Analyse efficace de réseaux de Petri par des techniques de compilation

Mise en œuvre avec LLVM

Lukasz Fronc

Encadrant : Franck Pommereau

Laboratoire IBISC, Université d'Évry

La vérification automatique : *Model-ckecking*

- Domaine majeur de recherche
  - Turing Award 2007 — E. Clarke, A. Emerson, J. Sifakis
- importance : systèmes critiques
  - transports — pilotage des avions, des trains
  - production de l'énergie — centrales nucléaires
  - applications militaires
  - ...
- Enjeux : des sytèmes toujours plus complexes

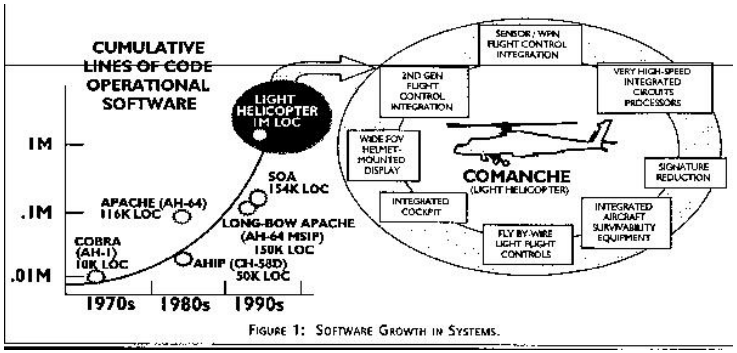


FIGURE 1: SOFTWARE GROWTH IN SYSTEMS.

Deux approches principales

- model-checking explicite
  - énumération de tous les états
  - possibilités de simulation
- model-checking symbolique
  - raisonnements sur les ensembles d'états

Explosion de l'espace d'états

- problème inhérent au model-checking dans les deux approches
- conjugué à l'augmentation de taille des modèles

Palliatifs

réductions, abstractions, calcul parallèle, compilation, ...

Deux approches principales

- model-checking explicite
  - énumération de tous les états
  - possibilités de simulation
- model-checking symbolique
  - raisonnements sur les ensembles d'états

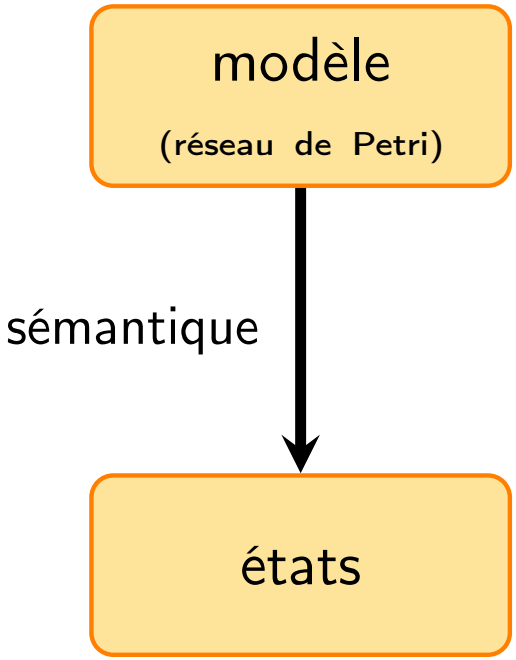
Explosion de l'espace d'états

- problème inhérent au model-checking dans les deux approches
- conjugué à l'augmentation de taille des modèles

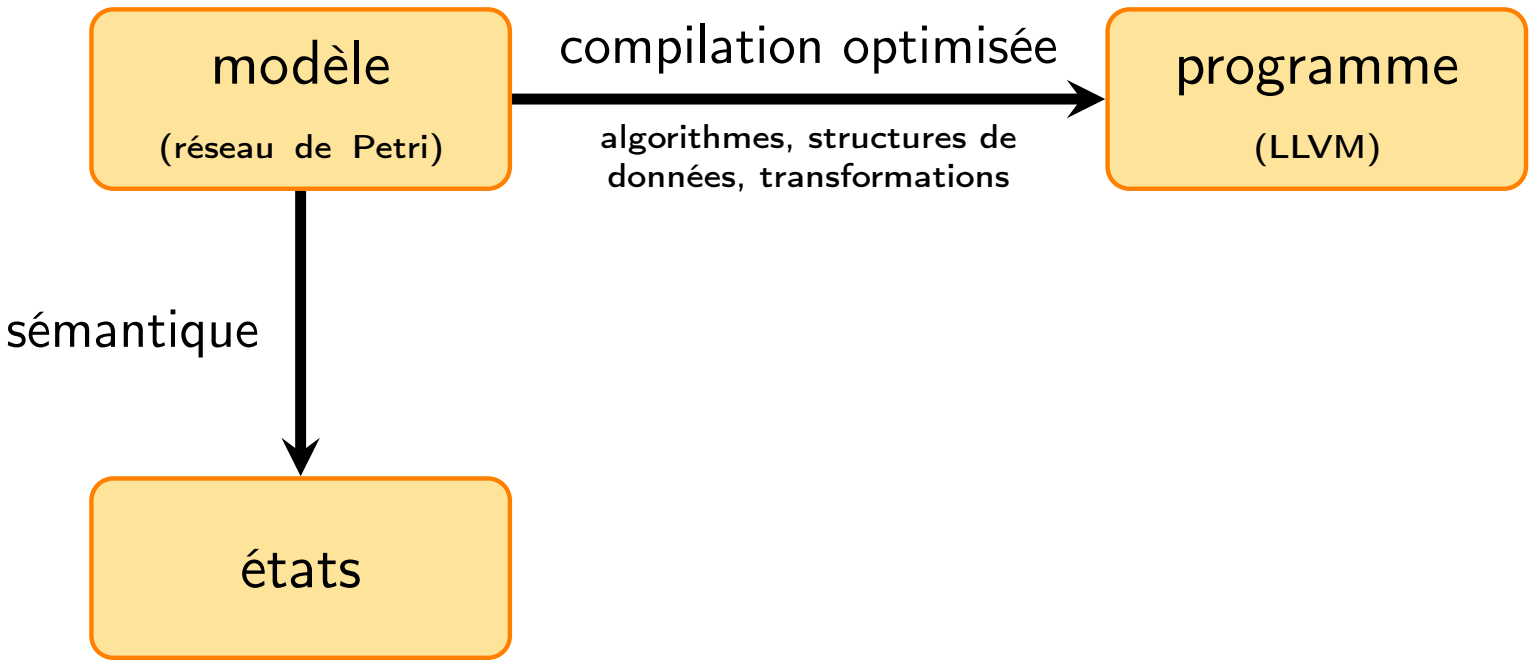
Palliatifs

réductions, abstractions, calcul parallèle, compilation, ...

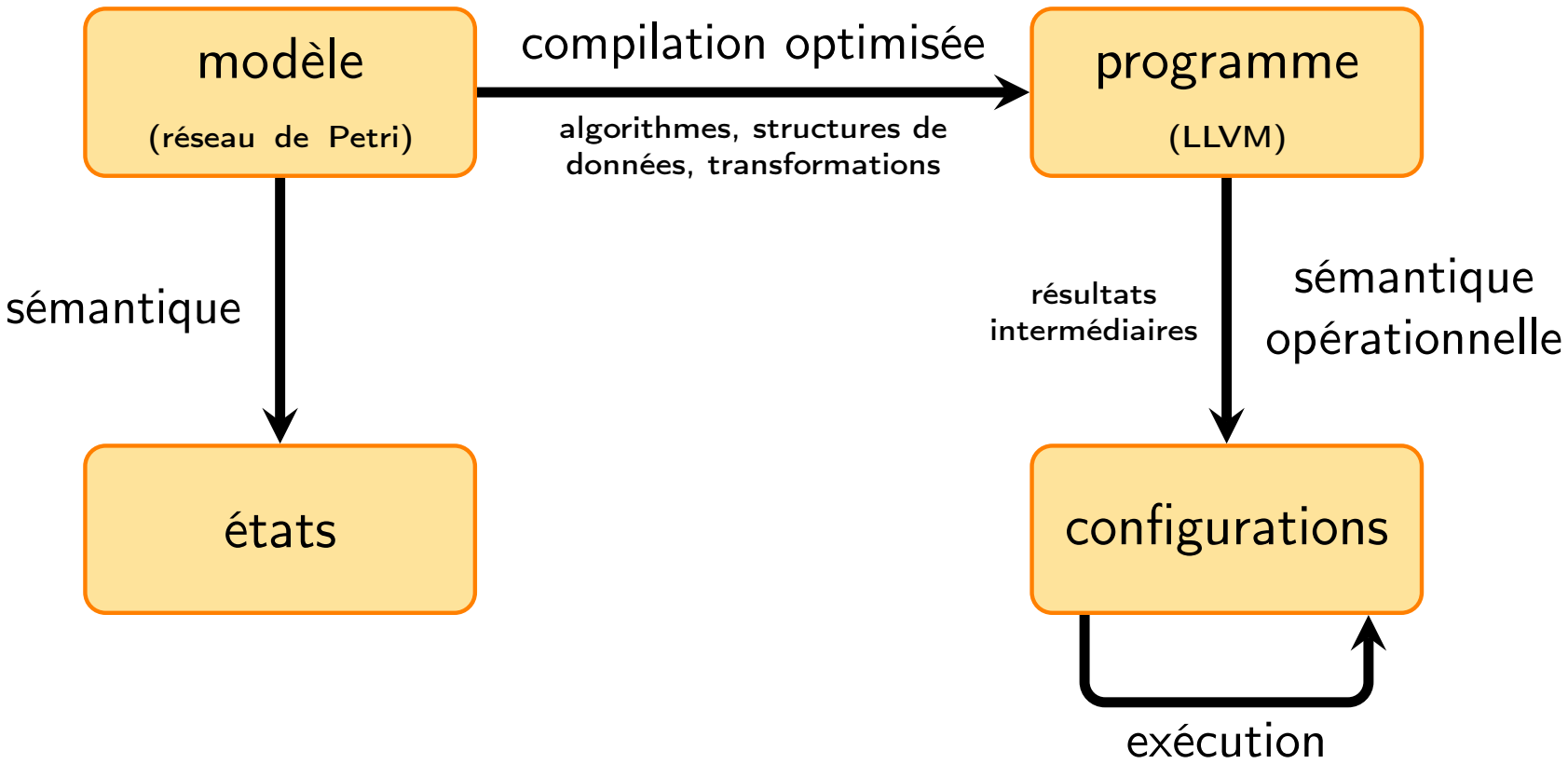
Contribution pour accélérer le calcul de l'espace d'états



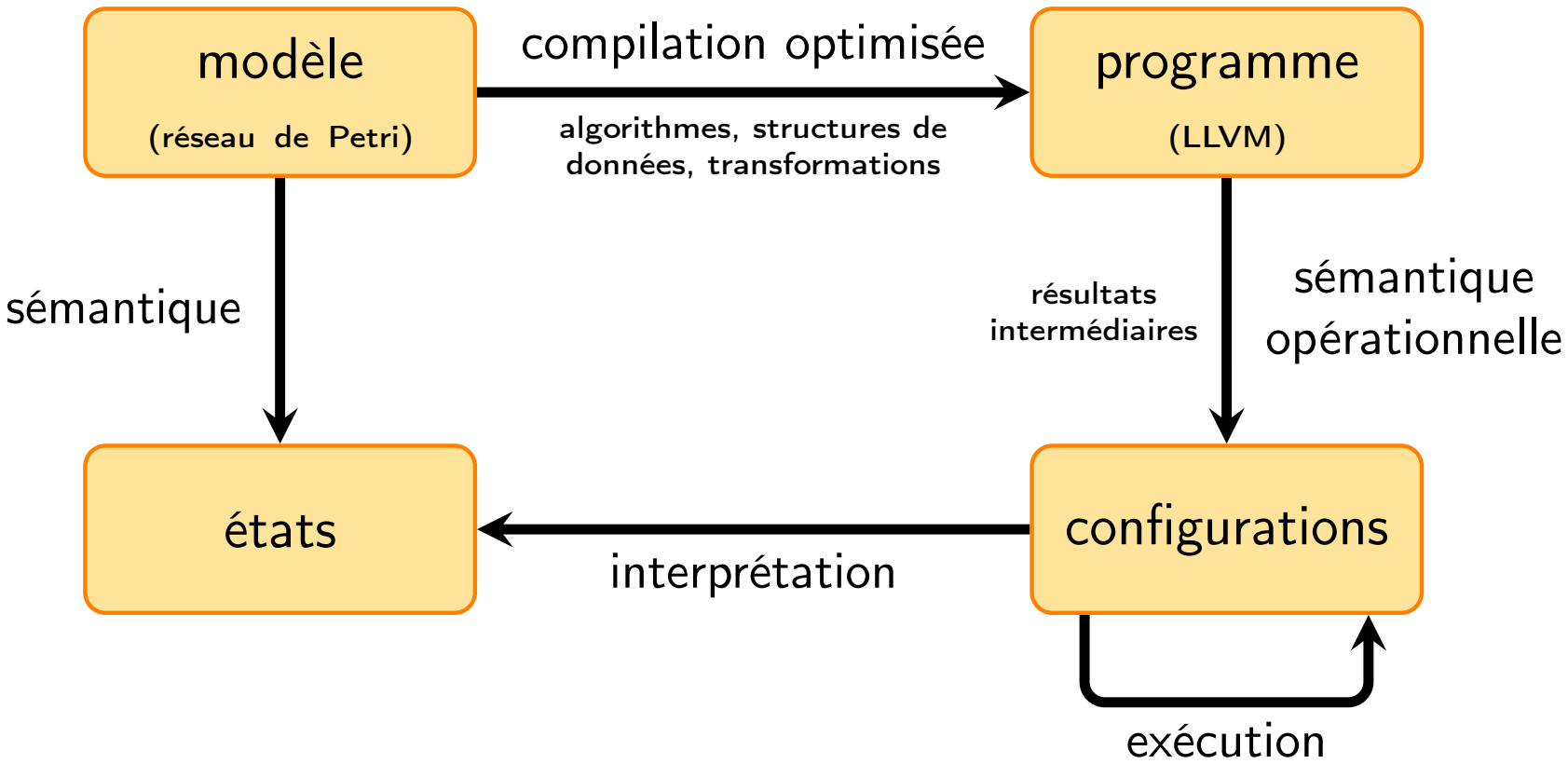
Contribution pour accélérer le calcul de l'espace d'états



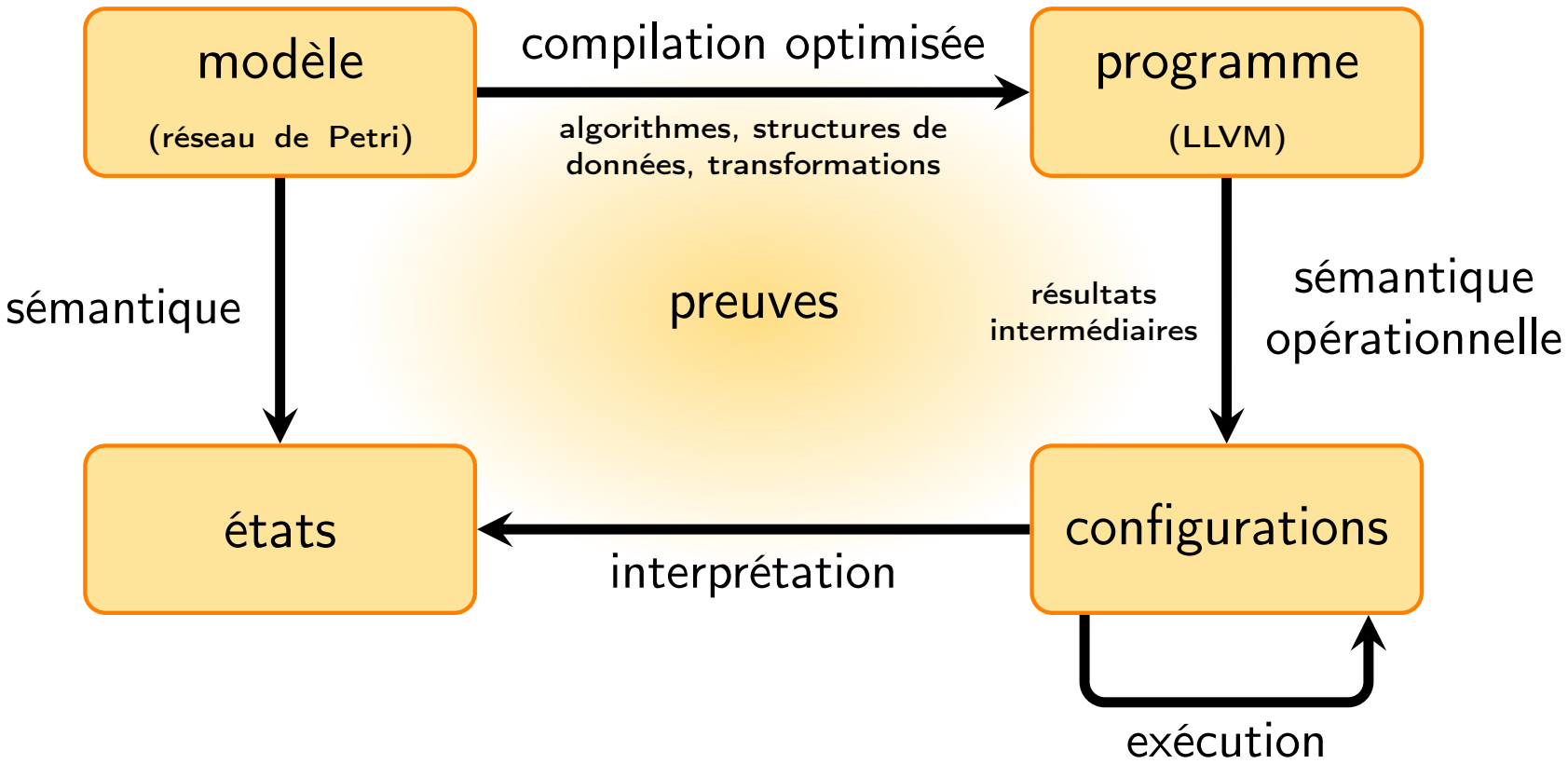
Contribution pour accélérer le calcul de l'espace d'états



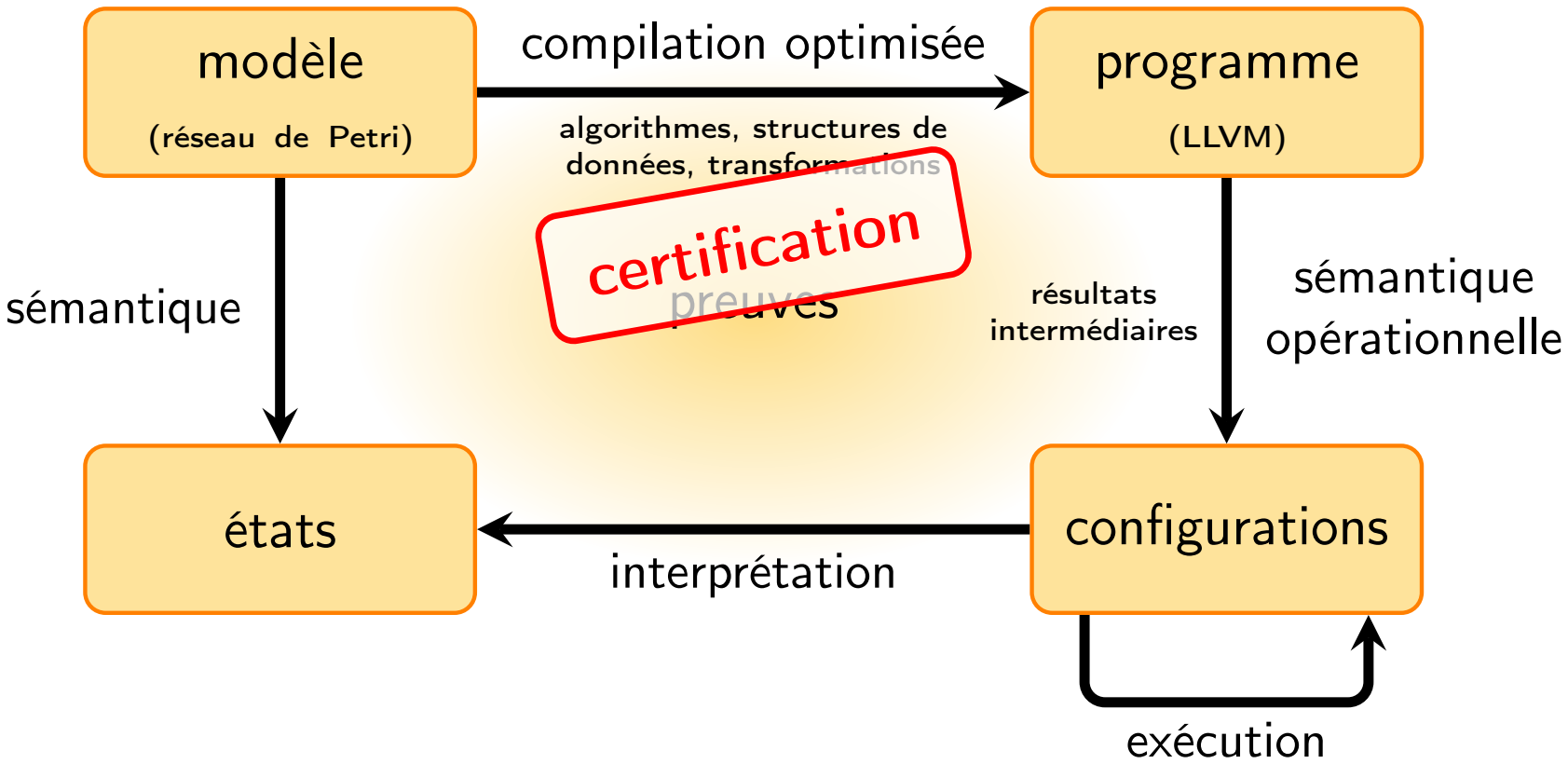
Contribution pour accélérer le calcul de l'espace d'états



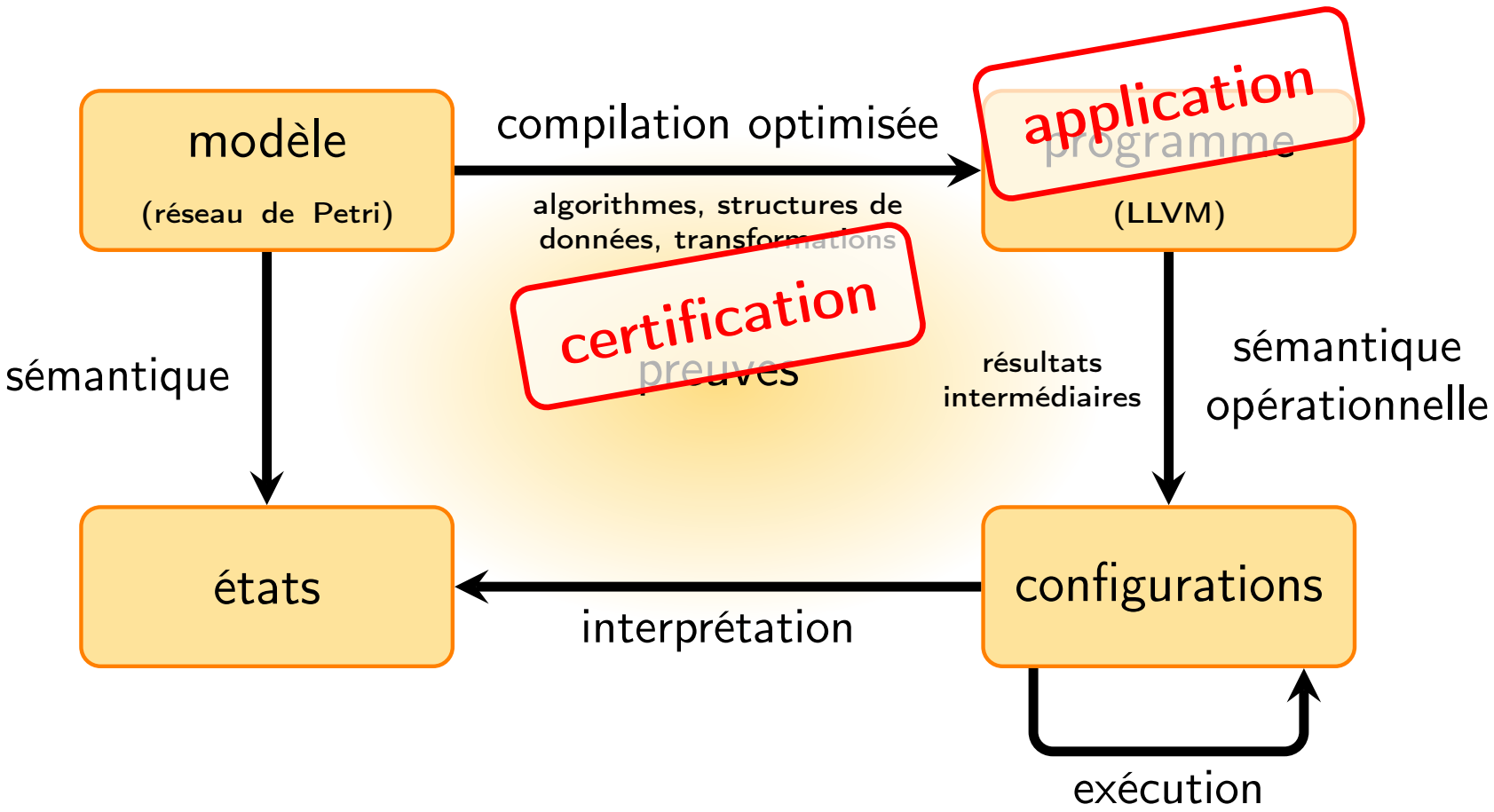
Contribution pour accélérer le calcul de l'espace d'états



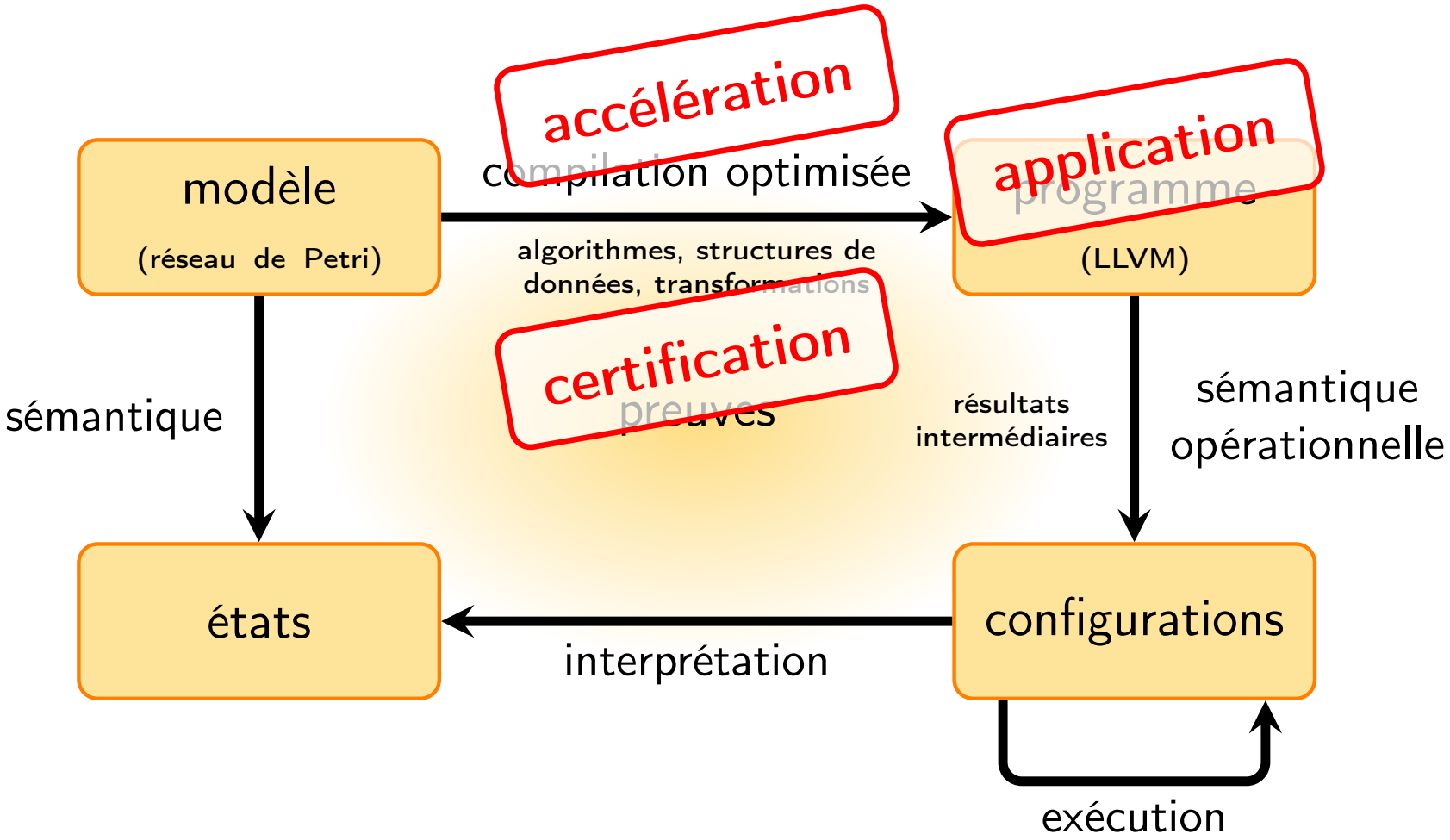
Contribution pour accélérer le calcul de l'espace d'états



Contribution pour accélérer le calcul de l'espace d'états



Contribution pour accélérer le calcul de l'espace d'états



Plan de l'exposé

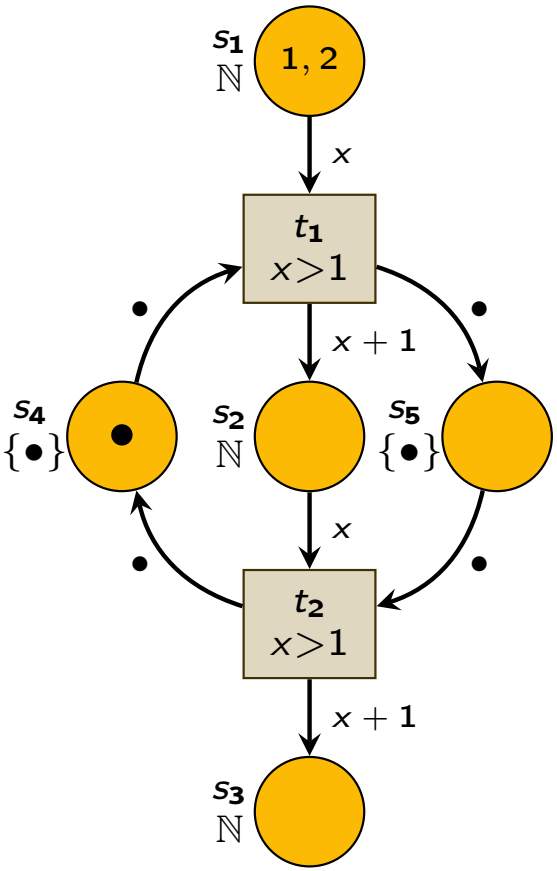
Contexte : compilation de réseaux de Petri

Contribution
- Architecture
- Sémantique de LLVM

Résultats

Conclusion

Compilation de réseaux de Petri



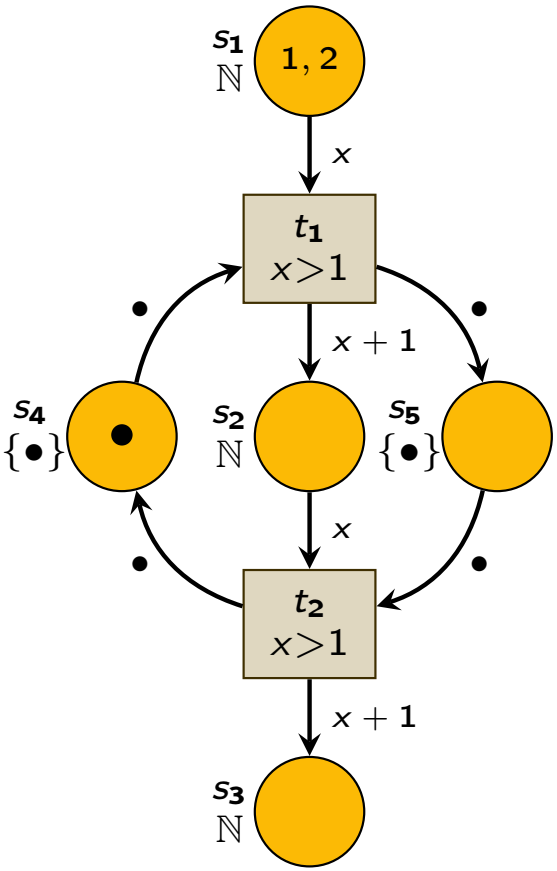
- ◇ utilisé par des outils réputés — Helena, Spin, ...
- ◇ disparition des structures de données — places, arcs, ...

  reste : marquages et fonctions
- ◇ compilations *vs* interprétation des expressions

  gardes, ...

- ▶ simplification, efficacité

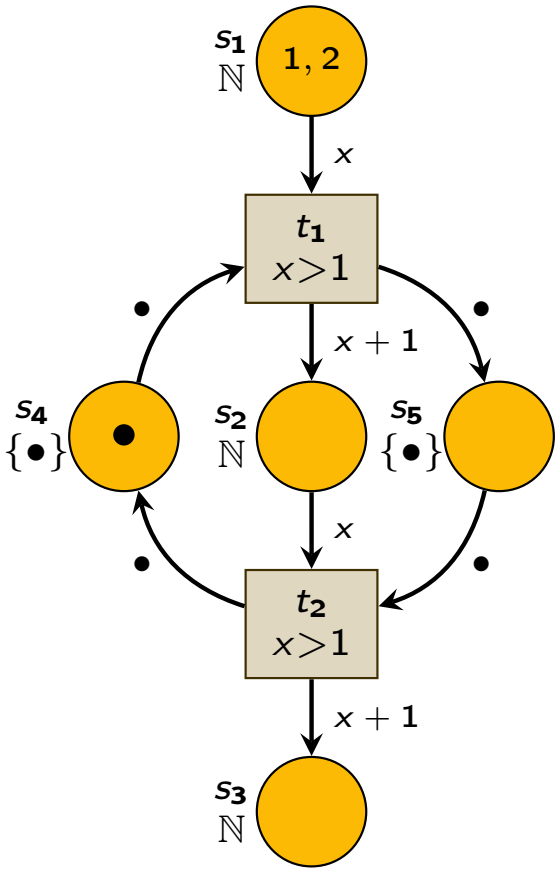
Compilation de réseaux de Petri



fire(M : *Marking*, t : *Trans*, m : *Mode*) : *Marking*

$M' \leftarrow copy(M)$
for *place* **in** $pre(t)$ **do**
  $arc \leftarrow getarc(place, t)$
  $M' \leftarrow M' - eval(arc, m)$
endfor
for *place* **in** $post(t)$ **do**
  $arc \leftarrow getarc(t, place)$
  $M' \leftarrow M' + eval(arc, m)$
endfor
return M'

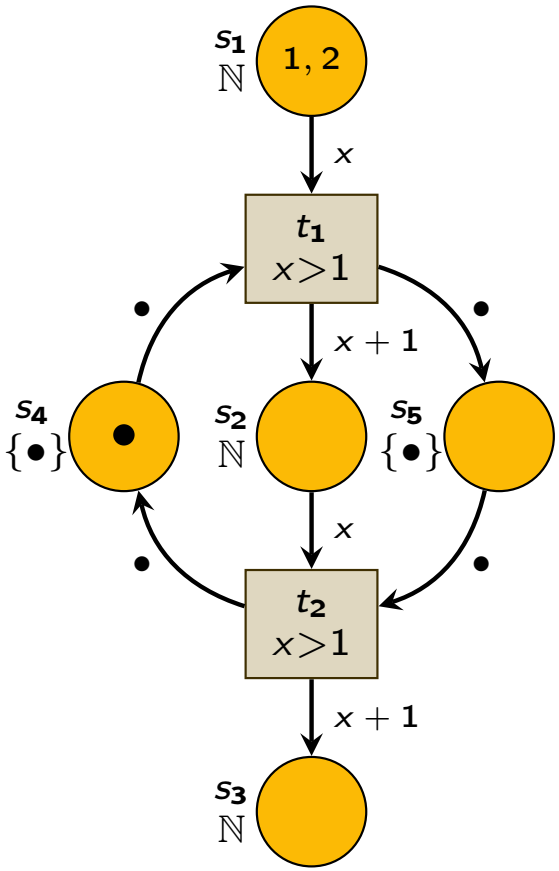
Compilation de réseaux de Petri



fire$_{t_1}(M : \mathit{Marking}, x : \mathbb{N}) : \mathit{Marking}$

$M' \leftarrow \mathit{copy}(M)$
$M'(s_1) \leftarrow M(s_1) - \{x\}$
$M'(s_4) \leftarrow M(s_4) - \{\bullet\}$
$M'(s_2) \leftarrow M(s_2) + \{x+1\}$
$M'(s_5) \leftarrow M(s_5) + \{\bullet\}$
return M'

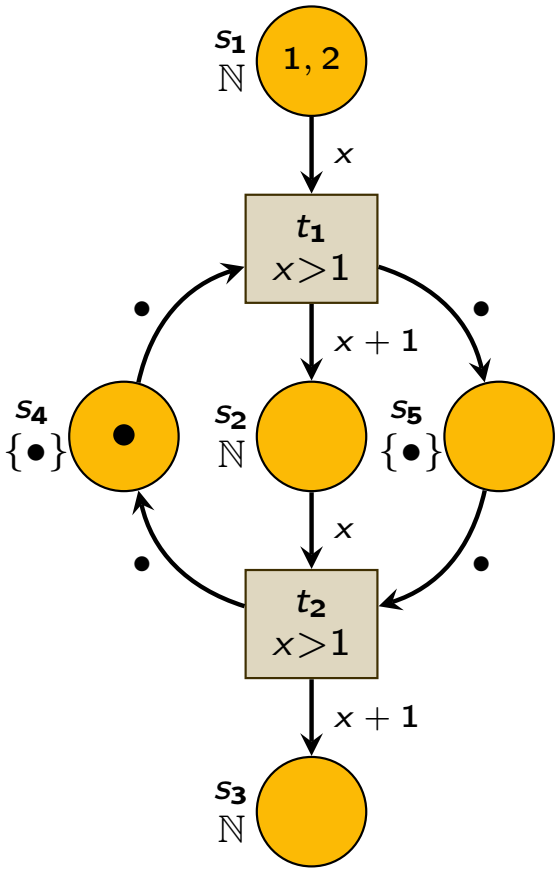
Compilation de réseaux de Petri



succ$_{t_1}$ $(M : \mathit{Marking}) : \mathit{MarkingSet}$

$\mathit{next} \leftarrow \emptyset$
for x **in** $M(s_1)$ **do**
  for token_{s_4} **in** $M(s_4)$ **do**
    if $x > 1$ **then**
      $\mathit{next} \leftarrow \mathit{next} \cup \{\mathit{fire}_{t_1}(M, x)\}$
    endif
  endfor
endfor
return next

Optimisations selon les spécificités du modèle



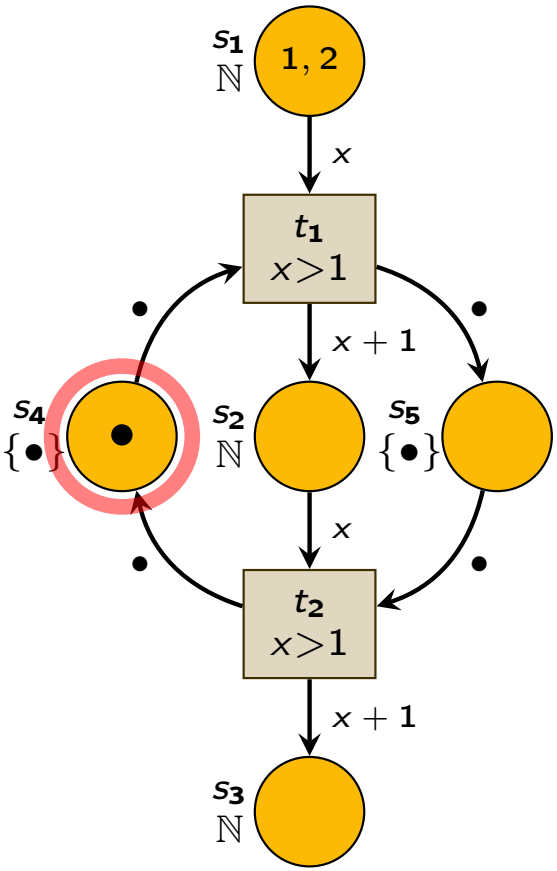
- modifier les algorithmes

  ex : boucles transformées en tests

- sélectionner les structures de données adaptés

  ex : encodages efficaces en calcul et en mémoire

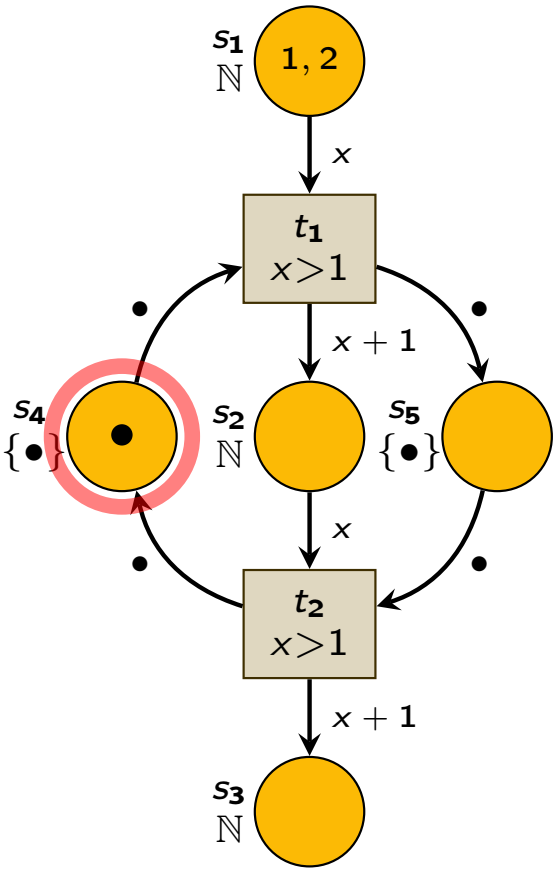
Optimisations selon les spécificités du modèle



succ$_{t_1}$ $(M : Marking) : Set$

```
next ← ∅
for x in M(s1) do
    for token_s4 in M(s4) do
        if x > 1 then
            next ← next ∪ {fire_t1(M, x)}
        endif
    endfor
endfor
return next
```

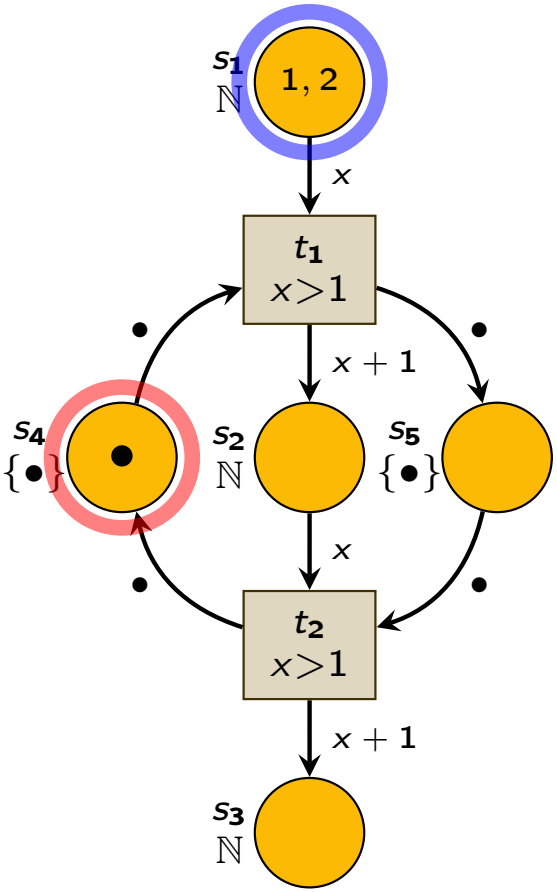
Optimisations selon les spécificités du modèle



succ$_{t_1}$ $(M : Marking) : Set$

```
next ← ∅
for x in M(s1) do
    if notempty(M(s4)) then
        if x > 1 then
            next ← next ∪ {fire_t1(M, x)}
        endif
    endif
endfor
return next
```

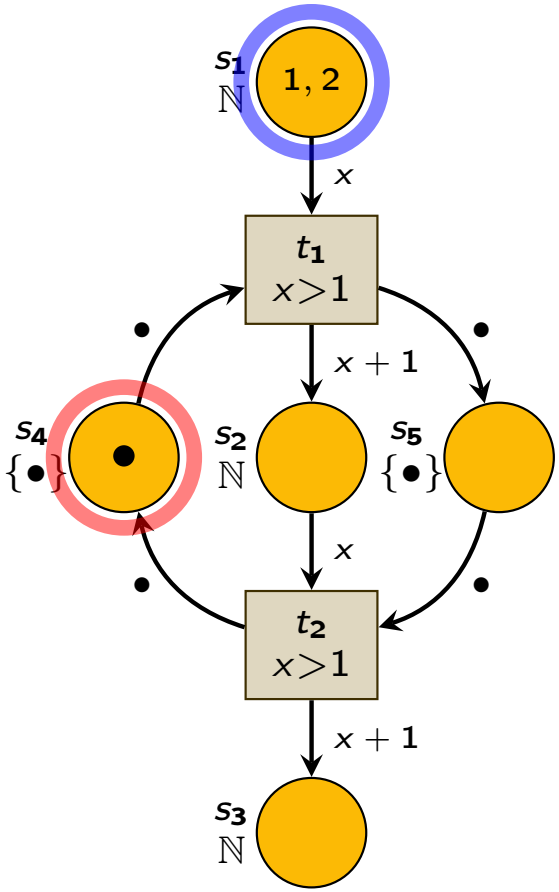
Optimisations selon les spécificités du modèle



succ$_{t_1}$ $(M : Marking) : Set$

```
next ← ∅
for x in M(s1) do
    if notempty(M(s4)) then
        if x > 1 then
            next ← next ∪ {fire_t1(M, x)}
        endif
    endif
endfor
return next
```

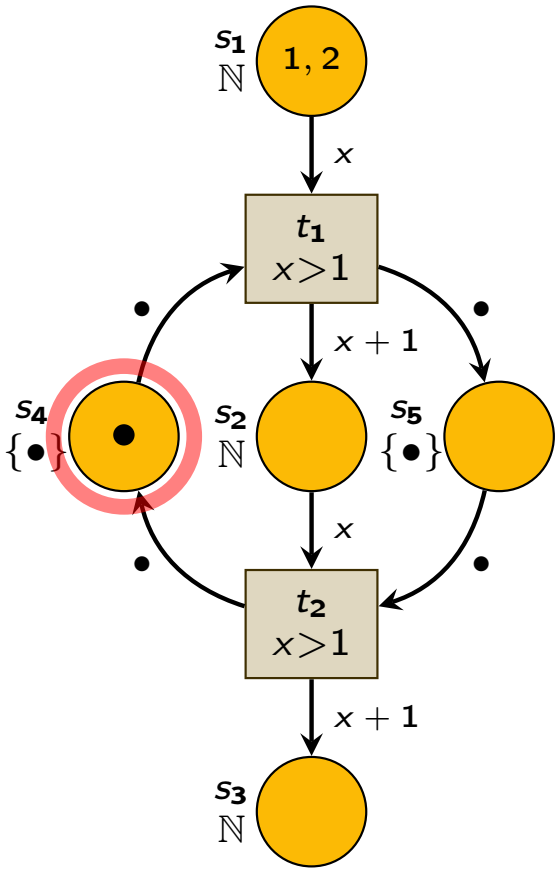
Optimisations selon les spécificités du modèle



succ$_{t_1}$ $(M : Marking) : Set$

```
next ← ∅
if notempty(M(s4)) then
    for x in M(s1) do
        if x > 1 then
            next ← next ∪ {fire_t1(M, x)}
        endif
    endfor
endif
return next
```

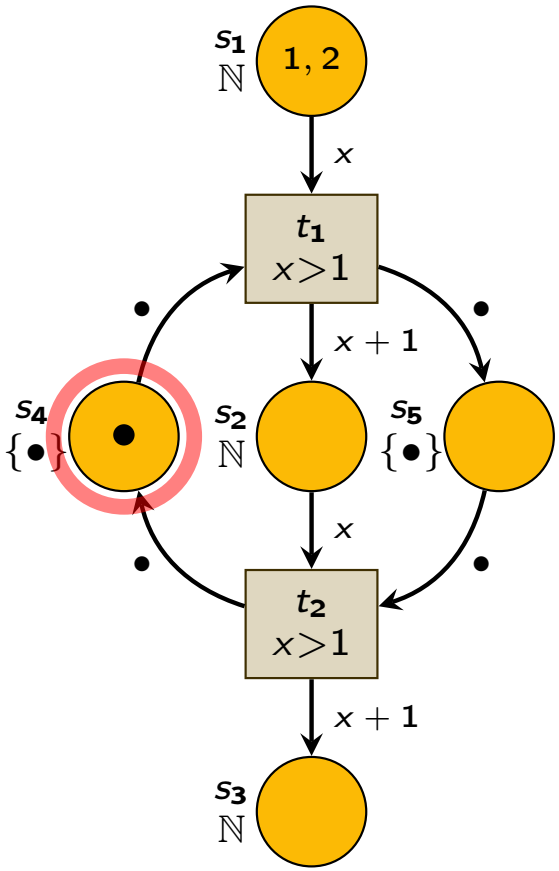
Optimisations selon les spécificités du modèle



succ$_{t_1}$ $(M : Marking) : Set$

```
next ← ∅
if notempty(M(s4)) then
    for x in M(s1) do
        if x > 1 then
            next ← next ∪ {fire_t1(M, x)}
        endif
    endfor
endif
return next
```

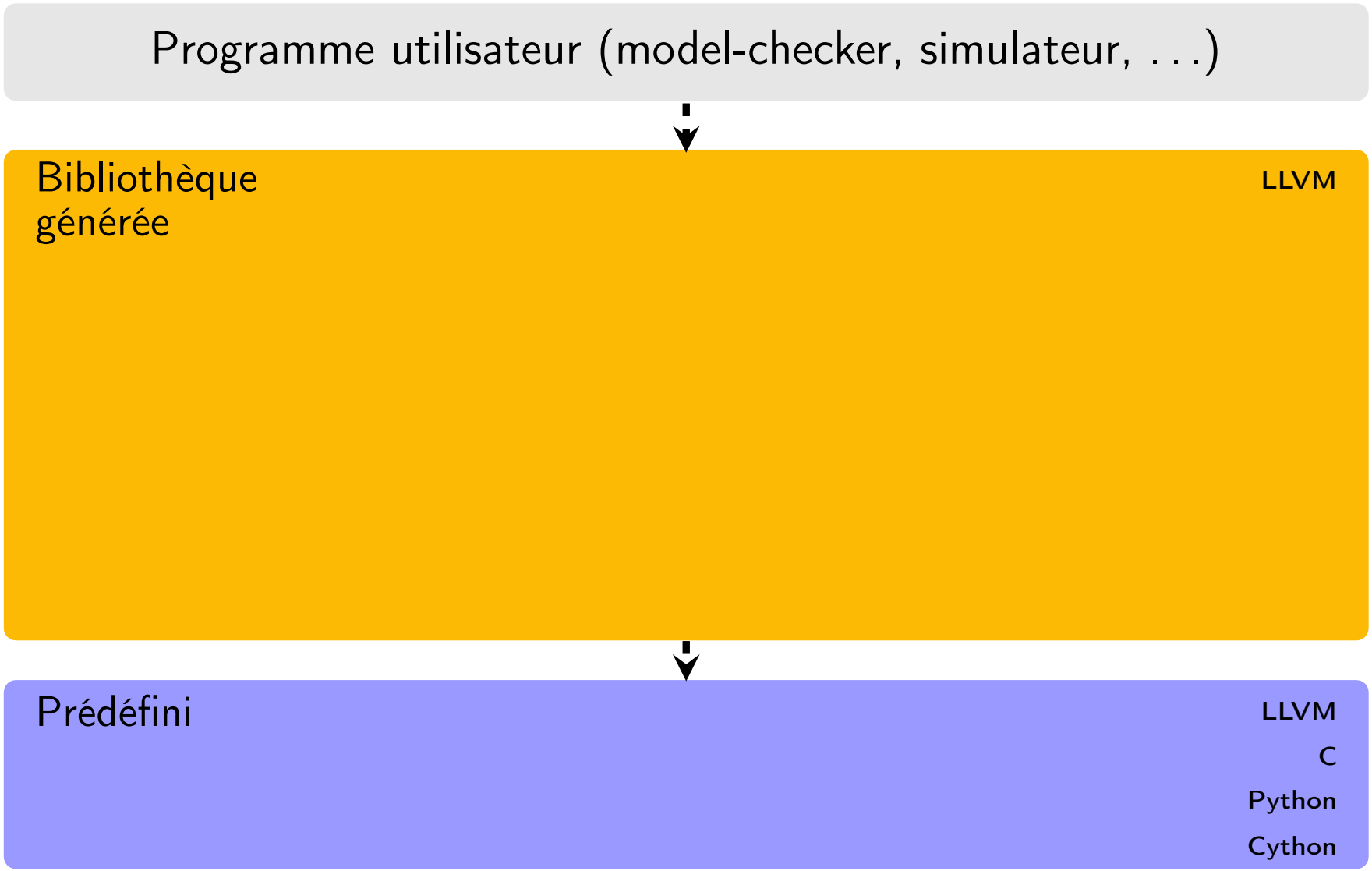
Optimisations selon les spécificités du modèle



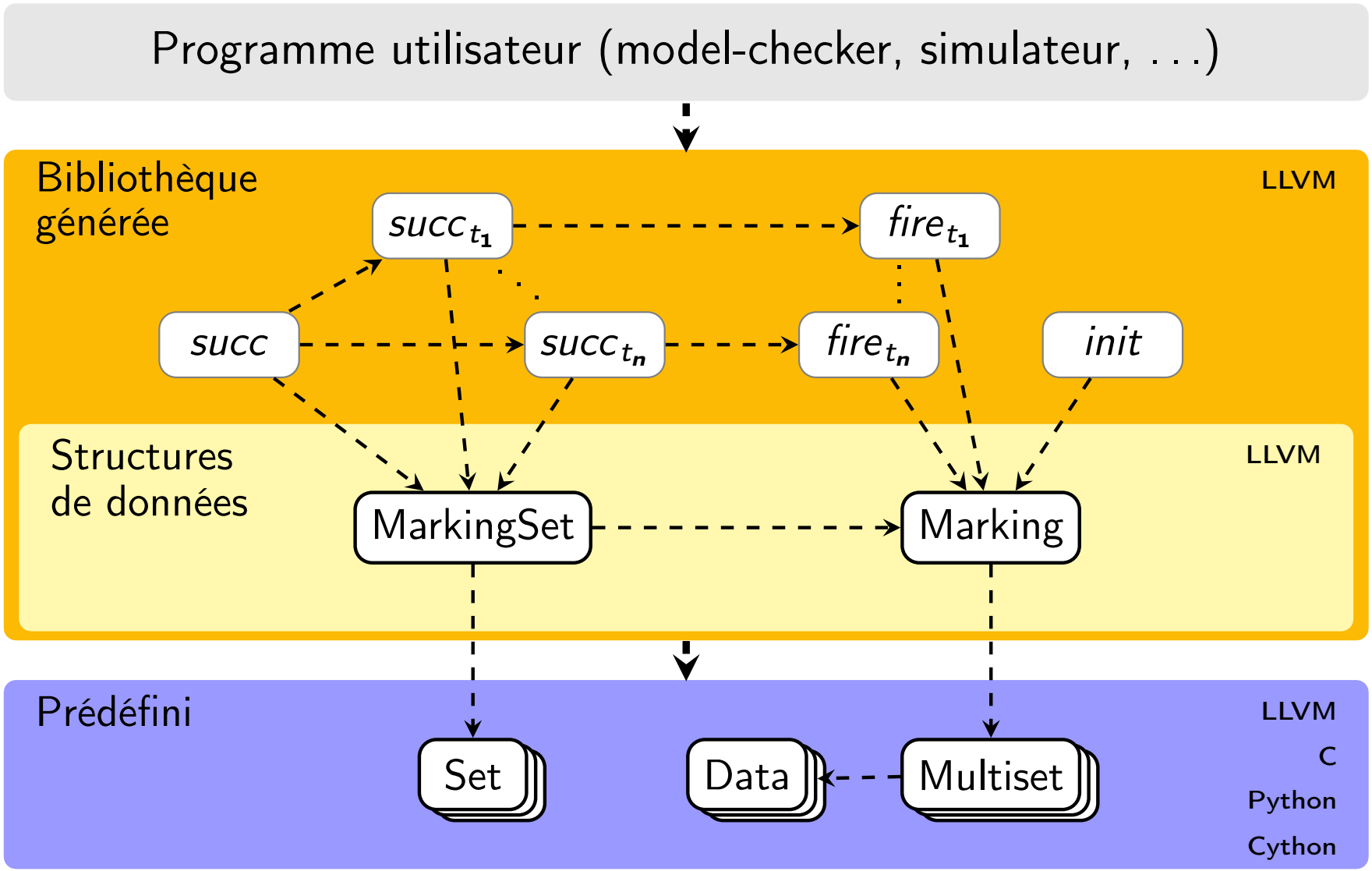
succ$_{t_1}$ $(M : Marking) : Set$

```
next ← ∅
if M(s4) then
    for x in M(s1) do
        if x > 1 then
            next ← next ∪ {fire_t1(M, x)}
        endif
    endfor
endif
return next
```

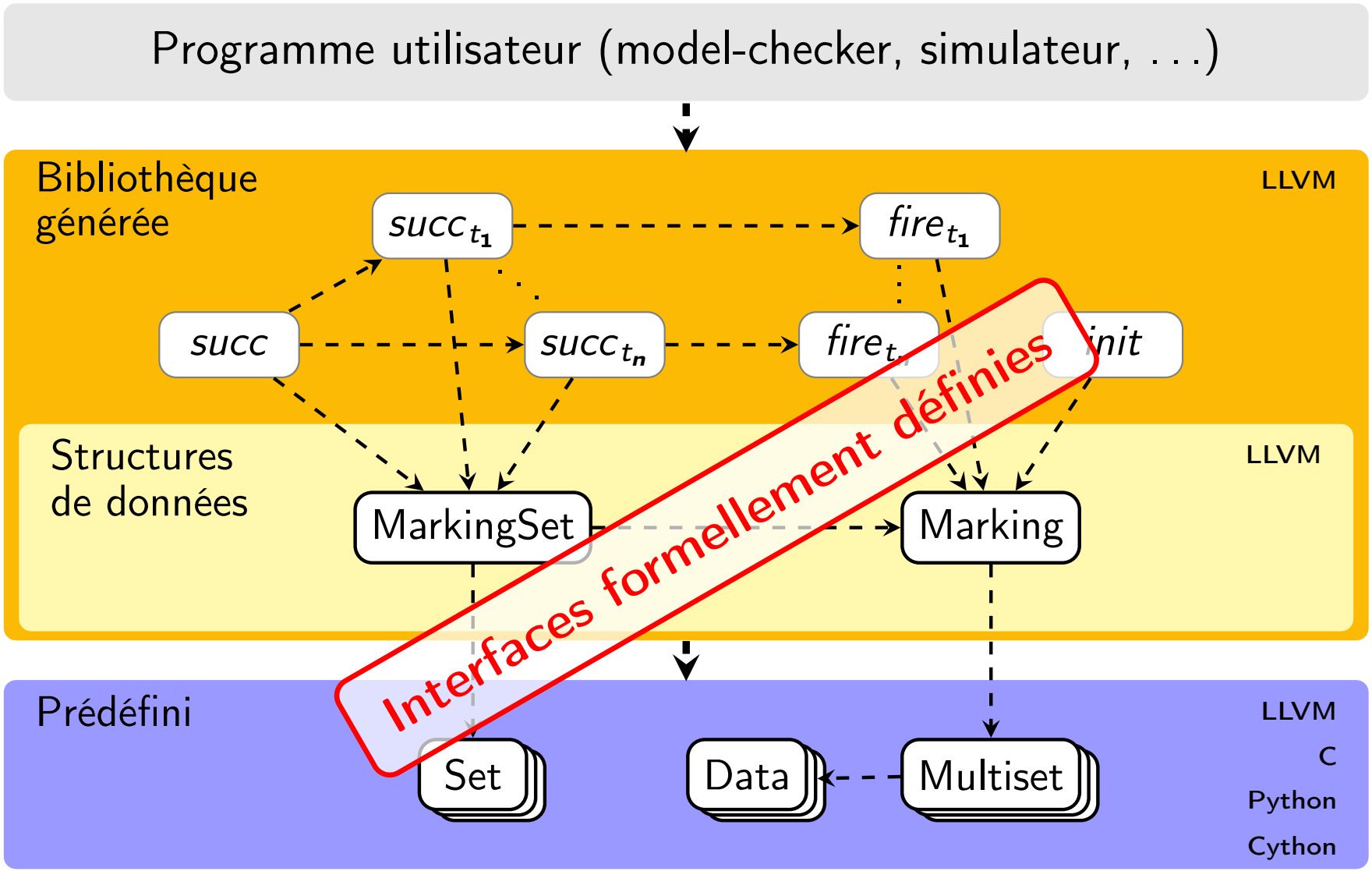
Structure de la proposition : bibliothèque et cadre formel



Structure de la proposition : bibliothèque et cadre formel



Structure de la proposition : bibliothèque et cadre formel



Low Level Virtual Machine

Une boite à outils de compilation

- initialement projet de recherche — Université d'Illinois (2003)
- adopté par l'industrie — Sun, Cray, Apple, Google, ...
- adopté par la recherche académique
- moderne, modulaire et multi-platefome
- langage d'assemblage de haut niveau — typé, multi-plateforme
- optimisations génériques — allocation de registres, ...

Contribution

- identification d'un fragment de LLVM — cible du compilateur
- sémantique opérationnelle — 15 règles

Syntaxe, programmes, mémoire

Syntaxe :

$$
\begin{array}{lll}
\mathit{seq} ::= \mathit{expr} & \mathit{expr} ::= \mathit{alloc}\ \mathit{type} & \mathit{cmd} ::= \mathit{br}\ \mathit{rvalue},\ \mathit{label},\ \mathit{label} \\
\quad \mid \mathit{cmd} & \quad \mid \mathit{add}\ \mathit{rvalue},\ \mathit{rvalue} & \quad \mid \mathit{var} = \mathit{expr} \\
\quad \mid \mathit{cmd};\ \mathit{seq} & \quad \mid \ldots & \quad \mid \ldots
\end{array}
$$

Syntaxe, programmes, mémoire

Syntaxe :

$$
\begin{array}{lll}
seq ::= expr & expr ::= alloc\ type & cmd ::= br\ rvalue,\ label,\ label \\
\quad \mid\ cmd & \quad \mid\ add\ rvalue,\ rvalue & \quad \mid\ var = expr \\
\quad \mid\ cmd;\ seq & \quad \mid\ \ldots & \quad \mid\ \ldots
\end{array}
$$

Programme :

$$P : label \rightarrow seq$$

$$f_1(\!| a_1, \ldots, a_n |\!) \mapsto sq_1 \quad br\ lbl_2$$

$$lbl_2 \mapsto sq_2$$

$$\vdots \quad \vdots \quad \vdots$$

Syntaxe, programmes, mémoire

Syntaxe :

$$
\begin{array}{lll}
\mathit{seq} ::= \mathit{expr} & \mathit{expr} ::= \mathit{alloc}\ \mathit{type} & \mathit{cmd} ::= \mathit{br}\ \mathit{rvalue},\ \mathit{label},\ \mathit{label} \\
\quad \mid \mathit{cmd} & \quad \mid \mathit{add}\ \mathit{rvalue},\ \mathit{rvalue} & \quad \mid \mathit{var} = \mathit{expr} \\
\quad \mid \mathit{cmd};\ \mathit{seq} & \quad \mid \ldots & \quad \mid \ldots
\end{array}
$$

Programme :

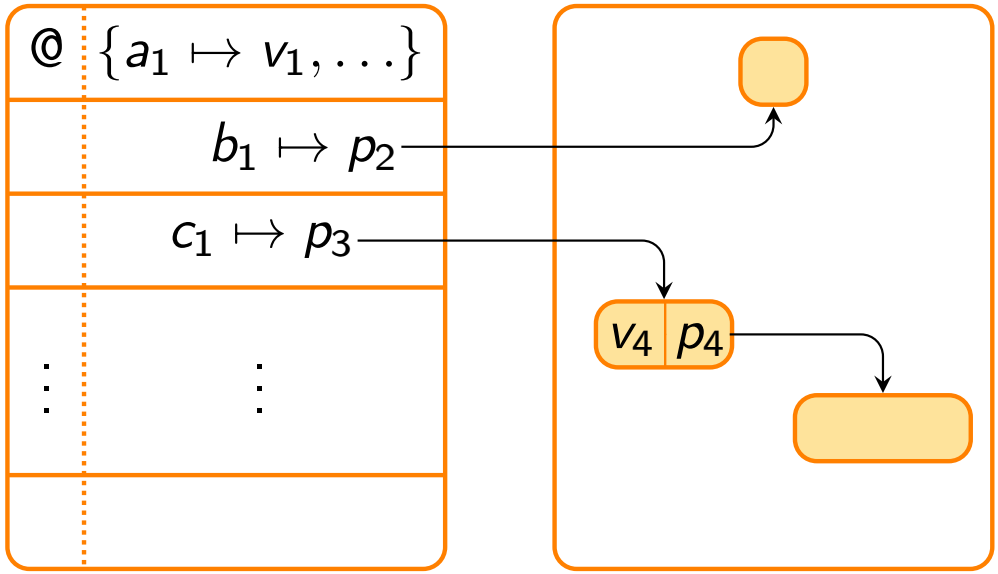
$$P : \mathit{label} \rightarrow \mathit{seq}$$

$$f_1(\!| a_1, \ldots, a_n |\!) \mapsto sq_1 \quad \textit{br } \mathit{lbl}_2$$

$$\mathit{lbl}_2 \mapsto sq_2$$

$$\vdots \quad \vdots \quad \vdots$$

Pile et tas :



Sémantique opérationnelle : exemple

$$\frac{\begin{array}{c} f(\!| a_1, \ldots, a_n |\!) \in dom(P) \\ F_0 = (\bot, f(\!| a_1, \ldots, a_n |\!), \{a_1 \mapsto F(r_1), \ldots, a_n \mapsto F(r_n)\}) \\ \left(\; P(f(\!| a_1, \ldots, a_n |\!)) \; \right)_{H,F_0} \rightsquigarrow^* \left(\; ret \; \right)_{H',F_0'} \end{array}}{\left(\; pcall \; f(r_1, \ldots, r_n) \; \right)_{H,F} \rightsquigarrow \left(\; skip \; \right)_{H',F}} \; pcall$$

Sémantique opérationnelle : exemple

$$\frac{f(\!|\, a_1, \ldots, a_n \,|\!) \in dom(P) \qquad F_0 = (\bot, f(\!|\, a_1, \ldots, a_n \,|\!), \{a_1 \mapsto F(r_1), \ldots, a_n \mapsto F(r_n)\}) \qquad \left(\ P(f(\!|\, a_1, \ldots, a_n \,|\!))\ \right)_{H,F_0} \rightsquigarrow^* \left(\ ret\ \right)_{H',F_0'}}{\left(\ pcall\ f(r_1, \ldots, r_n)\ \right)_{H,F} \rightsquigarrow \left(\ skip\ \right)_{H',F}}\ pcall$$

Sémantique opérationnelle : exemple

$$\frac{\begin{array}{c} f(\!| a_1, \ldots, a_n |\!) \in dom(P) \\ F_0 = (\bot, f(\!| a_1, \ldots, a_n |\!), \{a_1 \mapsto F(r_1), \ldots, a_n \mapsto F(r_n)\}) \\ \left(\; P(f(\!| a_1, \ldots, a_n |\!)) \; \right)_{H,F_0} \rightsquigarrow^* \left(\; ret \; \right)_{H',F_0'} \end{array}}{\left(\; pcall \; f(r_1, \ldots, r_n) \; \right)_{H,F} \rightsquigarrow \left(\; skip \; \right)_{H',F}} \; pcall$$

Sémantique opérationnelle : exemple

$$\frac{f(\!|\, a_1, \ldots, a_n \,|\!) \in dom(P) \qquad F_0 = (\bot, f(\!|\, a_1, \ldots, a_n \,|\!), \{a_1 \mapsto F(r_1), \ldots, a_n \mapsto F(r_n)\}) \qquad \left(\ P(f(\!|\, a_1, \ldots, a_n \,|\!))\ \right)_{H,F_0} \rightsquigarrow^* \left(\ ret\ \right)_{H',F_0'}}{\left(\ pcall\ f(r_1, \ldots, r_n)\ \right)_{H,F} \rightsquigarrow \left(\ skip\ \right)_{H',F}}\ pcall$$

Sémantique opérationnelle : exemple

$$\frac{f(\!|\, a_1, \ldots, a_n \,|\!) \in dom(P) \qquad F_0 = (\bot, f(\!|\, a_1, \ldots, a_n \,|\!), \{a_1 \mapsto F(r_1), \ldots, a_n \mapsto F(r_n)\}) \qquad \left(\ P(f(\!|\, a_1, \ldots, a_n \,|\!))\ \right)_{H,F_0} \rightsquigarrow^* \left(\ ret\ \right)_{H',F_0'}}{\left(\ pcall\ f(r_1, \ldots, r_n)\ \right)_{H,F} \rightsquigarrow \left(\ skip\ \right)_{H',F}}\ pcall$$

Sémantique opérationnelle : exemple

$$\frac{f(\!|\, a_1, \ldots, a_n \,|\!) \in dom(P) \qquad F_0 = (\bot, f(\!|\, a_1, \ldots, a_n \,|\!), \{a_1 \mapsto F(r_1), \ldots, a_n \mapsto F(r_n)\}) \qquad \left(\; P(f(\!|\, a_1, \ldots, a_n \,|\!)) \;\right)_{H,F_0} \rightsquigarrow^* \left(\; ret \;\right)_{H',F_0'}}{\left(\; pcall\ f(r_1, \ldots, r_n) \;\right)_{H,F} \rightsquigarrow \left(\; skip \;\right)_{H',F}}\ pcall$$

Sémantique opérationnelle : exemple

$$\frac{\begin{array}{c} f(\!|\, a_1, \ldots, a_n \,|\!) \in dom(P) \\ F_0 = (\bot, f(\!|\, a_1, \ldots, a_n \,|\!), \{a_1 \mapsto F(r_1), \ldots, a_n \mapsto F(r_n)\}) \\ \left(\ P(f(\!|\, a_1, \ldots, a_n \,|\!))\ \right)_{H,F_0} \rightsquigarrow^* \left(\ ret\ \right)_{H',F_0'} \end{array}}{\left(\ pcall\ f(r_1, \ldots, r_n)\ \right)_{H,F} \rightsquigarrow \left(\ skip\ \right)_{H',F}}\ pcall$$

Sémantique opérationnelle : exemple

$$\frac{\begin{array}{c} f(\!|\, a_1, \ldots, a_n \,|\!) \in dom(P) \\ F_0 = (\bot, f(\!|\, a_1, \ldots, a_n \,|\!), \{a_1 \mapsto F(r_1), \ldots, a_n \mapsto F(r_n)\}) \\ \left(\ P(f(\!|\, a_1, \ldots, a_n \,|\!))\ \right)_{H,F_0} \rightsquigarrow^* \left(\ ret\ \right)_{H',F_0'} \end{array}}{\left(\ pcall\ f(r_1, \ldots, r_n)\ \right)_{H,F} \rightsquigarrow \left(\ skip\ \right)_{H',F}}\ pcall$$

Sémantique opérationnelle : exemple

$$\frac{\begin{array}{c} f(\!|\, a_1, \ldots, a_n \,|\!) \in dom(P) \\ F_0 = (\bot, f(\!|\, a_1, \ldots, a_n \,|\!), \{a_1 \mapsto F(r_1), \ldots, a_n \mapsto F(r_n)\}) \\ \left(\ P(f(\!|\, a_1, \ldots, a_n \,|\!))\ \right)_{H,F_0} \rightsquigarrow^* \left(\ ret\ \right)_{H',F_0'} \end{array}}{\left(\ pcall\ f(r_1, \ldots, r_n)\ \right)_{H,F} \rightsquigarrow \left(\ skip\ \right)_{H',F}}\ pcall$$

Résultats théoriques

Certification de la bibliothèque générée

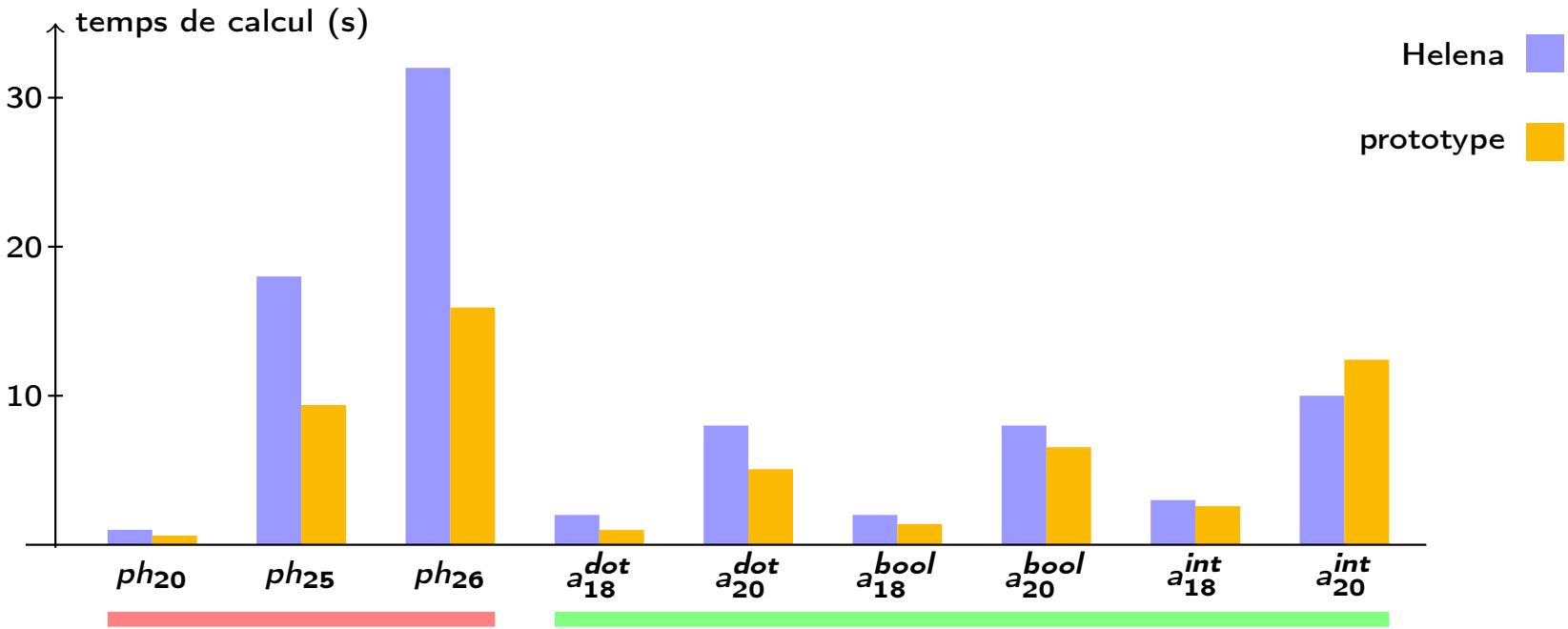
- ◇ algorithmes et structures — correction, terminaison
- ◇ différentes implantations — également prouvées
- ◇ optimisations — transformations de code
- ◇ résultats sur la sémantique — résultats intermédiaires

- ▶ de façon modulaire — axiomatisation des interfaces
- ▶ de façon détaillée — inductions, réductions, . . .

Résultats pratiques

Prototype de compilateur

- sur la classe de réseaux de Petri considérée
- implémente trois optimisations
- bonnes performances



Bilan

Problématique : cadre pour la compilation optimisée

- ◇ formalisé — certification du code généré
- ◇ implanté — 4000+ lignes de code, 4 langages
- ◇ appliqué — 2 études de cas, bonnes performances

Modularité, flexibilité, interopérabilité, accélération

- ◇ formalisation des interfaces
- ◇ preuves indépendantes
- ◇ optimisations variées — algos, structures
- ◇ compatibilité avec C — autres langages, outils
- ◇ comparaison avec Helena — bonnes performances

Perspectives

Dans l'immédiat

- ◇ publier le travail
- ◇ distribuer le prototype — intégration à Helena
- ◇ élargir la classe de réseaux de Petri
- ◇ nouvelles optimisations — partage de mémoire, ...
- ◇ études de cas — projets et stages

Sujet de thèse

Compilation optimisée pour le model-checking symbolique

- ◇ même démarche
- ◇ nouvelles techniques
- ◇ approche complémentaire — simulation, génération de code