An Abstract Separation Logic for Interlinked Extensible Records

Martin Bodin   Thomas Jensen   Alan Schmitt

Inria

29th of January

JFLA 2016

Objectif

- Analyser JavaScript, avec sa sémantique complexe ;
- Certifier l'analyseur ;
- POPL'14, JSCert, une sémantique formelle de JavaScript :

JavaScript JSCert

CPP'15 : Pourquoi redéfinir une sémantique ?

$$\frac{\ldots}{(1, E), \textit{if}\, s_1\, s_2 \Downarrow \ldots}$$

Sémantique

IfTrue

$$\frac{s_1, E \Downarrow E'}{(v, E)\,, \textit{if}\ s_1\ s_2,\ \Downarrow E'} \quad v \in \mathbb{Z}^\star$$

IfFalse

$$\frac{s_2, E \Downarrow E'}{(v, E)\,, \textit{if}\ s_1\ s_2 \Downarrow E'} \quad v \in \{0\}$$

CPP'15 : Pourquoi redéfinir une sémantique ?

Sémantique

IfTrue

$$\frac{s_1, E \Downarrow E'}{(v, E)\,, \textit{if}\ s_1\ s_2,\ \Downarrow E'} \quad v \in \mathbb{Z}^\star$$

IfFalse

$$\frac{s_2, E \Downarrow E'}{(v, E)\,, \textit{if}\ s_1\ s_2 \Downarrow E'} \quad v \in \{0\}$$

$$\frac{E, s_1 \Downarrow E'}{(1, E), \textit{if}\, s_1\ s_2 \Downarrow E'}$$

Dérivation concrète

CPP'15 : Pourquoi redéfinir une sémantique ?

Sémantique

$$\textsc{IfTrue}\quad \frac{s_1, E \Downarrow E'}{(v, E)\,, \textit{if}\ s_1\ s_2,\ \Downarrow E'} \quad v \in \mathbb{Z}^{\star}$$

$$\textsc{IfFalse}\quad \frac{s_2, E \Downarrow E'}{(v, E)\,, \textit{if}\ s_1\ s_2 \Downarrow E'} \quad v \in \{0\}$$

$$\frac{E, s_1 \Downarrow E'_1 \qquad E, s_2 \Downarrow E'_2}{(\top, E), \textit{if}\, s_1\ s_2 \Downarrow E'_1 \sqcup E'_2}$$

Dérivation Abstraite

CPP'15 : Différence entre sémantique abstraite et concrète

À partir des règles de dérivation, $\Downarrow$ et $\Downarrow^\sharp$ sont définies différemment.

Sémantique concrète $\Downarrow$	Sémantique abstraite $\Downarrow^\sharp$
On applique *n'importe quelle* règle qui s'applique	On applique *toutes* les règles qui s'appliquent

$$\dfrac{\begin{array}{cc} E_0^\sharp, s_1 \Downarrow^\sharp E_1^\sharp & E_0^\sharp, s_2 \Downarrow^\sharp E_2^\sharp \\ \uparrow \textsc{IfTrue} & \uparrow \textsc{IfFalse} \end{array}}{\left(v^\sharp, E_0^\sharp\right), \mathit{if}\ s_1\ s_2 \Downarrow^\sharp E_1^\sharp \sqcup E_2^\sharp}$$

CPP'15 : Différence entre sémantique abstraite et concrète

À partir des règles de dérivation, $\Downarrow$ et $\Downarrow^\sharp$ sont définies différemment.

Sémantique concrète $\Downarrow$	Sémantique abstraite $\Downarrow^\sharp$
On applique *n'importe quelle* règle qui s'applique	On applique *toutes* les règles qui s'appliquent
Interprétation inductive d'une dérivation $\Downarrow = \mathit{lfp}\left(\mathcal{F}\right)$	Interprétation coinductive d'une dérivation $\Downarrow^\sharp = \mathit{gfp}\left(\mathcal{F}^\sharp\right)$

$$\dfrac{\begin{array}{cc} E_0^\sharp, s_1 \Downarrow^\sharp E_1^\sharp & E_0^\sharp, s_2 \Downarrow^\sharp E_2^\sharp \\ \uparrow \textsc{IfTrue} & \uparrow \textsc{IfFalse} \end{array}}{\left(v^\sharp, E_0^\sharp\right), \mathit{if}\ s_1\ s_2 \Downarrow^\sharp E_1^\sharp \sqcup E_2^\sharp}$$

CPP'15 : Différence entre sémantique abstraite et concrète

À partir des règles de dérivation, $\Downarrow$ et $\Downarrow^\sharp$ sont définies différemment.

Sémantique concrète $\Downarrow$	Sémantique abstraite $\Downarrow^\sharp$
On applique *n'importe quelle* règle qui s'applique	On applique *toutes* les règles qui s'appliquent
Interprétation inductive d'une dérivation $\Downarrow = \mathit{lfp}\,(\mathcal{F})$	Interprétation coinductive d'une dérivation $\Downarrow^\sharp = \mathit{gfp}\left(\mathcal{F}^\sharp\right)$
	On autorise des approximations

$$\textsc{Weaken}\ \frac{P' \sqsubseteq P \qquad \{P\}\,p\,\{Q\} \qquad Q \sqsubseteq Q'}{\{P'\}\,p\,\{Q'\}}$$

CPP'15 : Sémantique abstraite sous forme de point fixe

- À partir d'un ensemble $(\sigma, t, r) \in \Downarrow_0^\sharp$.
- Application de la règle i : *apply*$_i \left(\Downarrow_0^\sharp\right)$

CPP'15 : Sémantique abstraite sous forme de point fixe

- À partir d'un ensemble $(\sigma, t, r) \in \Downarrow_0^\sharp$.
- Application de la règle i : $\mathit{apply}_i\left(\Downarrow_0^\sharp\right)$
- Application de la règle $\textsc{Weaken}$:

$$\mathit{glue}_i^\sharp\left(\Downarrow_0^\sharp\right) = \left\{ (\sigma, t, r) \;\middle|\; \begin{array}{l} \exists \sigma_0, \exists r_0, \\ \quad \sigma \sqsubseteq^\sharp \sigma_0 \wedge r_0 \sqsubseteq^\sharp r \wedge \\ \quad (\sigma_0, t, r_0) \in \mathit{apply}_i\left(\Downarrow_0^\sharp\right) \end{array} \right\}$$

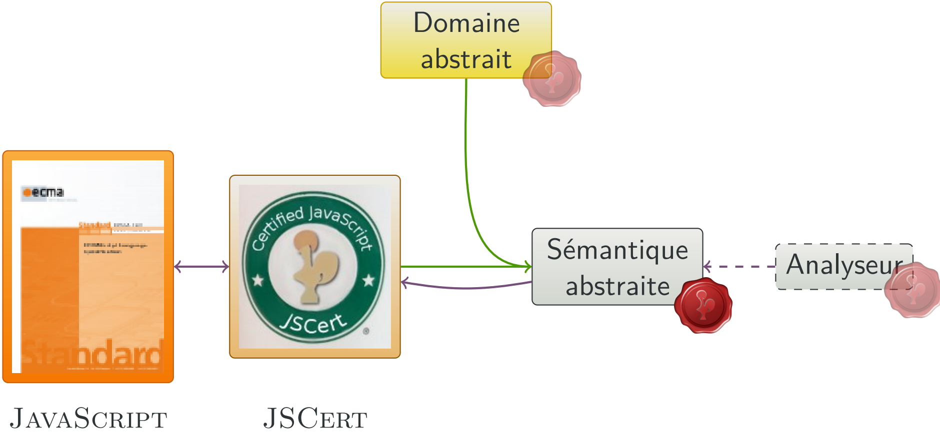
CPP'15 : Sémantique abstraite sous forme de point fixe

- À partir d'un ensemble $(\sigma, t, r) \in \Downarrow_0^\sharp$.
- Application de la règle i : $\mathit{apply}_i\left(\Downarrow_0^\sharp\right)$
- Application de la règle $\textsc{Weaken}$:

$$\mathit{glue}_i^\sharp\left(\Downarrow_0^\sharp\right) = \left\{ (\sigma, t, r) \;\middle|\; \begin{array}{l} \exists \sigma_0, \exists r_0, \\ \quad \sigma \sqsubseteq^\sharp \sigma_0 \wedge r_0 \sqsubseteq^\sharp r \wedge \\ \quad (\sigma_0, t, r_0) \in \mathit{apply}_i\left(\Downarrow_0^\sharp\right) \end{array} \right\}$$

- Application de toutes les règles : $\mathcal{F}^\sharp$
- $\Downarrow^\sharp = \mathit{gfp}\left(\mathcal{F}^\sharp\right)$

Situation globale



1. Une Sémantique abstraite basée sur la sémantique concrète

2. Logique de séparation

3. Ajout des nœuds résumés

Domaine mémoire de JavaScript

- Pas d'arithmétique pointeur !
- Des localités (« locations ») $l^i \in \mathit{Loc} \subseteq \mathit{Val}$.
- Un tas $H : \mathit{Loc} \rightharpoonup \mathfrak{F} \rightharpoonup \mathit{Val}$.

Noms de champs, champs spéciaux (prototypes, etc.)

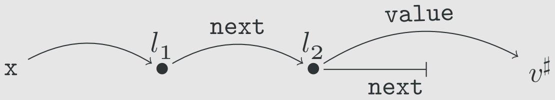
- Allocation de nouvelles localités.
- Ajout/suppression de champs.
- Possibilité de tester si un champ existe.

Interlinked Extensible Records / Enregistrements extensibles intriqués.

Logique de séparation

$$\phi ::= \textit{emp} \mid \phi_1 \star \phi_2 \mid \mathtt{x} \doteq v^\sharp \mid l \mapsto \{o\}$$
$$o ::= \mathtt{f} : v^\sharp,\, o \mid \_ : v^\sharp$$

$\mathtt{x} \doteq l_1 \star l_1 \mapsto \{\mathtt{next} : l_2, \_ : \boxtimes\} \star l_1 \mapsto \left\{\mathtt{next} : \textit{nil}, \mathtt{value} : v^\sharp, \_ : \boxtimes\right\}$



Logique de séparation

$$\phi ::= \textsf{\textit{emp}} \mid \phi_1 \star \phi_2 \mid \mathtt{x} \doteq v^\sharp \mid l \mapsto \{o\}$$
$$o ::= \mathtt{f} : v^\sharp,\ o \mid \_ : v^\sharp$$

The lattice of abstract values

$$+, \pm, -, \ldots \in v^\sharp$$
$$l \in v^\sharp$$
$$\textsf{\textit{nil}} \in v^\sharp$$
$$\boxtimes \in v^\sharp$$

Les différentes valeurs peuvent être mélangées : $+ \sqcup l \sqcup \boxtimes$.

La Règle de contexte

$$\frac{\phi, s \Downarrow^{\sharp} \phi'}{\phi \star \phi_c, s \Downarrow^{\sharp} \phi' \star \phi_c}\ \textsc{Frame}$$

- Tout ce qui n'est pas explicitement changé est inchangé.
- Permet de se focaliser lors de l'analyse de fonctions.

La Règle de contexte

$$\textsc{Frame}\quad \frac{\phi, s \Downarrow^{\sharp} \phi'}{\phi \star \phi_c, s \Downarrow^{\sharp} \phi' \star \phi_c}$$

- Impossible de changer ϕ en ϕ' si l'interface change, même si $\gamma(\phi) = \gamma(\phi')$.
- Solution : conserver l'interface dans une *membrane*.

$$\begin{aligned}&(l_a \to l_1, l_b \to l_2 \,|\, l_1 \mapsto \{\texttt{f} : l_2, \_ : \boxtimes\})\\ &= (l_a \to l_2, l_b \to l_3 \,|\, l_2 \mapsto \{\texttt{f} : l_3, \_ : \boxtimes\})\end{aligned}$$

Quid des boucles ?

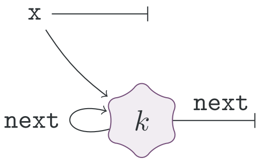
- Une abstraction simple et générique : les nœuds résumés.

$\texttt{x} := \textit{nil};$
$\textit{while}\,?\,($
$\quad \texttt{t} := \{\};$
$\quad \texttt{t.next} := \texttt{x};$
$\quad \texttt{x} := \texttt{t}$
$)$

Quid des boucles ?

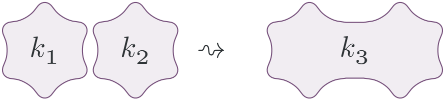
- Une abstraction simple et générique : les nœuds résumés.

$\mathtt{x} := \textit{nil};$
$\textit{while}\,?\,($
$\quad \mathtt{t} := \{\}^k;$
$\quad \mathtt{t.next} := \mathtt{x};$
$\quad \mathtt{x} := \mathtt{t}$
$)$



$$k \mapsto \{\mathtt{next} : k \sqcup \textit{nil},\ \_ : \boxtimes\}$$

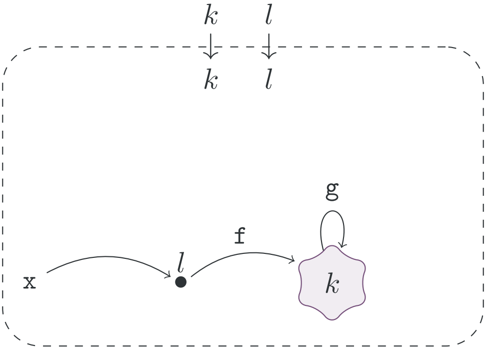
Les nœuds résumés changent les interfaces !



$$k_1 \mapsto \{\texttt{f} : +, \texttt{\_} : \boxtimes\} \star k_2 \mapsto \{\texttt{f} : +, \texttt{\_} : \boxtimes\}$$
$$\rightsquigarrow k_3 \mapsto \{\texttt{f} : +, \texttt{\_} : \boxtimes\}$$

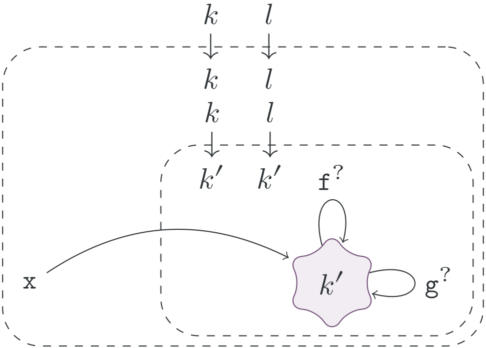
Un exemple de manipulation de membrane

$$(k \to k, l \to l \,|\, \mathtt{x} \doteq l \star l \mapsto \{\mathtt{f} : k,\ \_ : \boxtimes\} \star k \mapsto \{\mathtt{g} : k,\ \_ : \boxtimes\})$$



Un exemple de manipulation de membrane

$$(k \to k, l \to l \mid \mathtt{x} \doteq l \star l \mapsto \{\mathtt{f} : k,\ \_ : \boxtimes\} \star k \mapsto \{\mathtt{g} : k,\ \_ : \boxtimes\})$$

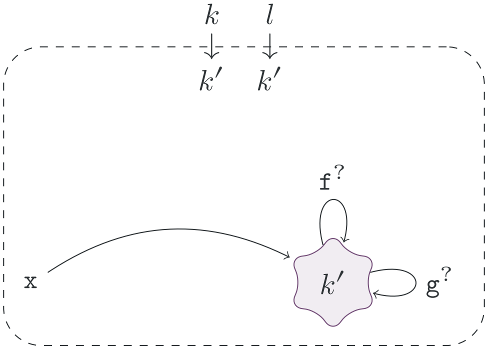


$$(k \to k, l \to l \mid \mathtt{x} \doteq l)$$

$$\circledast \left(k \to k', l \to k' \,\middle|\, k' \mapsto \left\{\mathtt{f} : k' \sqcup \boxtimes,\ \mathtt{g} : k' \sqcup \boxtimes,\ \_ : \boxtimes\right\}\right)$$

Un exemple de manipulation de membrane

$$\left(k \to k, l \to l \,\middle|\, \mathtt{x} \doteq l \star l \mapsto \{\mathtt{f} : k,\ \_ : \boxtimes\} \star k \mapsto \{\mathtt{g} : k,\ \_ : \boxtimes\}\right)$$

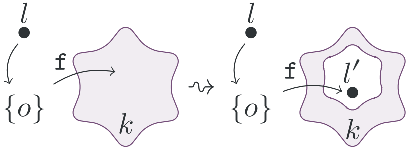


$$\left(k \to k', l \to k' \,\middle|\, \mathtt{x} \doteq k' \star k' \mapsto \left\{\mathtt{f} : k' \sqcup \boxtimes, \mathtt{g} : k' \sqcup \boxtimes,\ \_ : \boxtimes\right\}\right)$$

D'autres manipulations de membranes

- Transormer un nœud l précis en un nœud résumé k,

- Matérialisation de nœuds résumés,



- Séparer les nœuds résumés selon un critère de filtre...

Conclusion et travaux futurs

- Vers un domaine pour JavaScript,
  - Objects extensibles.
  - Reste les clôtures, domaines pour chaînes de caractère, etc.
- Compatible avec la logique de séparation,
- Compatible avec l'interprétation abstraite certifiée.

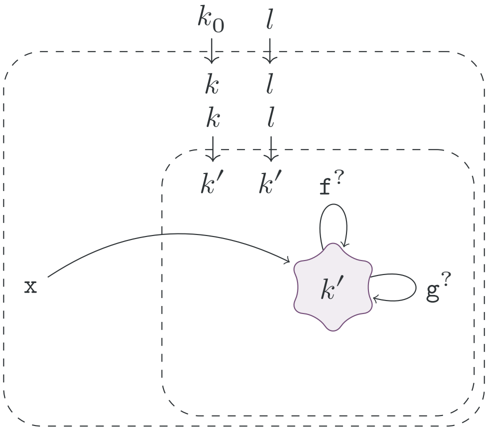
Étapes suivantes

- Coq en cours.
- Un analyseur certifié d'un langage intermédiaire.
- Passer à l'échelle de JavaScript.

Merci d'avoir écouté !

$$(k \to k \,|\, \mathtt{x} \doteq l)$$

$$\circledast \left(k \to k', l \to k' \,\middle|\, k' \mapsto \{\mathtt{f} : k' \sqcup \boxtimes, \mathtt{g} : k' \sqcup \boxtimes, \_ : \boxtimes\}\right)$$



Avez-vous des questions ?

1. Une Sémantique abstraite basée sur la sémantique concrète
2. Logique de séparation
3. Ajout des nœuds résumés

Planches pour questions

- ▸ Syntax
- ▸ Membrane syntax
- ▸ Dark matter
- ▸ Why the Dark matter ?
- ▸ Abstract rules
- ▸ A more complex example

Syntaxe

$$s ::= \textit{skip} \quad \mid s_1;\ s_2 \quad \mid \textit{if}\, e\, s_1\, s_2$$
$$\mid \textit{while}\, e\, s \quad \mid \textit{throw} \quad \mid \mathtt{x} := e$$
$$\mid e_1\,\mathtt{.f} := e_2 \quad \mid \textit{delete}\, e\,\mathtt{.f}$$

$$e ::= n \in \mathbb{Z} \quad \mid\ ? \quad \mid \mathtt{x} \in \textit{Var} \quad \mid \textit{nil}$$
$$\mid \{\} \quad \mid e\,\mathtt{.f} \quad \mid \mathtt{f}\ \mathtt{in}\, e \quad \mid \neg\ e$$
$$\mid =\ e_1\ e_2 \quad \mid \bowtie\ e_1\ e_2 \quad (\bowtie\ \in \{>, +, -\})$$

Syntax With Membranes

$$\phi ::= \textit{emp} \mid \phi_1 \star \phi_2 \mid \mathtt{x} \doteq v^\sharp \mid h \mapsto \{o\}$$
$$h ::= l \mid k$$

$$o ::= \mathtt{f} : v^\sharp,\, o \mid \_ : v^\sharp$$

$$m \in \mathfrak{M} ::= h \to h_1 + \ldots + h_n \mid \nu h$$

$$\Phi ::= (M \mid \phi) \qquad M \in \mathcal{P}_f(\mathfrak{M})$$

- Syntax
- Membrane syntax
- Dark matter
- Why the Dark matter ?
- Abstract rules
- A more complex example

~~The~~ One Problem of JavaScript

An Introduction to Separation Logic, by John C. Reynolds

The soundness of the frame rule is surprisingly sensitive to the semantics of our programming language. Suppose, for example, we changed the behavior of deallocation, so that, instead of causing a memory fault, `dispose x` behaved like `skip` when the value of `x` was not in the domain of the heap. Then $\{emp\}\mathtt{dispose\ x}\{emp\}$ would be valid, and the frame rule could be used to infer $\{emp \star \mathtt{x} \doteq 10\}\mathtt{dispose\ x}\{emp \star \mathtt{x} \doteq 10\}$. Then, since *emp* is a neutral element for $\star$, we would have $\{\mathtt{x} \doteq 10\}\mathtt{dispose\ x}\{\mathtt{x} \doteq 10\}$, which is patently false.

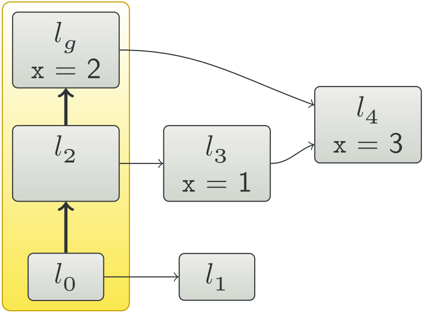
- JavaScript's `delete` does exactly this.

~~The~~ One Problem of JavaScript

An Introduction to Separation Logic, by John C. Reynolds

$$\dfrac{\dfrac{\overline{\{\mathit{emp}\}\mathtt{dispose\ x}\{\mathit{emp}\}}\ \textsc{AlreadyDisposed}}{\{\mathit{emp} \star \mathtt{x} \doteq 10\}\mathtt{dispose\ x}\{\mathit{emp} \star \mathtt{x} \doteq 10\}}\ \textsc{Frame}}{\{\mathtt{x} \doteq 10\}\mathtt{dispose\ x}\{\mathtt{x} \doteq 10\}}\ \textsc{Rewrite}$$

- JavaScript's `delete` does exactly this.



lexical environment

It is very important to track the absence of properties in objects.

About the Dark Matter "$\boxtimes$"

Can you tell the difference between these formulae ?

- *emp*
- *True*
- $l \mapsto \{\mathtt{f} : \boxtimes, \_ : \boxtimes\}$
- $l \mapsto \{\mathtt{f} : \boxtimes, \_ : \boxtimes\} \star \phi$
- $l \mapsto \{\mathtt{f} : \bot, \_ : \boxtimes\}$

About the Dark Matter "$\boxtimes$"

Can you tell the difference between these formulae ?

- *emp*
- *True*
- $l \mapsto \{\texttt{f} : \boxtimes, \_ : \boxtimes\}$
- $l \mapsto \{\texttt{f} : \boxtimes, \_ : \boxtimes\} \star \phi$
- $l \mapsto \{\texttt{f} : \bot, \_ : \boxtimes\} \ = \mathit{False}$

$$\gamma\,(\mathit{emp}) = \gamma\,(l \mapsto \{\texttt{f} : \boxtimes, \_ : \boxtimes\})$$

CPP'15 in a Nutshell : Abstract Rules

IfTrue

$$\frac{E, s_1 \Downarrow E'}{(v, E)\,, \textit{if}\ s_1\ s_2 \Downarrow E'} \quad v \in \mathbb{Z}^\star$$

IfFalse

$$\frac{E, s_2 \Downarrow E'}{(v, E)\,, \textit{if}\ s_1\ s_2 \Downarrow E'} \quad v \in \{0\}$$

$$\downarrow$$

IfTrue

$$\frac{E^\sharp, s_1 \Downarrow^\sharp E'^\sharp}{\left(v^\sharp, E^\sharp\right), \textit{if}\ s_1\ s_2 \Downarrow^\sharp E'^\sharp} \quad \gamma\left(v^\sharp\right) \cap \mathbb{Z}^\star \neq \emptyset$$

IfFalse

$$\frac{E^\sharp, s_2 \Downarrow^\sharp E'^\sharp}{\left(v^\sharp, E^\sharp\right), \textit{if}\ s_1\ s_2 \Downarrow^\sharp E'^\sharp} \quad \gamma\left(v^\sharp\right) \cap \{0\} \neq \emptyset$$

- ▸ Syntax
- ▸ Membrane syntax
- ▸ Dark matter
- ▸ Why the Dark matter ?
- ▸ Abstract rules
- ▸ A more complex example

Updating the Frame Rule

The $\star$ for membraned formulae $\Phi = (M \mid \phi)$

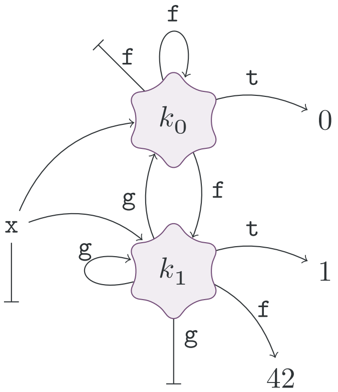
$$(M \mid \phi) \circledstar (M_c \mid \phi_c) = \text{«} \, (M_c \mid (M \mid \phi) \star \phi_c) \, \text{»}$$

$$\dfrac{\Phi, s \Downarrow^\sharp \Phi'}{\Phi \circledstar \Phi_c, s \Downarrow^\sharp \Phi' \circledstar \Phi_c}\ \textsc{Frame}$$

- Syntax
- Membrane syntax
- Dark matter
- Why the Dark matter ?
- Abstract rules
- A more complex example

Example

$\mathtt{x} := \mathit{nil};$
$\mathit{while}\,?\,($
$\quad \mathtt{t} := \mathtt{x};$
$\quad \mathtt{x} := \{\};$
$\quad \mathit{if}\,?\,($
$\quad\quad \mathtt{x.t} := 0;$
$\quad\quad \mathtt{x.f} := \mathtt{t}$
$\quad)($
$\quad\quad \mathtt{x.t} := 1;$
$\quad\quad \mathtt{x.g} := \mathtt{t};$
$\quad\quad \mathtt{x.f} := 42$
$\quad)$
$);$
$\mathit{while}\,\mathtt{x} \neq \mathit{nil}\,($
$\quad \mathit{if}\,\mathtt{x.t}\,(\mathtt{x} := \mathtt{x.g})(\mathtt{x} := \mathtt{x.f})$
$)$



We can express a loop invariant without adding new constructions !

- Syntax
- Membrane syntax
- Dark matter
- Why the Dark matter ?
- Abstract rules
- A more complex example

1. Une Sémantique abstraite basée sur la sémantique concrète

2. Logique de séparation

3. Ajout des nœuds résumés