

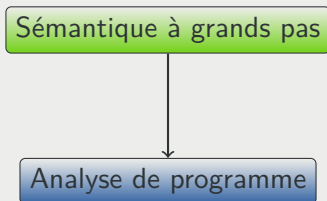
# Pretty-big-step-semantics-based Certified Abstract Interpretation

Martin Bodin, Thomas Jensen, Alan Schmitt

9 Janvier 2014

JFLA 2014

Comment passer de façon systématique d'une sémantique à une analyse de programme certifiée ?



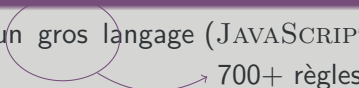
## Ce que nous avons

- Une formalisation d'un gros langage (JAVASCRIPT).
- Sémantique écrite sous une forme particulière (*Pretty-Big-Step*).

## Ce que nous voulons

- Définir des analyses.
- Analyses *certifiées*.

## Ce que nous avons

- Une formalisation d'un gros langage (JAVASCRIPT).  
 700+ règles
- Sémantique écrite sous une forme particulière (*Pretty-Big-Step*).

## Ce que nous voulons

- Définir des analyses.
- Analyses *certifiées*.

# Un Langage jouet : O'WHILE

$s ::=$

- | skip
- |  $s_1; s_2$
- | if  $e$  then  $s_1$  else  $s_2$
- | while  $e$  do  $s$
- |  $x = e$
- |  $e_1.f = e_2$
- | delete  $e.f$

$e ::=$

- |  $c$
- |  $x$
- |  $e_1$  op  $e_2$
- | { }
- |  $e.f$

# Pretty-Big-Step

Un problème des sémantiques à grands pas

Duplication des règles dès l'ajout d'exceptions, de divergence, etc.

$$\frac{E, H, \underline{e} \rightarrow r \quad E' = E[x \mapsto v]}{S, \underline{x} = e \rightarrow E', H} \text{ASG}$$

Le Pretty-Big-Step

- Une sémantique à grands pas.
- Des formes intermédiaires comme en petits pas.

$$\frac{E, H, \underline{e} \rightarrow r \quad E, H, x =_1 r \rightarrow r'}{E, H, \underline{x} = e \rightarrow r'} \text{ASG}$$

$$\frac{E' = E[x \mapsto v]}{S, \underline{x} =_1 (E, H, v) \rightarrow E', H} \text{ASG1}$$

# Pretty-Big-Step

Un problème des sémantiques à grands pas

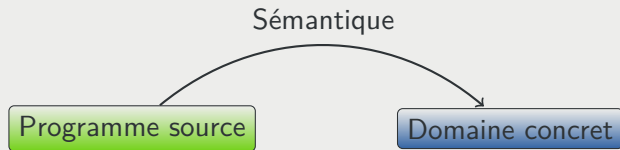
Duplication des règles dès l'ajout d'exceptions, de divergence, etc.

$$\frac{E, H, \underline{e} \rightarrow r \quad E' = E[x \mapsto v]}{S, \underline{x = e} \rightarrow E', H} \text{ASG}$$

## Le Pretty-Big-Step

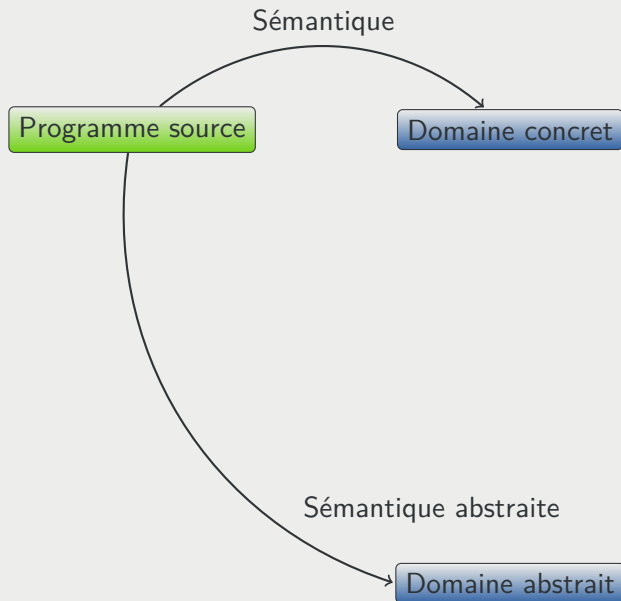
- Une sémantique à grands pas.
- Des formes intermédiaires comme en petits pas.

$$\frac{E, H, \underline{e} \rightarrow r \quad E, H, \underline{x =_1 r} \rightarrow r'}{E, H, \underline{x = e} \rightarrow r'} \text{ASG} \qquad \frac{E' = E[x \mapsto v]}{S, \underline{x =_1 (E, H, v)} \rightarrow E', H} \text{ASG1}$$

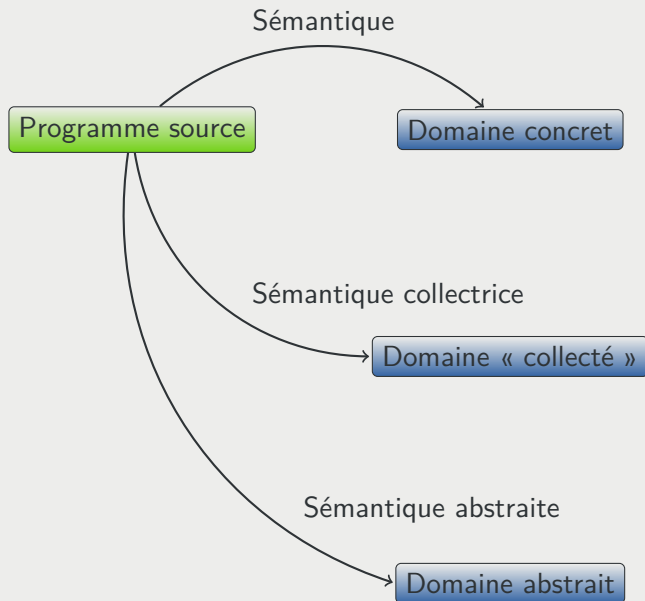




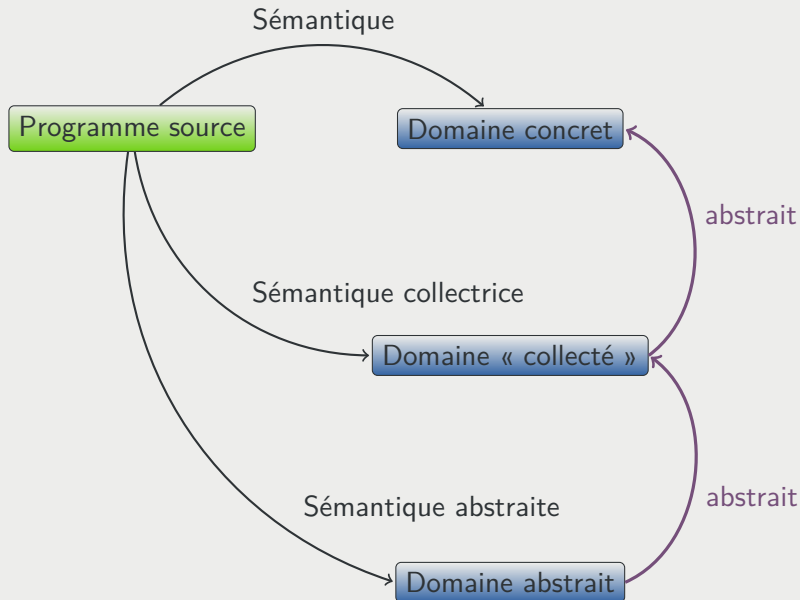
# Les Analyses : le Schéma classique



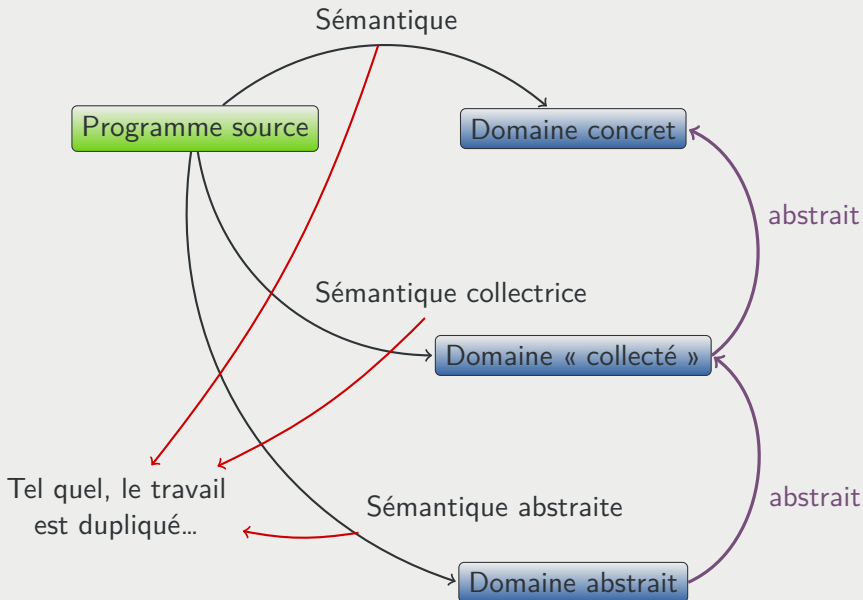
# Les Analyses : le Schéma classique



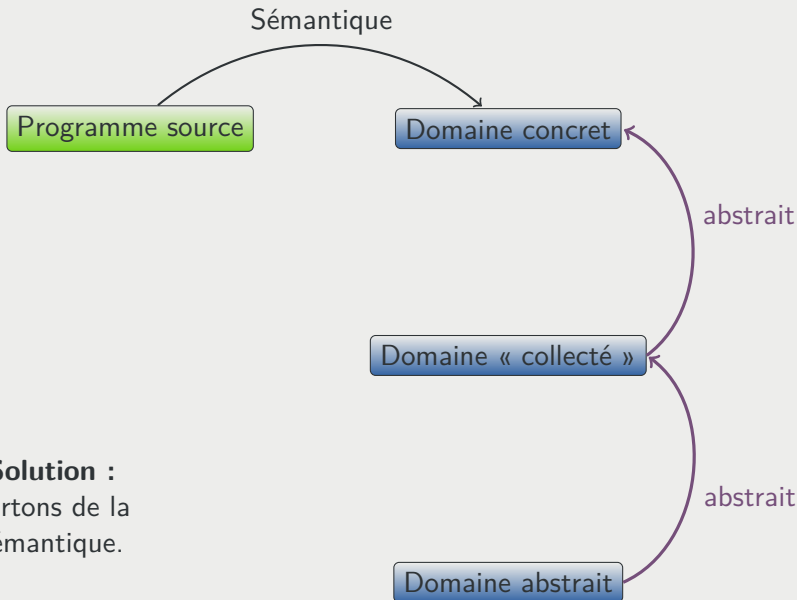
# Les Analyses : le Schéma classique



# Les Analyses : le Schéma classique



# Les Analyses : le Schéma classique

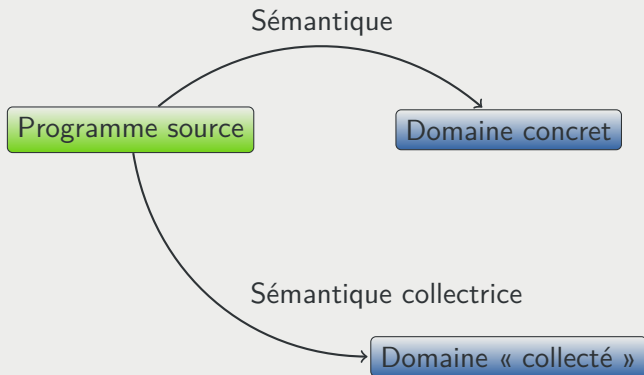


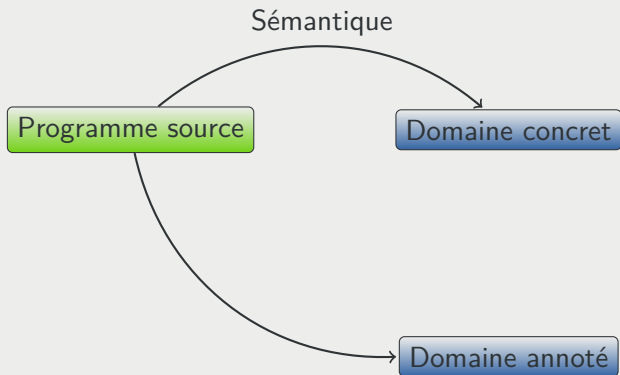
**Solution :**  
partons de la  
sémantique.

- 1 Annotations
  - Traces
  - Flots directs

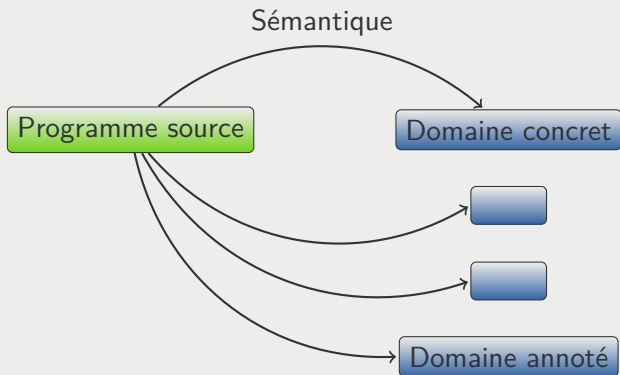
- 2 Sémantique abstraite

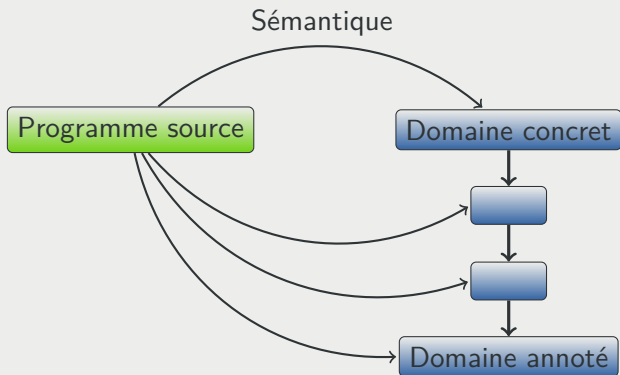
# Annoter la sémantique

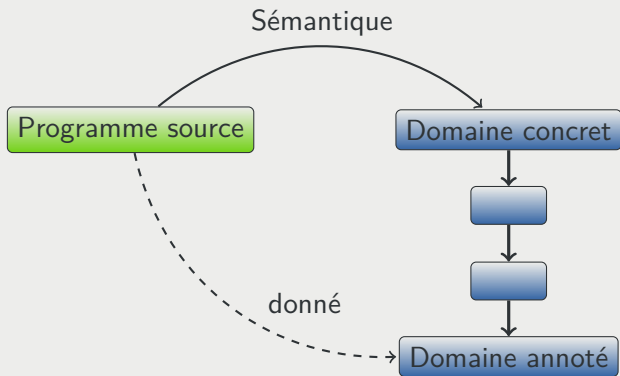












# Un exemple de chaîne d'annotations

- Nous voulons détecter des flots du type  $x^{P1} \rightleftharpoons y^{P2}$ .

```
t = x ;  
/* ... */  
y = t
```

- On a besoin d'avoir les traces.
- On a besoin d'avoir un environnement stockant les dates de dernières modifications.

# Un exemple de chaîne d'annotations

- Nous voulons détecter des flots du type  $x^{T_1} \rightsquigarrow y^{T_2}$ .

```
t = x ;  
/* ... */  
y = t
```

- On a besoin d'avoir les traces.
- On a besoin d'avoir un environnement stockant les dates de dernières modifications.

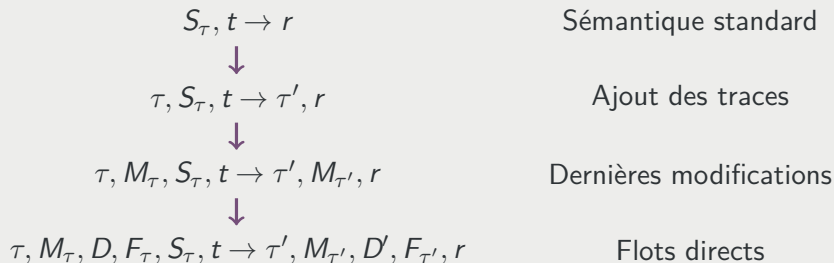
# Un exemple de chaîne d'annotations

- Nous voulons détecter des flots du type  $x^{T1} \rightsquigarrow y^{T2}$ .

```
t = x ;  
/* ... */  
y = t
```

- On a besoin d'avoir les traces.
- On a besoin d'avoir un environnement stockant les dates de dernières modifications.

# Un exemple de chaîne d'annotations



On a rendu le flot explicite dans la dérivation...  
mais *sans ajouter d'information* !

# Les Annotations sont locales

N'oublions pas les 700+ règles de JAVASCRIPT !

On se refuse tout copier/coller.

En Pretty-Big-step, il n'y a que trois types de règles

$$\frac{}{a \rightarrow a}$$

$$\frac{a \rightarrow a}{a \rightarrow a}$$

$$\frac{a \rightarrow a \quad a \rightarrow a}{a \rightarrow a}$$

$$\frac{\frac{a_1 \rightarrow a_2}{\quad} \quad \frac{a_4 \rightarrow a_5}{\quad} \quad \frac{a_3 \rightarrow a_6}{\quad}}{a_0 \rightarrow a_7}$$



# Les Annotations sont locales

N'oublions pas les 700+ règles de JAVASCRIPT !

On se refuse tout copier/coller.

En Pretty-Big-step, il n'y a que trois types de règles

$$\frac{}{a \rightarrow a}$$

$$\frac{a \rightarrow a}{a \rightarrow a}$$

$$\frac{a \rightarrow a \quad a \rightarrow a}{a \rightarrow a}$$

$$\frac{\frac{a_1 \rightarrow a_2}{\quad} \quad \frac{\frac{a_4 \rightarrow a_5}{\quad} \quad a_3 \rightarrow a_6}{\quad}}{a_0 \rightarrow a_7}$$

# Les Annotations sont locales

N'oublions pas les 700+ règles de JAVASCRIPT !

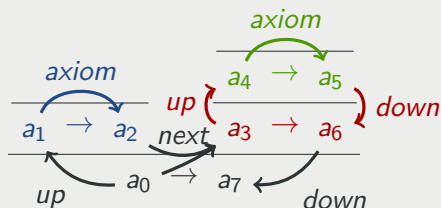
On se refuse tout copier/coller.

En Pretty-Big-step, il n'y a que trois types de règles

$$\frac{}{a \rightarrow a}$$

$$\frac{a \rightarrow a}{a \rightarrow a}$$

$$\frac{a \rightarrow a \quad a \rightarrow a}{a \rightarrow a}$$

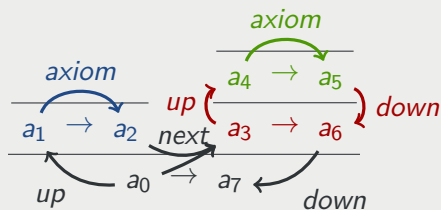


# Définition des Traces

Les **traces**  $\tau$  représentent un point dans l'arbre de dérivation, c'est à dire le *temps d'exécution*.

Sensible au flot

$$\text{ASG} \frac{S, \underline{e} \rightarrow r_0 \quad S, \underline{x} =_1 r_0 \rightarrow r}{S, \underline{x} = e \rightarrow r}$$

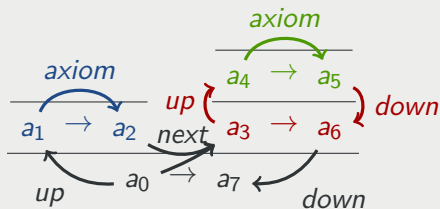


# Définition des Traces

Les **traces**  $\tau$  représentent un point dans l'arbre de dérivation, c'est à dire le *temps d'exécution*.

Sensible au flot

$$\text{ASG} \frac{S, e \rightarrow r_0 \quad S, \underline{x =_1 r_0} \rightarrow r}{\tau_0, S, \underline{x = e} \rightarrow r}$$



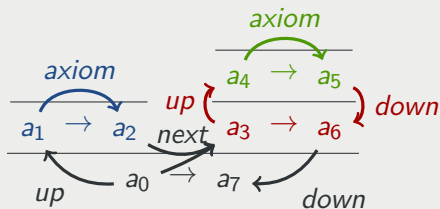
# Définition des Traces

Les **traces**  $\tau$  représentent un point dans l'arbre de dérivation, c'est à dire le *temps d'exécution*.

Sensible au flot

$$\tau_1 = \tau_0 \# [\text{ASGE}]$$

$$\text{ASG} \frac{\tau_1, S, \underline{e} \rightarrow r_0 \quad S, \underline{x} =_1 r_0 \rightarrow r}{\tau_0, S, \underline{x} = e \rightarrow r}$$



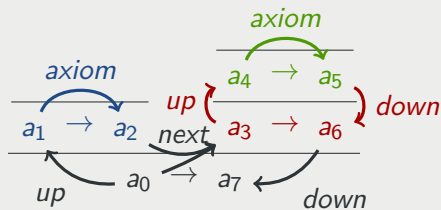
# Définition des Traces

Les **traces**  $\tau$  représentent un point dans l'arbre de dérivation, c'est à dire le *temps d'exécution*.

→ Sensible au flot

$$\tau_1 = \tau_0 \# [\text{ASGE}]$$

$$\text{ASG} \frac{\tau_1, S, \underline{e} \rightarrow \tau_2, r_0 \quad S, \underline{x} =_1 r_0 \rightarrow r}{\tau_0, S, \underline{x} = e \rightarrow r}$$

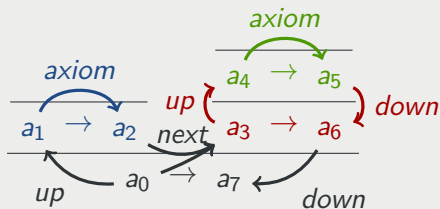


# Définition des Traces

Les **traces**  $\tau$  représentent un point dans l'arbre de dérivation, c'est à dire le *temps d'exécution*.

Sensible au flot

$$\text{ASG} \frac{\tau_1 = \tau_0 \# [\text{ASGE}] \quad \tau_3 = \tau_2 \# [\text{ASG1}] \quad \tau_1, S, \underline{e} \rightarrow \tau_2, r_0 \quad \tau_3, S, \underline{x} =_1 r_0 \rightarrow r}{\tau_0, S, \underline{x} = e \rightarrow r}$$

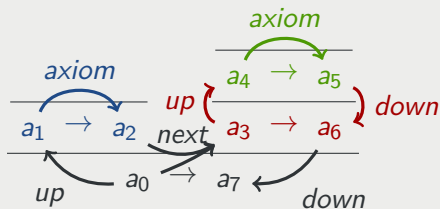


# Définition des Traces

Les **traces**  $\tau$  représentent un point dans l'arbre de dérivation, c'est à dire le *temps d'exécution*.

Sensible au flot

$$\begin{array}{c} \tau_1 = \tau_0 \# [\text{ASGE}] \quad \tau_3 = \tau_2 \# [\text{ASG1}] \\ \tau_1, S, \underline{e} \rightarrow \tau_2, r_0 \quad \tau_3, S, \underline{x =_1 r_0} \rightarrow \tau_4, r \\ \text{ASG} \frac{}{\tau_0, S, \underline{x = e} \rightarrow r} \end{array}$$



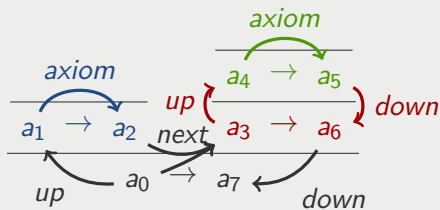


# Définition des Traces

Les **traces**  $\tau$  représentent un point dans l'arbre de dérivation, c'est à dire le *temps d'exécution*.

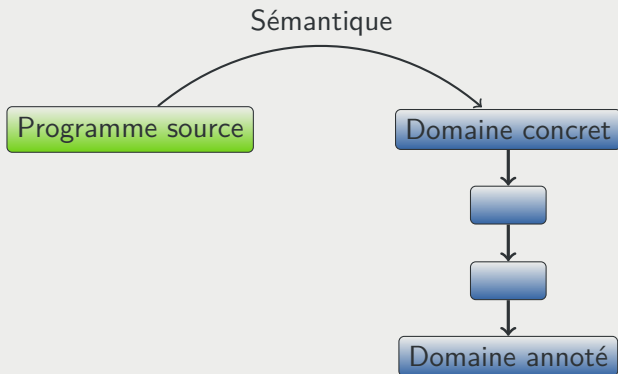
Sensible au flot

$$\text{ASG} \frac{\tau_1 = \tau_0 \# [\text{ASGE}] \quad \tau_3 = \tau_2 \# [\text{ASG1}] \quad \tau_5 = \tau_4 \# [\text{ASG}] \quad \tau_1, S, \underline{e} \rightarrow \tau_2, r_0 \quad \tau_3, S, \underline{x} =_1 r_0 \rightarrow \tau_4, r}{\tau_0, S, \underline{x} = e \rightarrow \tau_5, r}$$



## Et les 700+ règles de JAVASCRIPT ?

Si l'on doit définir plusieurs fonctions par règles, cela ne simplifie pas tant que cela la tâche...



En fait, la plupart du temps, ces fonctions ne font que propager leur annotation.

# Partir de la sémantique

$$\frac{}{S, \text{skip} \rightarrow S} \text{SKIP} \quad \frac{S, s_1 \rightarrow r \quad S, r; s_2 \rightarrow r'}{S, s_1; s_2 \rightarrow r'} \text{SEQ} \quad \frac{S', s \rightarrow r}{S, S'; s \rightarrow r} \text{SEQ1} \quad \frac{S, e \rightarrow r \quad S, \text{ifl}(r, s_1, s_2) \rightarrow r'}{S, \text{if } e \text{ then } s_1 \text{ else } s_2 \rightarrow r'} \text{IF}$$

$$\frac{S', s_1 \rightarrow r}{S, \text{ifl}((S', \text{true}), s_1, s_2) \rightarrow r} \text{IFTRUE} \quad \frac{S', s_2 \rightarrow r}{S, \text{ifl}((S', \text{false}), s_1, s_2) \rightarrow r} \text{IFFALSE} \quad \frac{S, e \rightarrow r \quad S, \text{while1}(r, e, s) \rightarrow r'}{S, \text{while } e \text{ do } s \rightarrow r'} \text{WHILE}$$

$$\frac{S', s \rightarrow r \quad S', \text{while2}(r, e, s) \rightarrow r'}{S, \text{while1}((S', \text{true}), e, s) \rightarrow r'} \text{WHILETRUE1} \quad \frac{S', \text{while } e \text{ do } s \rightarrow r}{S, \text{while2}(S', e, s) \rightarrow r} \text{WHILETRUE2}$$

$$\frac{}{S, \text{while1}((S', \text{false}), e, s) \rightarrow S'} \text{WHILEFALSE}$$

$$\frac{E, H, e \rightarrow r \quad E, H, x =_1 r \rightarrow r'}{E, H, x = e \rightarrow r'} \text{ASG}$$

$$\frac{E' = E[x \mapsto v]}{S, x =_1 (E, H, v) \rightarrow E', H} \text{ASG1} \quad \frac{S, e_1 \rightarrow r \quad S, r; f =_1 e_2 \rightarrow r'}{S, e_1; f = e_2 \rightarrow r'} \text{FLDASG} \quad \frac{S', e \rightarrow r \quad S', l; f =_2 r \rightarrow r'}{S, (S', l); f =_1 e \rightarrow r'} \text{FLDASG1}$$

$$\frac{H[l] = o \quad o' = o[f \mapsto v] \quad H' = H[l \mapsto o']}{S, l; f =_2 (E, H, v) \rightarrow E, H'} \text{FLDASG2} \quad \frac{S, e \rightarrow r \quad S, \text{delete1 } r; f \rightarrow r'}{S, \text{delete } e; f \rightarrow r'} \text{DEL}$$

$$\frac{H[l] = o \quad o[f] \neq \perp \quad o' = o[f \mapsto \perp] \quad H' = H[l \mapsto o']}{S, \text{delete1 } (E, H, l); f \rightarrow E, H'} \text{DEL1}$$

$$\frac{}{S, c \rightarrow S, c} \text{CST}$$

$$\frac{E[x] = v}{E, H, x \rightarrow E, H, v} \text{VAR}$$

$$\frac{S, e_1 \rightarrow r \quad S, r \text{ op}_1 e_2 \rightarrow r'}{S, e_1 \text{ op}_1 e_2 \rightarrow r'} \text{BIN}$$

$$\frac{S', e_2 \rightarrow r \quad S', v_1 \text{ op}_2 r \rightarrow r'}{S, (S', v_1) \text{ op}_2 e_2 \rightarrow r'} \text{BIN1}$$

$$\frac{v = v_1 \text{ op}_2 v_2}{S, v_1 \text{ op}_2 (S, v_2) \rightarrow S, v} \text{BIN2}$$

$$\frac{H[l] = \perp \quad H' = H[l \mapsto \{\}] }{E, H, \{\} \rightarrow E, H', l} \text{OBJ}$$

$$\frac{S, e \rightarrow r \quad S, r; i; f \rightarrow r'}{S, e; f \rightarrow r'} \text{FLD}$$

$$\frac{H[l] = o \quad o[f] = v}{E, H, (E', H', l); i; f \rightarrow E', H', v} \text{FLD1}$$

$$\frac{\text{abort}(t_e) = r}{S, t_e \rightarrow r} \text{ABORT}$$

# Partir de la sémantique

$$\frac{}{S, \text{skip} \rightarrow S} \text{SKIP} \quad \frac{S, s_1 \rightarrow r \quad S, r; s_2 \rightarrow r'}{S, s_1; s_2 \rightarrow r'} \text{SEQ} \quad \frac{S', s \rightarrow r}{S, S'; s \rightarrow r} \text{SEQ1} \quad \frac{S, e \rightarrow r \quad S, \text{ifl}(r, s_1, s_2) \rightarrow r'}{S, \text{if } e \text{ then } s_1 \text{ else } s_2 \rightarrow r'} \text{IF}$$

$$\frac{S', s_1 \rightarrow r}{S, \text{ifl}((S', \text{true}), s_1, s_2) \rightarrow r} \text{IFTRUE} \quad \frac{S', s_2 \rightarrow r}{S, \text{ifl}((S', \text{false}), s_1, s_2) \rightarrow r} \text{IFFALSE} \quad \frac{S, e \rightarrow r \quad S, \text{while1}(r, e, s) \rightarrow r'}{S, \text{while } e \text{ do } s \rightarrow r'} \text{WHILE}$$

$$\frac{S', s \rightarrow r \quad S', \text{while2}(r, e, s) \rightarrow r'}{S, \text{while1}((S', \text{true}), e, s) \rightarrow r'} \text{WHILETRUE1} \quad \frac{S', \text{while } e \text{ do } s \rightarrow r}{S, \text{while2}(S', e, s) \rightarrow r} \text{WHILETRUE2}$$

$$\frac{}{S, \text{while1}((S', \text{false}), e, s) \rightarrow S'} \text{WHILEFALSE}$$

$$\frac{E, H, e \rightarrow r \quad E, H, x =_1 r \rightarrow r'}{E, H, x = e \rightarrow r'} \text{ASG}$$

$$\frac{E' = E[x \mapsto v]}{S, x =_1 (E, H, v) \rightarrow E', H} \text{ASG1} \quad \frac{S, e_1 \rightarrow r \quad S, r; f =_1 e_2 \rightarrow r'}{S, e_1; f = e_2 \rightarrow r'} \text{FLDASG} \quad \frac{S', e \rightarrow r \quad S', f =_2 r \rightarrow r'}{S, (S', f); f =_1 e \rightarrow r'} \text{FLDASG1}$$

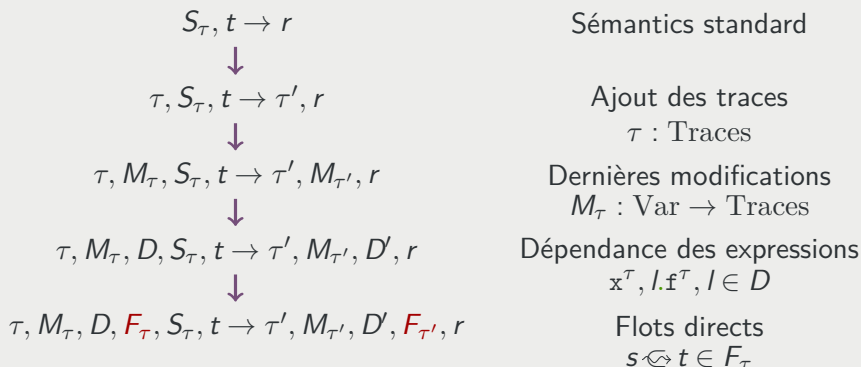
$$\frac{H[l] = o \quad o' = o[f \mapsto v] \quad H' = H[l \mapsto o']}{S, l; f =_2 (E, H, v) \rightarrow E, H'} \text{FLDASG2} \quad \frac{S, e \rightarrow r \quad S, \text{delete1 } r; f \rightarrow r'}{S, \text{delete } e; f \rightarrow r'} \text{DEL}$$

$$\frac{H[l] = o \quad o[f] \neq \perp \quad o' = o[f \mapsto \perp] \quad H' = H[l \mapsto o']}{S, \text{delete1 } (E, H, l); f \rightarrow E, H'} \text{DEL1}$$

$$\frac{}{S, c \rightarrow S, c} \text{CST} \quad \frac{E[x] = v}{E, H, x \rightarrow E, H, v} \text{VAR}$$

$$\frac{S, e_1 \rightarrow r \quad S, r \text{ op }_1 e_2 \rightarrow r'}{S, e_1 \text{ op }_1 e_2 \rightarrow r'} \text{BIN} \quad \frac{S', e_2 \rightarrow r \quad S', v_1 \text{ op }_2 r \rightarrow r'}{S, (S', v_1) \text{ op }_2 e_2 \rightarrow r'} \text{BIN1} \quad \frac{v = v_1 \text{ op }_2 v_2}{S, v_1 \text{ op }_2 (S, v_2) \rightarrow S, v} \text{BIN2}$$

$$\frac{H[l] = \perp \quad H' = H[l \mapsto \{\}] }{E, H, \{\} \rightarrow E, H', l} \text{OBJ} \quad \frac{S, e \rightarrow r \quad S, r; i; f \rightarrow r'}{S, e; f \rightarrow r'} \text{FLD} \quad \frac{H[l] = o \quad o[f] = v}{E, H, (E', H', l); i; f \rightarrow E', H', v} \text{FLD1} \quad \frac{\text{abort}(t_e) = r}{S, t_e \rightarrow r} \text{ABORT}$$



Ces flots  $s \curvearrowright t$  représentent une dépendance entre :

- Une **source** : un objet  $l$ , une variable  $x^\tau$  ou un champ d'un objet  $l.f^\tau$ .
- Une **cible** : une variable  $x^\tau$  ou un champ d'un objet  $l.f^\tau$ .

```
Section LastModified.
```

```
Variable Locations : Annotations.
```

```
Definition ModifiedAnnots := annot_s_r Locations.
```

```
Record LastModifiedHeaps : Type :=
```

```
  makeLastModifiedHeaps {  
    LCEnvironment : heap var ModifiedAnnots;  
    LCHeap : heap loc (heap prop_name ModifiedAnnots)  
  }.
```

```
Definition LastModified :=
```

```
  ConstantAnnotations LastModifiedHeaps.
```

Definition LastModifiedAxiom\_s (r : LastModifiedHeaps)

EHto (R : red\_stat Locations EHto) :=

let LCE := LCEnvironment r in

let LCH := LCHeap r in

let (\_, tau) := extract\_annot\_s R in

match R with

| red\_stat\_ext\_stat\_assign\_1 \_ \_ \_ \_ \_ x \_ \_ =>

let LCE' := write LCE x tau

in makeLastModifiedHeaps LCE' LCH

| red\_stat\_stat\_delete \_ \_ \_ \_ \_ l \_ f \_ \_ \_ \_ \_ =>

let aob := read LCH l

in let LCH' := write LCH l (write aob f tau)

in makeLastModifiedHeaps LCE LCH'

| red\_stat\_ext\_stat\_set\_2 \_ \_ \_ \_ \_ \_ l \_ f \_ \_ \_ \_ \_ =>

let aob := read LCH l

in let LCH' := write LCH l (write aob f tau)

in makeLastModifiedHeaps LCE LCH'

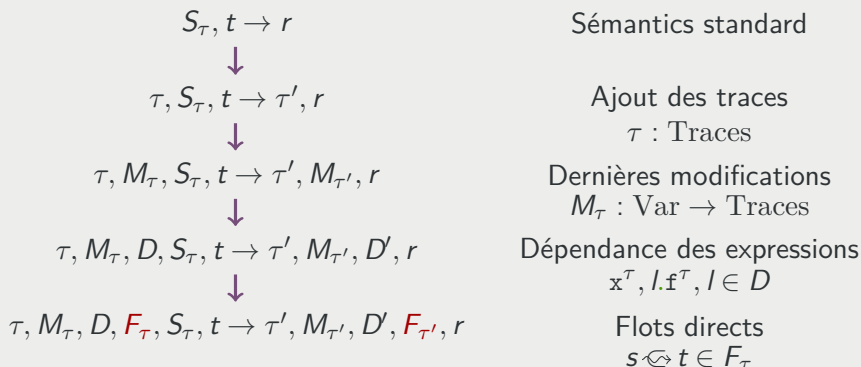
| \_ => makeLastModifiedHeaps LCE LCH

end.

```
Definition annotLastModified :=
  makeIterativeAnnotations LastModified
    (init_e Transmit) (axiom_e Transmit) (up_e Transmit) (down_e
      Transmit) (next_e Transmit)
    (up_s_e Transmit) (next_e_s Transmit)
    (init_s Transmit) LastModifiedAxiom_s (up_s Transmit) (down_s
      Transmit) (next_s Transmit).

End LastModified.
```





Ces flots  $s \curvearrowright t$  représentent une dépendance entre :

- Une **source** : un objet  $l$ , une variable  $x^\tau$  ou un champ d'un objet  $l.f^\tau$ .
- Une **cible** : une variable  $x^\tau$  ou un champ d'un objet  $l.f^\tau$ .

# Correction des annotations

Il reste maintenant à prouver certaines propriétés de correction sur ces annotations.

## Par exemple

Si  $\tau, M_\tau, D, F_\tau, S_\tau, t \rightarrow \tau', M_{\tau'}, D', F_{\tau'}, r$

et  $x^{\tau_1} \rightsquigarrow y^{\tau_2} \in F_{\tau'}$ ,

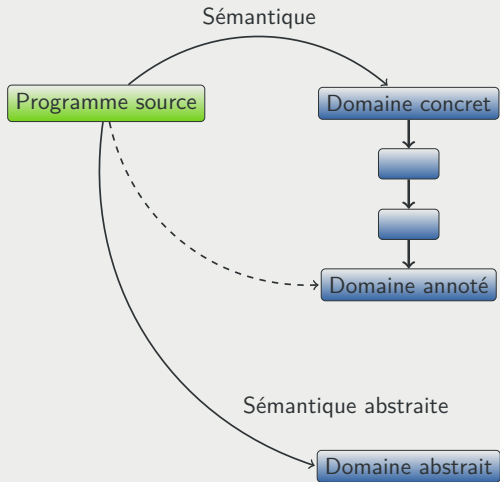
alors  $M_{\tau_2}[x] = \tau_1$ .

« Si F affirme que x au temps  $\tau_1$  flotte dans y au temps  $\tau_2$ , alors x n'a pas été modifié entre-temps. »

Mais cela est facilité car la sémantique annotée et la sémantique standard représentent exactement la même dérivation.

- 1 Annotations
  - Traces
  - Flots directs

- 2 Sémantique abstraite



- Les objets sont abstraits par le point de programme où ils ont été alloués.

$$l^\# \in \text{Loc}^\# = \mathcal{P}(PP)$$

- Les valeurs abstraites sont à la fois des locations abstraites et l'ensemble des variables dont elles dépendent.

$$v^\# \in \text{Val}^\# = \text{Loc}^\# \times \mathcal{P}(\text{Var} \times PP)$$

- L'environnement et le tas stockent les « temps » de dernières modifications.

$$E^\sharp \in \text{Env}^\sharp = \text{Var} \rightarrow (\mathcal{P}(PP) \times \text{Val}^\sharp)$$

$$H^\sharp \in \text{Heap}^\sharp = \text{Loc}^\sharp \rightarrow \text{Field} \rightarrow (\mathcal{P}(PP) \times \text{Val}^\sharp)$$

- Les flots abstraits sont abstraits de la même manière :

$$\text{Store}^\sharp = (\text{Var} \times PP) + (PP \times \text{Field} \times PP) \quad \text{Source}^\sharp = PP + \text{Store}^\sharp$$

$$F \in \text{Dep}^\sharp = \mathcal{P}(\text{Source}^\sharp \times \text{Store}^\sharp)$$

$$\frac{E^\sharp, H^\sharp, e \rightarrow^\sharp v^\sharp, d^\sharp}{E^\sharp, H^\sharp, \underline{x^p = e} \rightarrow^\sharp E^\sharp [x \mapsto (\{p\}, v^\sharp)], H^\sharp, (v_l^\sharp \cup d^\sharp) \hookrightarrow^\sharp x^p} \text{ASG}$$

$$\frac{\begin{array}{l} E^\sharp \sqsubseteq E_0^\sharp \quad H^\sharp \sqsubseteq H_0^\sharp \\ E_0^\sharp, H_0^\sharp, s \rightarrow^\sharp E_1^\sharp, H_1^\sharp, F^\sharp \\ E_1^\sharp \sqsubseteq E_0^\sharp \quad H_1^\sharp \sqsubseteq H_0^\sharp \end{array}}{E^\sharp, H^\sharp, \underline{\text{while } e \text{ do } s} \rightarrow^\sharp E_0^\sharp, H_0^\sharp, F^\sharp} \text{WHILE}$$

On définit une relation d'abstraction  $\prec$  entre

- Traces et  $PP$ ,
- Heap et  $\text{Heap}^\sharp$ ,
- ...

Theorem (En cours)

Si

$$\square, \emptyset, \square, \emptyset, \underline{s} \rightarrow \tau, M_\tau, F_\tau, E_\tau, H_\tau$$

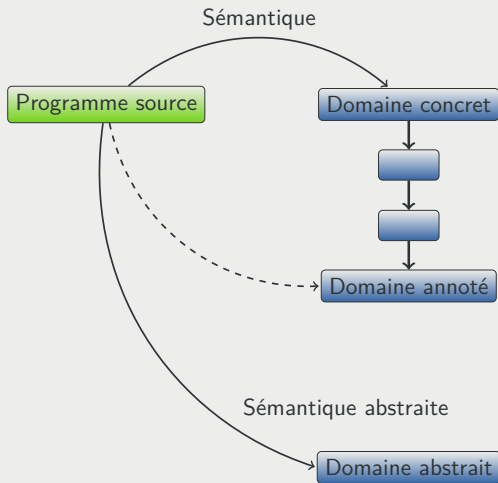
et

$$\perp, \perp, \underline{s} \rightarrow^\sharp E^\sharp, H^\sharp, F^\sharp$$

alors  $E \prec E^\sharp$ ,  $H \prec H^\sharp$  et  $F_\tau \prec F^\sharp$ .

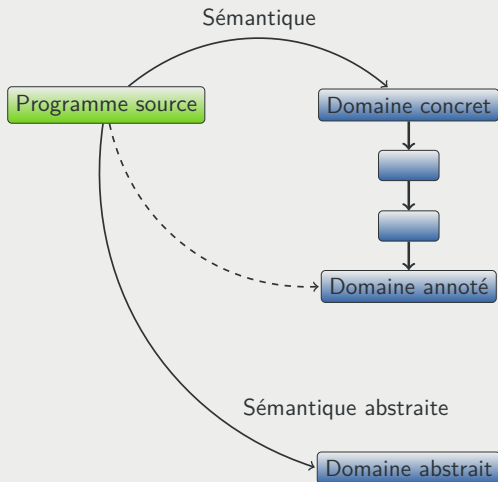
# Conclusion

- Une architecture générale pour instrumenter une sémantique.
- Passe à l'échelle.
- Formalisé en Coq.





Merci pour votre écoute !



$$\frac{E[x] = v \quad M[x] = \tau_0}{\tau, M, \mathbf{d}, E, H, \underline{x} \rightarrow \tau', M, \mathbf{d} \cup \{x^{\tau_0}\}, E, H, v} \text{VAR} \qquad \frac{H[l] = \perp \quad H = H[l \mapsto \{\}] \quad M' = M[l \mapsto \tau']}{\tau, M, \mathbf{d}, E, H, \underline{\{ \}} \rightarrow \tau', M, \mathbf{d} \cup \{l^{\tau'}\}, E, H', l} \text{OBJ}$$

$$\frac{H[l] = o \quad o[f] = v \quad M[(l, M[l], f)] = \tau_0}{\tau, M, \mathbf{d}, E, H, \underline{(l, H', f)} \cdot f \rightarrow \tau', M, \mathbf{d} \cup \{(l, M[l], f)^{\tau_0}\}, E', H', v} \text{FLD1}$$

$$\frac{\tau_1, M, \emptyset, S, e \rightarrow \tau_2, M', \mathbf{d}, r \quad \tau_3, M', \emptyset, S, \text{ifl}(r, s_1, s_2) \rightarrow \tau_4, M'', \emptyset, r'}{\tau_0, M, \emptyset, S, \text{if } e \text{ then } s_1 \text{ else } s_2 \rightarrow \tau_5, M'', \emptyset, r'} \text{IF}$$

$$\frac{\tau_1, M, \emptyset, S, e \rightarrow \tau_2, M', \mathbf{d}, r \quad \tau_3, M', \emptyset, S, \text{while1}(r, x, s) \rightarrow \tau_4, M'', \emptyset, r'}{\tau_0, M, \emptyset, S, \text{while } e \text{ do } s \rightarrow \tau_5, M'', \emptyset, r'} \text{WHILE}$$

$$\frac{\tau_1, M, \emptyset, S, e \rightarrow \tau_2, M', \mathbf{d}, r \quad \tau_3, M', \mathbf{d}, S, \underline{x} =_1 r \rightarrow \tau_4, M'', \emptyset, r'}{\tau_0, M, \emptyset, S, \underline{x} = e \rightarrow \tau_5, M'', \emptyset, r'} \text{ASG} \qquad \frac{E' = E[x \mapsto v] \quad M' = M[x \mapsto \tau']}{\tau, M, \mathbf{d}, S, \underline{x} =_1 (E, H, v) \rightarrow \tau', M', \emptyset, E', H} \text{ASG1}$$

$$\frac{\tau_1, M, \emptyset, S, e_1 \rightarrow \tau_2, M', \mathbf{d}, r \quad \tau_3, M', \emptyset, S, r.f =_1 e_2 \rightarrow \tau_4, M'', \emptyset, r'}{\tau_0, M, \emptyset, S, \underline{e_1.f} = e_2 \rightarrow \tau_5, M'', \emptyset, r'} \text{FLDASG}$$

$$\frac{\tau_1, M, \emptyset, S', e \rightarrow \tau_2, M', \mathbf{d}, r \quad \tau_3, M', \mathbf{d}, S', l.f =_2 r \rightarrow \tau_4, M'', \emptyset, r'}{\tau_0, M, \emptyset, S, \underline{(S', x).f} =_1 e \rightarrow \tau_5, M'', \emptyset, r'} \text{FLDASG1}$$

$$\frac{H[l] = o \quad o' = o[f \mapsto v] \quad H = H[l \mapsto o'] \quad M' = M[(l, M[l], f) \mapsto \tau']}{\tau, M, \mathbf{d}, S, \underline{l.f} =_2 (E, H, v) \rightarrow \tau', M', \emptyset, E, H'} \text{FLDASG2}$$

$$\frac{\tau_1, M, \emptyset, S, e \rightarrow \tau_2, M', \mathbf{d}, r \quad \tau_3, M', \emptyset, S, \text{delete1 } r.f \rightarrow \tau_4, M'', \emptyset, r'}{\tau_0, M, \emptyset, S, \underline{\text{delete } e.f} \rightarrow \tau_5, M'', \emptyset, r'} \text{DELETE}$$

- 1 Annotations
  - Traces
  - Flots directs

- 2 Sémantique abstraite