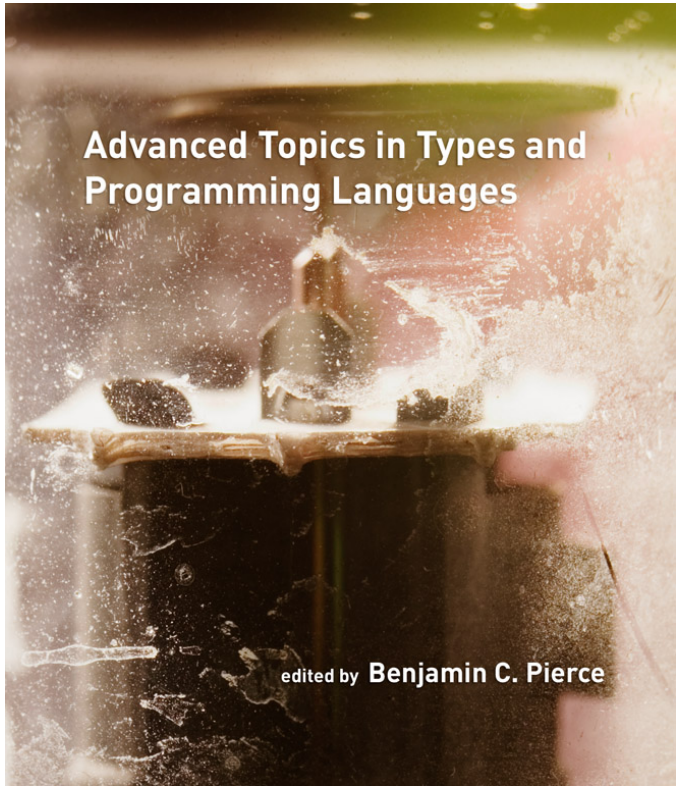
Les systèmes de types ont des limites

> Beware of bugs in the above code; I have only proved it correct, not tried it.
>
> D. Knuth

Pour aller plus loin

Types:





Modélisation: `http://sardes.inrialpes.fr/kells/`