Analyses statiques : certifier et quantifier

David Cachera

ENS Cachan, antenne de Bretagne
équipe-projet Celtique, IRISA/INRIA Rennes - Bretagne Atlantique
30 août 2010

habilitation à diriger des recherches

Contexte scientifique

Contexte scientifique

Nous accordons notre confiance au quotidien à des systèmes informatiques...

Contexte scientifique

Nous accordons notre confiance au quotidien à des systèmes informatiques...

Contexte scientifique

Nous accordons notre confiance au quotidien à des systèmes informatiques...

Contexte scientifique

Nous accordons notre confiance au quotidien à des systèmes informatiques...

Contexte scientifique

Nous accordons notre confiance au quotidien à des systèmes informatiques...

... de plus en plus invisibles et *embarqués*

Contexte scientifique

Nous accordons notre confiance au quotidien à des systèmes informatiques...

... de plus en plus invisibles et *embarqués*

Contexte scientifique

Nous accordons notre confiance au quotidien à des systèmes informatiques...

... de plus en plus invisibles et *embarqués*

Contexte scientifique

Nous accordons notre confiance au quotidien à des systèmes informatiques...

... de plus en plus invisibles et *embarqués*

Les erreurs coûtent cher

- en argent
- en vies humaines
- à l'environnement

Contexte scientifique

Nous accordons notre confiance au quotidien à des systèmes informatiques...

... de plus en plus invisibles et *embarqués*

Les erreurs coûtent cher

- en argent
- en vies humaines
- à l'environnement

Comment s'assurer de leur fiabilité ?

Vérification formelle

Montrer formellement qu'un programme (ou système) est correct vis-à-vis de certaines propriétés

Nécessite un modèle précis (une **sémantique**)

- vérification de modèle (*model checking*)
  explorer les états possibles du système
- démonstration « automatique » de théorèmes
  preuve construite par un ordinateur avec l'aide d'un humain : **assistant de preuve**
- **analyse statique**
  calcul automatique de certaines propriétés d'un programme, sans l'exécuter

Vérifier quoi ?

Pas tout : intrinsèquement impossible

- certaines propriétés
- pas toujours de réponse

Qualitatif ou quantitatif ?

- dans un circuit, un certain signal sera toujours vrai pendant une période déterminée
- un programme n'atteindra jamais un état « dangereux »
- la consommation de ressources induite par l'exécution d'un programme restera sous un certain seuil

Contenu de l'exposé

Vérification dans le modèle polyédrique

- mixte matériel-logiciel
- utilisation d'assistants de preuve et définition d'un système de preuve ad hoc

Analyse statique certifiée

- fiabilité des outils de vérification eux-mêmes
- certification d'analyses statiques à l'aide de l'assistant de preuve Coq

Analyse statique quantitative

- traiter d'autres types de propriétés du logiciel
- définition d'un cadre d'analyse statique

Contexte humain

Katell Morin-Allory

Patrice Quinton

Tanguy Risset

Sanjay Rajopadhye

David Pichardie

Thomas Jensen

Gerardo Schneider

Vlad Rusu

Pascal Sotin

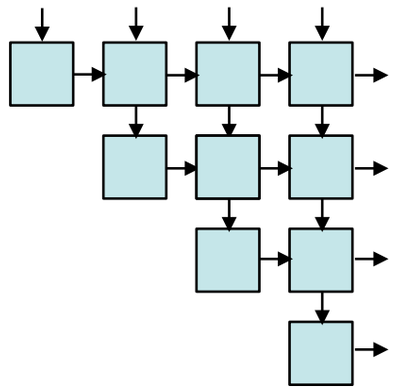
Arnaud Jobin

ens CACHAN BRETAGNE

Vérification dans le modèle polyédrique

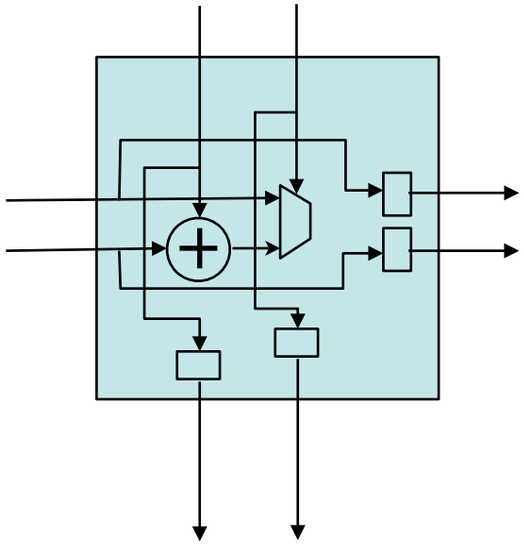
Le modèle polyédrique [KMW 67]

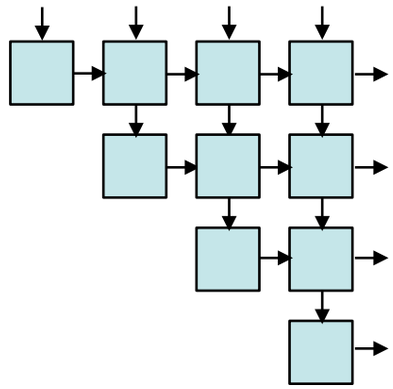
Idée : représenter de façon compacte un système (matériel ou logiciel) composé d'un grand nombre de blocs identiques qui interagissent



Le modèle polyédrique [KMW 67]

Idée : représenter de façon compacte un système (matériel ou logiciel) composé d'un grand nombre de blocs identiques qui interagissent





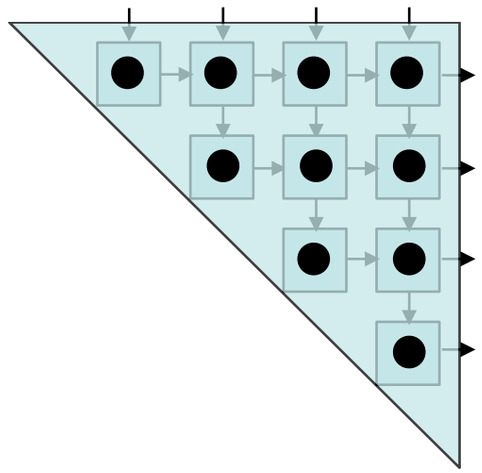
chaque bloc contient des signaux

et des

opérations élémentaires

Le modèle polyédrique [KMW 67]

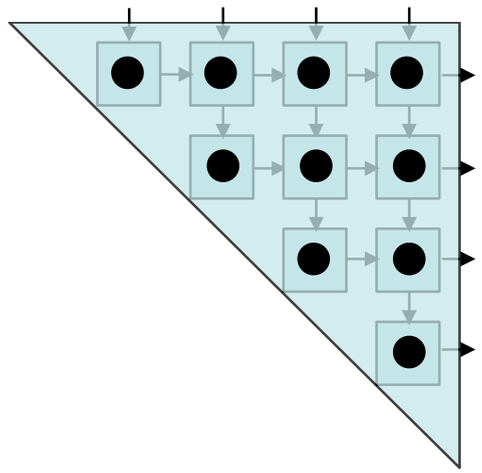
Idée : représenter de façon compacte un système (matériel ou logiciel) composé d'un grand nombre de blocs identiques qui interagissent



blocs et signaux ont des indices multidimensionnels qui sont les points entiers d'un polyèdre

Le modèle polyédrique [KMW 67]

Idée : représenter de façon compacte un système (matériel ou logiciel) composé d'un grand nombre de blocs identiques qui interagissent



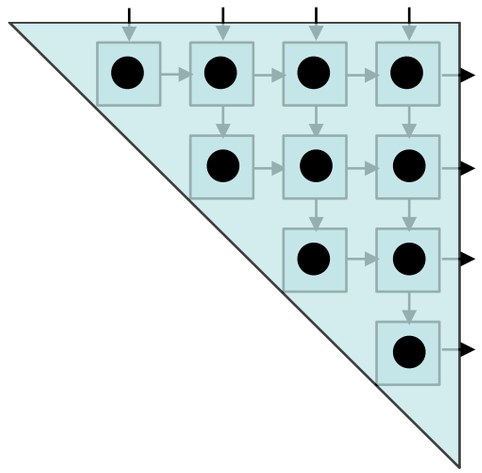
blocs et signaux ont des indices multidimensionnels qui sont les points entiers d'un polyèdre

Les signaux sont décrits par des équations, mutuellement récursives

$$Out_1(i,j,t) = In_1(i,j,t-1)$$
$$In_1(i,j,t) = Out_1(i-1,j,t)$$

Le modèle polyédrique [KMW 67]

Idée : représenter de façon compacte un système (matériel ou logiciel) composé d'un grand nombre de blocs identiques qui interagissent



blocs et signaux ont des indices multidimensionnels qui sont les points entiers d'un polyèdre

Les signaux sont décrits par des équations, mutuellement récursives

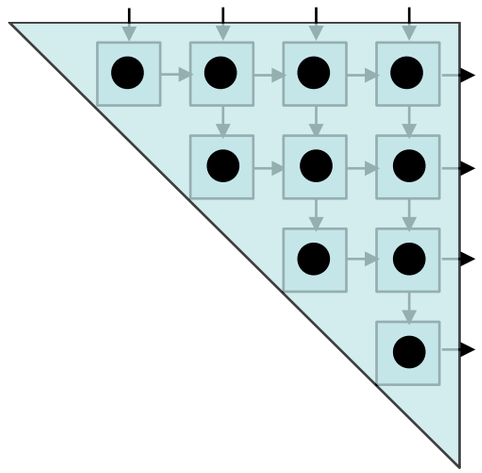
$$Out_1(i,j,t) = In_1(i,j,t-1)$$

décalage temporel

$$In_1(i,j,t) = Out_1(i-1,j,t)$$

Le modèle polyédrique [KMW 67]

Idée : représenter de façon compacte un système (matériel ou logiciel) composé d'un grand nombre de blocs identiques qui interagissent



blocs et signaux ont des indices multidimensionnels qui sont les points entiers d'un polyèdre

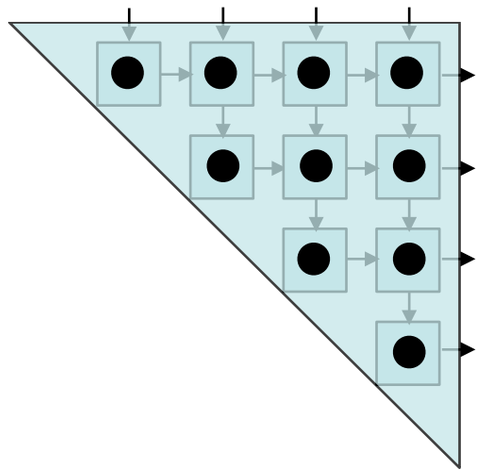
Les signaux sont décrits par des équations, mutuellement récursives

$$Out_1(i,j,t) = In_1(i,j,t-1)$$ décalage temporel

$$In_1(i,j,t) = Out_1(i-1,j,t)$$ communication

Le modèle polyédrique [KMW 67]

Idée : représenter de façon compacte un système (matériel ou logiciel) composé d'un grand nombre de blocs identiques qui interagissent



blocs et signaux ont des indices multidimensionnels qui sont les points entiers d'un polyèdre

Les signaux sont décrits par des équations, mutuellement récursives

$$Out_1(i,j,t) = In_1(i,j,t-1)$$ décalage temporel

$$In_1(i,j,t) = Out_1(i-1,j,t)$$ communication

Défi : vérifier des systèmes décrits avec des paramètres symboliques

Vérification dans le modèle polyédrique

Vérification dans le modèle polyédrique

Synthèse de haut niveau

- description abstraite (spécification) du système raffinée par transformations successives jusqu'à une description détaillée (implémentation)

Vérification dans le modèle polyédrique

Synthèse de haut niveau

description abstraite (spécification) du système raffinée par transformations successives jusqu'à une description détaillée (implémentation)

Pourquoi vérifier ?

les étapes de transformation préservent la sémantique : si la description initiale est correcte, l'implémentation doit l'être aussi

- correction de la description initiale ?
- optimisations introduites « manuellement »

Vérification dans le modèle polyédrique

Synthèse de haut niveau

description abstraite (spécification) du système raffinée par transformations successives jusqu'à une description détaillée (implémentation)

Pourquoi vérifier ?

les étapes de transformation préservent la sémantique : si la description initiale est correcte, l'implémentation doit l'être aussi

- correction de la description initiale ?
- optimisations introduites « manuellement »

Que vérifier ?

- des propriétés fonctionnelles (un multiplieur calcule une multiplication)
- des propriétés de contrôle : un signal de contrôle a bien la valeur voulue

Deux approches

Deux approches

1. Utilisation d'assistants de preuve
   - traduction directe des équations décrivant le système dans le langage de l'assistant
   - les propriétés sont prouvées grâce à la logique de celui-ci
   - expériences avec PVS et Coq
     création automatique de schémas d'induction [DEA David Pichardie]

Deux approches

1. Utilisation d'assistants de preuve
   - traduction directe des équations décrivant le système dans le langage de l'assistant
   - les propriétés sont prouvées grâce à la logique de celui-ci
   - expériences avec PVS et Coq

     création automatique de schémas d'induction

     [DEA David Pichardie]

2. Création d'une technique de preuve adaptée
   - tirer parti de la régularité du modèle polyédrique
   - montrer des propriétés du type « tel signal est vrai sur tous les points d'un certain polyèdre »

     [thèse Katell Morin-Allory]

Construction d'une preuve

Système de preuve

- règles « classiques »
- règles utilisant des calculs polyédriques

Difficulté : traitement des équations récursives

→ solution originale, combinaison de

- substitutions syntaxiques
- recherche de motifs
- heuristiques d'agrandissement des domaines

Résultat : méthode et outil de construction semi-automatique d'un arbre de preuve

intervention de l'utilisateur pour l'application des heuristiques d'élargissement

Bilan

Publications

- **Embedding of Systems of Affine Recurrence Equations in Coq**. *TPHOLs 2003*. Avec David Pichardie.
- **Proving Parameterized Systems: The Use of Pseudo-Pipelines in Polyhedral Logic**. *CHARME* 2005. Avec Katell Morin-Allory.
- **Verification of safety properties for parameterized regular systems**. *Trans. on Embedded Computing Systems*, 2005. Avec Katell Morin-Allory

Développement de prototypes (Coq et Mathematica)

- démontrer la faisabilité des approches
- intégrer la vérification formelle dans l'environnement de synthèse développé par l'équipe COSI

Analyse statique : les ingrédients de base

Analyse statique

Rappel : on cherche à calculer certaines propriétés d'un programme, sans exécuter celui-ci

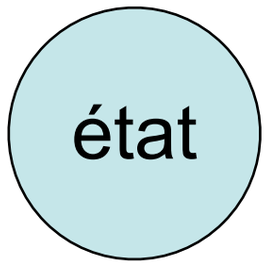
Comme on ne peut pas « calculer » le comportement (la sémantique) d'un programme pour n'importe laquelle de ses entrées, on en calcule une approximation

Cadre théorique général : l'interprétation abstraite [CC77]

- construction de sémantiques approchées correctes par construction
- notion de meilleure approximation

Sémantique concrète

Système de transitions



Notations

- $State$: ensemble des états
- $\mathscr{S}_{init}$: états initiaux
- $Safe$: états acceptables

États accessibles pour un programme P

$$[\![P]\!] = \{\sigma \in State \mid \exists \sigma_0 \in \mathscr{S}_{init}, \sigma_0 \rightarrow^* \sigma\}$$

Treillis complet des états

$$(\mathscr{P}(State), \subseteq, \cup, \cap)$$

Sémantique concrète

Système de transitions

Notations

- $State$: ensemble des états
- $\mathscr{S}_{init}$: états initiaux
- $Safe$: états acceptables

États accessibles pour un programme P

$$[\![P]\!] = \{\sigma \in State \mid \exists \sigma_0 \in \mathscr{S}_{init}, \sigma_0 \rightarrow^* \sigma\}$$

Treillis complet des états

$$(\mathscr{P}(State), \subseteq, \cup, \cap)$$

Domaines abstraits

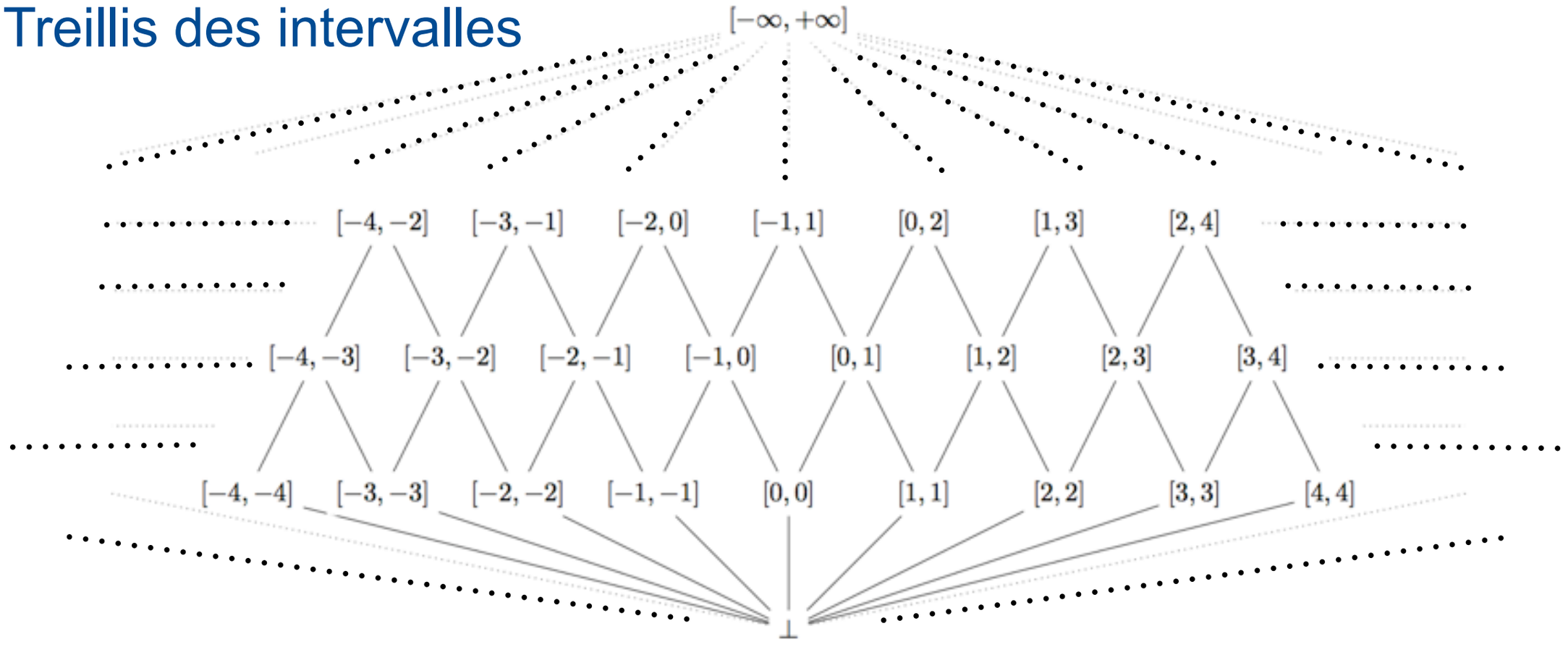
Exemple : variables entières

$$State = Var \to \mathbb{Z}$$

Un ensemble d'entiers est abstrait par un intervalle les contenant tous

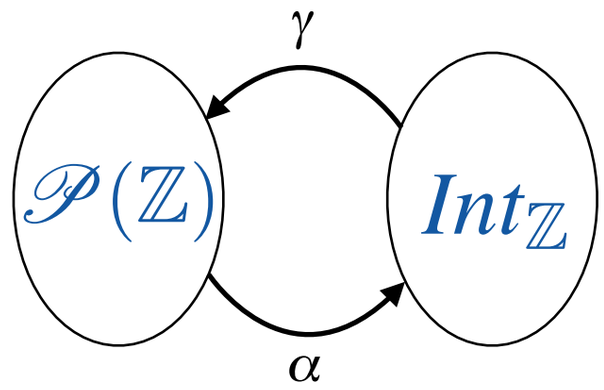
$$State^\sharp = Var \to Int_{\mathbb{Z}}$$

Treillis des intervalles



Lien entre concret et abstrait

Domaines abstraits et concrets sont liés par une connexion de Galois



Les fonctions α et γ

- sont monotones
- forment une paire résiduée

$$\alpha \circ \gamma \leq Id^{\sharp} \qquad Id \subseteq \gamma \circ \alpha$$

Ex : intervalles

$$\alpha(\{1,3\}) = [1,3] \qquad \gamma([1,3]) = \{1,2,3\} \qquad \alpha(\{1,2,3\}) = [1,3]$$

Sémantique abstraite

Fonction de transfert abstraite :

$F : \mathscr{P}(State) \to \mathscr{P}(State)$, fonction de transfert concrète,

Déf : $F^\sharp : State^\sharp \to State^\sharp$ est une approximation correcte de F si

$$\alpha \circ F \leq^\sharp F^\sharp \circ \alpha \qquad (F \circ \gamma \subseteq \gamma \circ F^\sharp)$$

Généralement, le résultat de l'analyse est solution d'une équation de point fixe

$$State^\sharp \ni \sigma^\sharp = F^\sharp(\sigma^\sharp)$$

Ex : $State^\sharp = PC \to (Var \to Int_{\mathbb{Z}})$

Notation : $P \vdash \sigma^\sharp$

Résultat final

Par passage au point fixe, la condition

$$F \circ \gamma \leq \gamma \circ F^\sharp$$

devient

$$[\![P]\!] \subseteq \gamma(\sigma^\sharp)$$

si on vérifie

$$\gamma(\sigma^\sharp) \subseteq Safe$$

alors on aura

$$[\![P]\!] \subseteq Safe$$

Analyses statiques certifiées

Outil de vérification ? Oui mais...

Il faut aussi avoir confiance dans les outils de vérification

- complexes
- base formelle : spécification et preuve de correction
- mais l'outil lui-même est un programme !
- il faut donc s'assurer de la correction de son implémentation

Outil de vérification ? Oui mais...

$$
\begin{array}{ll}
 & \dot{\ddot{\alpha}}[\![P]\!](\mathrm{Post}[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!]) \\
= & \quad \wr \text{def. (110) of } \dot{\ddot{\alpha}}[\![P]\!] \wr \\
 & \ddot{\alpha}[\![P]\!] \circ \mathrm{Post}[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{def. (103) of Post} \wr \\
 & \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau^\star[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!]] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{big step operational semantics (93)} \wr \\
 & \ddot{\alpha}[\![P]\!] \circ \mathrm{post}\,[(1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t) \cup (1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \\
 & \tau^f)] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{Galois connection (98) so that post preserves joins} \wr \\
 & \ddot{\alpha}[\![P]\!] \circ (\mathrm{post}\,[(1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t)] \mathbin{\dot{\cup}} \\
 & \mathrm{post}[(1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f)]) \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{Galois connection (106) so that } \ddot{\alpha}[\![P]\!] \text{ preserves joins} \wr \\
 & (\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[(1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t)] \circ \ddot{\gamma}[\![P]\!]) \mathbin{\ddot{\cup}} (\ddot{\alpha}[\![P]\!] \circ \\
 & \mathrm{post}[(1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f)] \circ \ddot{\gamma}[\![P]\!]) \\
\dddot{\sqsubseteq} & \quad \wr \text{lemma (5.3) and similar one for the } \texttt{else} \text{ branch} \wr \\
 & \lambda J \cdot \text{let } J^{t'} = \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \mathrm{at}_P[\![S_t]\!] \ ? \ J_{\mathrm{at}_P[\![S_t]\!]} \mathbin{\dot{\cup}} \mathrm{Abexp}[\![B]\!](J_\ell) \ ¿ \ J_l) \text{ in} \qquad (120) \\
 & \quad \text{let } J^{t''} = \mathrm{APost}[\![S_t]\!](J^{t'}) \text{ in} \\
 & \quad\quad \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \ell' \ ? \ J^{t''}_{\ell'} \mathbin{\dot{\cup}} J^{t''}_{\mathrm{after}_P[\![S_t]\!]} \ ¿ \ J^{t''}_l) \\
 & \quad \ddot{\cup} \\
 & \quad \text{let } J^{f'} = \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \mathrm{at}_P[\![S_f]\!] \ ? \ J_{\mathrm{at}_P[\![S_f]\!]} \mathbin{\dot{\cup}} \mathrm{Abexp}[\![T(\neg B)]\!](J_\ell) \ ¿ \ J_l) \text{ in} \\
 & \quad\quad \text{let } J^{f''} = \mathrm{APost}[\![S_f]\!](J^{f'}) \text{ in} \\
 & \quad\quad\quad \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \ell' \ ? \ J^{f''}_{\ell'} \mathbin{\dot{\cup}} J^{f''}_{\mathrm{after}_P[\![S_f]\!]} \ ¿ \ J^{f''}_l) \\
= & \quad \wr \text{by grouping similar terms} \wr \\
 & \lambda J \cdot \text{let } J^{t'} = \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \mathrm{at}_P[\![S_t]\!] \ ? \ J_{\mathrm{at}_P[\![S_t]\!]} \mathbin{\dot{\cup}} \mathrm{Abexp}[\![B]\!](J_\ell) \ ¿ \ J_l) \\
 & \quad \text{and } J^{f'} = \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \mathrm{at}_P[\![S_f]\!] \ ? \ J_{\mathrm{at}_P[\![S_f]\!]} \mathbin{\dot{\cup}} \mathrm{Abexp}[\![T(\neg B)]\!](J_\ell) \ ¿ \ J_l) \text{ in} \\
 & \quad\quad \text{let } J^{t''} = \mathrm{APost}[\![S_t]\!](J^{t'}) \\
 & \quad\quad \text{and } J^{f''} = \mathrm{APost}[\![S_f]\!](J^{f'}) \text{ in} \\
 & \quad\quad\quad \lambda l \in \mathrm{in}_P[\![P]\!] \cdot (l = \ell' \ ? \ J^{t''}_{\ell'} \mathbin{\dot{\cup}} J^{t''}_{\mathrm{after}_P[\![S_t]\!]} \mathbin{\dot{\cup}} J^{f''}_{\ell'} \mathbin{\dot{\cup}} J^{f''}_{\mathrm{after}_P[\![S_f]\!]} \ ¿ \ J^{t''}_l \mathbin{\dot{\cup}} J^{f''}_l) \\
= & \quad \wr \text{by locality (113) and labelling scheme (59) so that in particular } J^{t''}_{\ell'} = J^{t'}_{\ell'} = J^t_{\ell'} = J^f_{\ell'} \\
 & \quad = J^{f'}_{\ell'} = J^{f''}_{\ell'} \text{ and APost}[\![S_t]\!] \text{ and APost}[\![S_f]\!] \text{ do not interfere} \wr
\end{array}
$$

©P.Cousot

...ns les outils de vérification

...n et preuve de correction

...n programme !

... correction de son

Outil de vérification ? Oui mais...

$$
\begin{array}{ll}
 & \dot{\alpha}[\![P]\!](\mathrm{Post}[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!]) \\
= & \quad \wr \text{def. (110) of } \dot{\alpha}[\![P]\!] \wr \\
 & \ddot{\alpha}[\![P]\!] \circ \mathrm{Post}[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{def. (103) of Post} \wr \\
 & \ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau^\star[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!]] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{big step operational semantics (93)} \wr \\
 & \ddot{\alpha}[\![P]\!] \circ \mathrm{post}\,[(1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t) \cup (1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \\
 & \tau^f)] \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{Galois connection (98) so that post preserves joins} \wr \\
 & \ddot{\alpha}[\![P]\!] \circ (\mathrm{post}\,[(1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t)] \mathbin{\dot{\cup}} \\
 & \mathrm{post}[(1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f)]) \circ \ddot{\gamma}[\![P]\!] \\
= & \quad \wr \text{Galois connection (106) so that } \ddot{\alpha}[\![P]\!] \text{ preserves joins} \wr \\
 & (\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[(1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t)] \circ \ddot{\gamma}[\![P]\!]) \mathbin{\ddot{\cup}} (\ddot{\alpha}[\![P]\!] \circ \\
 & \mathrm{post}[(1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f)] \circ \ddot{\gamma}[\![P]\!]) \\
\dot{\sqsubseteq} & \quad \wr \text{lemma (5.3) and similar one for the } \texttt{else} \text{ branch} \wr \\
 & \lambda J \bullet \text{let } J^{t'} = \lambda l \in \mathrm{in}_P[\![P]\!] \bullet (l = \mathrm{at}_P[\![S_t]\!] \ ? \ J_{\mathrm{at}_P[\![S_t]\!]} \mathbin{\dot{\cup}} \mathrm{Abexp}[\![B]\!](J_\ell) \ ¿ \ J_l) \text{ in} \qquad (120) \\
 & \quad \text{let } J^{t''} = \mathrm{APost}[\![S_t]\!](J^{t'}) \text{ in} \\
 & \quad \lambda l \in \mathrm{in}_P[\![P]\!] \bullet (l = \ell' \ ? \ J^{t''}_{\ell'} \mathbin{\dot{\cup}} J^{t''}_{\mathrm{after}_P[\![S_t]\!]} \ ¿ \ J^{t''}_l) \\
 & \ddot{\cup} \\
 & \text{let } J^{f'} = \lambda l \in \mathrm{in}_P[\![P]\!] \bullet (l = \mathrm{at}_P[\![S_f]\!] \ ? \ J_{\mathrm{at}_P[\![S_f]\!]} \mathbin{\dot{\cup}} \mathrm{Abexp}[\![T(\neg B)]\!](J_\ell) \ ¿ \ J_l) \text{ in} \\
 & \quad \text{let } J^{f''} = \mathrm{APost}[\![S_f]\!](J^{f'}) \text{ in} \\
 & \quad \lambda l \in \mathrm{in}_P[\![P]\!] \bullet (l = \ell' \ ? \ J^{f''}_{\ell'} \mathbin{\dot{\cup}} J^{f''}_{\mathrm{after}_P[\![S_f]\!]} \ ¿ \ J^{f''}_l) \\
= & \quad \wr \text{by grouping similar terms} \wr \\
 & \lambda J \bullet \text{let } J^{t'} = \lambda l \in \mathrm{in}_P[\![P]\!] \bullet (l = \mathrm{at}_P[\![S_t]\!] \ ? \ J_{\mathrm{at}_P[\![S_t]\!]} \mathbin{\dot{\cup}} \mathrm{Abexp}[\![B]\!](J_\ell) \ ¿ \ J_l) \\
 & \quad \text{and } J^{f'} = \lambda l \in \mathrm{in}_P[\![P]\!] \bullet (l = \mathrm{at}_P[\![S_f]\!] \ ? \ J_{\mathrm{at}_P[\![S_f]\!]} \mathbin{\dot{\cup}} \mathrm{Abexp}[\![T(\neg B)]\!](J_\ell) \ ¿ \ J_l) \text{ in} \\
 & \quad \text{let } J^{t''} = \mathrm{APost}[\![S_t]\!](J^{t'}) \\
 & \quad \text{and } J^{f''} = \mathrm{APost}[\![S_f]\!](J^{f'}) \text{ in} \\
 & \quad \lambda l \in \mathrm{in}_P[\![P]\!] \bullet (l = \ell' \ ? \ J^{t''}_{\ell'} \mathbin{\dot{\cup}} J^{t''}_{\mathrm{after}_P[\![S_t]\!]} \mathbin{\dot{\cup}} J^{f''}_{\ell'} \mathbin{\dot{\cup}} J^{f''}_{\mathrm{after}_P[\![S_f]\!]} \ ¿ \ J^{t''}_l \mathbin{\dot{\cup}} J^{f''}_l) \\
= & \quad \wr \text{by locality (113) and labelling scheme (59) so that in particular } J^{t''}_{\ell'} = J^{t'}_{\ell'} = J^t_{\ell'} = J^f_{\ell'} \\
 & = J^{f'}_{\ell'} = J^{f''}_{\ell'} \text{ and } \mathrm{APost}[\![S_t]\!] \text{ and } \mathrm{APost}[\![S_f]\!] \text{ do not interfere} \wr
\end{array}
$$

©P.Cousot

```
matrix_t* _matrix_alloc_int(const int mr, const int nc)
{
  matrix_t* mat = (matrix_t*)malloc(sizeof(matrix_t));
  mat->nbrows = mat->_maxrows = mr;
  mat->nbcolumns = nc;
  mat->_sorted = s;
  if (mr*nc>0){
    int i;
    pkint_t* q;
    mat->_pinit = _vector_alloc_int(mr*nc);
    mat->p = (pkint_t**)malloc(mr * sizeof(pkint_t*));
    q = mat->_pinit;
    for (i=0;i<mr;i++){
      mat->p[i]=q;
      q=q+nc;
    }}
  return mat;
}


void backsubstitute(matrix_t* con, int rank)
{
  int i,j,k;
  for (k=rank-1; k>=0; k--) {
    j = pk_cherni_intp[k];
    for (i=0; i<k; i++) {
      if (pkint_sgn(con->p[i][j]))
        matrix_combine_rows(con,i,k,i,j);
    }
    for (i=k+1; i<con->nbrows; i++) {
      if (pkint_sgn(con->p[i][j]))
        matrix_combine_rows(con,i,k,i,j);
    }}
}
```

©B.Jeannet

Outil de vérification ? Oui mais...

$\dot{\alpha}[\![P]\!](\mathrm{Post}[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!])$

$=$ ⟅def. (110) of $\dot{\alpha}[\![P]\!]$⟆

$\ddot{\alpha}[\![P]\!] \circ \mathrm{Post}[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!] \circ \ddot{\gamma}[\![P]\!]$

$=$ ⟅def. (103) of Post⟆

$\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[\tau^\star[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!]] \circ \ddot{\gamma}[\![P]\!]$

$=$ ⟅big step operational semantics (93)⟆

$\ddot{\alpha}[\![P]\!] \circ \mathrm{post}\,[(1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t) \cup (1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f)] \circ \ddot{\gamma}[\![P]\!]$

$=$ ⟅Galois connection (98) so that post preserves joins⟆

$\ddot{\alpha}[\![P]\!] \circ (\mathrm{post}\,[(1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t)] \dot{\cup} \mathrm{post}[(1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f)]) \circ \ddot{\gamma}[\![P]\!]$

$=$ ⟅Galois connection (106) so that $\ddot{\alpha}[\![P]\!]$ preserves joins⟆

$(\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[(1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t)] \circ \ddot{\gamma}[\![P]\!]) \ddot{\cup} (\ddot{\alpha}[\![P]\!] \circ \mathrm{post}[(1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f)] \circ \ddot{\gamma}[\![P]\!])$

$\ddot{\sqsubseteq}$ ⟅lemma (5.3) and similar one for the **else** branch⟆

$\lambda J^\bullet$ let $J^{t'} = \lambda l \in \mathrm{in}_P[\![P]\!]^\bullet(l = \mathrm{at}_P[\![S_t]\!]\ ?\ J_{\mathrm{at}_P[\![S_t]\!]} \dot{\cup} \mathrm{Abexp}[\![B]\!](J_\ell)\ ¿\ J_l)$ in (120)

let $J^{t''} = \mathrm{APost}[\![S_t]\!](J^{t'})$ in

$\lambda l \in \mathrm{in}_P[\![P]\!]^\bullet(l = \ell'\ ?\ J^{t''}_{\ell'} \dot{\cup} J^{t''}_{\mathrm{after}_P[\![S_t]\!]}\ ¿\ J^{t''}_l)$

$\ddot{\cup}$

let $J^{f'} = \lambda l \in \mathrm{in}_P[\![P]\!]^\bullet(l = \mathrm{at}_P[\![S_f]\!]\ ?\ J_{\mathrm{at}_P[\![S_f]\!]} \dot{\cup} \mathrm{Abexp}[\![T(\neg B)]\!](J_\ell)\ ¿\ J_l)$ in

let $J^{f''} = \mathrm{APost}[\![S_f]\!](J^{f'})$ in

$\lambda l \in \mathrm{in}_P[\![P]\!]^\bullet(l = \ell'\ ?\ J^{f''}_{\ell'} \dot{\cup} J^{f''}_{\mathrm{after}_P[\![S_f]\!]}\ ¿\ J^{f''}_l)$

$=$ ⟅by grouping similar terms⟆

$\lambda J^\bullet$ let $J^{t'} = \lambda l \in \mathrm{in}_P[\![P]\!]^\bullet(l = \mathrm{at}_P[\![S_t]\!]\ ?\ J_{\mathrm{at}_P[\![S_t]\!]} \dot{\cup} \mathrm{Abexp}[\![B]\!](J_\ell)\ ¿\ J_l)$

and $J^{f'} = \lambda l \in \mathrm{in}_P[\![P]\!]^\bullet(l = \mathrm{at}_P[\![S_f]\!]\ ?\ J_{\mathrm{at}_P[\![S_f]\!]} \dot{\cup} \mathrm{Abexp}[\![T(\neg B)]\!](J_\ell)\ ¿\ J_l)$ in

let $J^{t''} = \mathrm{APost}[\![S_t]\!](J^{t'})$

and $J^{f''} = \mathrm{APost}[\![S_f]\!](J^{f'})$ in

$\lambda l \in \mathrm{in}_P[\![P]\!]^\bullet(l = \ell'\ ?\ J^{t''}_{\ell'} \dot{\cup} J^{t''}_{\mathrm{after}_P[\![S_t]\!]} \dot{\cup} J^{f''}_{\ell'} \dot{\cup} J^{f''}_{\mathrm{after}_P[\![S_f]\!]}\ ¿\ J^{t''}_l \dot{\cup} J^{f''}_l)$

$=$ ⟅by locality (113) and labelling scheme (59) so that in particular $J^{t''}_{\ell'} = J^{t'}_{\ell'} = J^t_{\ell'} = J^f_{\ell'}$ $= J^{f'}_{\ell'} = J^{f''}_{\ell'}$ and $\mathrm{APost}[\![S_t]\!]$ and $\mathrm{APost}[\![S_f]\!]$ do not interfere⟆

©P.Cousot

?

```
matrix_t* _matrix_alloc_int(const int mr, const int nc)
{
  matrix_t* mat = (matrix_t*)malloc(sizeof(matrix_t));
  mat->nbrows = mat->_maxrows = mr;
  mat->nbcolumns = nc;
  mat->_sorted = s;
  if (mr*nc>0){
    int i;
    pkint_t* q;
    mat->_pinit = _vector_alloc_int(mr*nc);
    mat->p = (pkint_t**)malloc(mr * sizeof(pkint_t*));
    q = mat->_pinit;
    for (i=0;i<mr;i++){
      mat->p[i]=q;
      q=q+nc;
    }}
  return mat;
}
```

```
void backsubstitute(matrix_t* con, int rank)
{
  int i,j,k;
  for (k=rank-1; k>=0; k--) {
    j = pk_cherni_intp[k];
    for (i=0; i<k; i++) {
      if (pkint_sgn(con->p[i][j]))
        matrix_combine_rows(con,i,k,i,j);
    }
    for (i=k+1; i<con->nbrows; i++) {
      if (pkint_sgn(con->p[i][j]))
        matrix_combine_rows(con,i,k,i,j);
    }}
}
```

©B.Jeannet

Défi

Construire des analyseurs statiques

- dont l'implémentation est certifiée correcte
- avec l'assistant de preuve Coq (logique constructive)
- en utilisant le mécanisme d'extraction de Coq pour obtenir l'implémentation OCaml de l'analyseur à partir de sa preuve
- sans introduire d'axiomes dans la logique de Coq

Feuille de route

- Définir en Coq les ingrédients d'une analyse statique
- Analyse spécifiée par un ensemble d'inéquations (contraintes)
- Montrer (en Coq) l'implication

$$\forall P, \sigma^\sharp.\ P \vdash \sigma^\sharp \Rightarrow [\![P]\!] \subseteq \gamma(\sigma^\sharp)$$

- Utiliser un solveur de point fixe pour calculer $[\![P]\!]^\sharp$ tel que

$$P \vdash [\![P]\!]^\sharp$$

- On a alors une approximation fiable et correcte de la sémantique de P

$$[\![P]\!] \subseteq \gamma([\![P]\!]^\sharp)$$

Construction d'une analyse

Un exemple : mini langage de manipulation de pile

```
Definition pc := word.
Definition var := word.
Inductive instruction := Push (n:integer) | Pop | Load (x:var) | Store (x:var)
                       | Binop (op:binop) | If (c:cmp) (i:pc) | Goto (i:pc)
Definition program := list (pc * instruction).
```

Domaines sémantiques

```
Definition locvar := var → integer.
Definition opstack := list integer.
Inductive state :=  St (i:pc) (s:opstack) (l:locvar) | Error.
```

Sémantique concrète

```
Inductive step (p:program) : state → state → Prop :=
| step_push : ∀ i s l n, instr_at p i = Some (Push n) →
              step (St i s l) (St (next i) (n :: s) l)
| step_load : ∀ i s l x v, instr_at p i = Some (Load x) →
              l x = v → step (St i s l) (St (next i) (v::s) l)

  ...
```

puis définition inductive des états atteignables

Construction d'une analyse

Un exemple : mini langage de manipulation de pile

```
Definition pc := word.
Definition var := word.
Inductive instruction := Push (n:integer) | Pop | Load (x:var) | Store (x:var)
                       | Binop (op:binop) | If (c:cmp) (i:pc) | Goto (i:pc)
Definition program := list (pc * instruction).
```

type somme

Domaines sémantiques

```
Definition locvar := var → integer.
Definition opstack := list integer.
Inductive state :=  St (i:pc) (s:opstack) (l:locvar) | Error.
```

Sémantique concrète

```
Inductive step (p:program) : state → state → Prop :=
| step_push : ∀ i s l n, instr_at p i = Some (Push n) →
              step (St i s l) (St (next i) (n :: s) l)
| step_load : ∀ i s l x v, instr_at p i = Some (Load x) →
              l x = v → step (St i s l) (St (next i) (v::s) l)
```

...

puis définition inductive des états atteignables

Construction d'une analyse

Un exemple : mini langage de manipulation de pile

```
Definition pc := word.
Definition var := word.
Inductive instruction := Push (n:integer) | Pop | Load (x:var) | Store (x:var)
                       | Binop (op:binop) | If (c:cmp) (i:pc) | Goto (i:pc)
Definition program := list (pc * instruction).
```

type somme

Domaines sémantiques

```
Definition locvar := var → integer.
Definition opstack := list integer.
Inductive state :=  St (i:pc) (s:opstack) (l:locvar) | Error.
```

$$L = \mathbb{V} \rightarrow \mathbb{Z}$$
$$S = (\mathbb{Z})^*$$
$$State = PC \times S \times L$$

Sémantique concrète

```
Inductive step (p:program) : state → state → Prop :=
| step_push : ∀ i s l n, instr_at p i = Some (Push n) →
              step (St i s l) (St (next i) (n :: s) l)
| step_load : ∀ i s l x v, instr_at p i = Some (Load x) →
              l x = v → step (St i s l) (St (next i) (v::s) l)
```

...

puis définition inductive des états atteignables

Construction d'une analyse

Un exemple : mini langage de manipulation de pile

```
Definition pc := word.
Definition var := word.
Inductive instruction := Push (n:integer) | Pop | Load (x:var) | Store (x:var)
                       | Binop (op:binop) | If (c:cmp) (i:pc) | Goto (i:pc)
Definition program := list (pc * instruction).
```

Domaines sémantiques

```
Definition locvar := var → integer.
Definition opstack := list integer.
Inductive state :=  St (i:pc) (s:opstack) (l:locvar) | Error.
```

$$L = \mathbb{V} \rightarrow \mathbb{Z}$$
$$S = (\mathbb{Z})^*$$
$$State = PC \times S \times L$$

définition inductive de fonction

Sémantique concrète

```
Inductive step (p:program) : state → state → Prop :=
| step_push : ∀ i s l n, instr_at p i = Some (Push n) →
              step (St i s l) (St (next i) (n :: s) l)
| step_load : ∀ i s l x v, instr_at p i = Some (Load x) →
              l x = v → step (St i s l) (St (next i) (v::s) l)
```

$$\frac{instrAt_P\ i = \mathrm{push\ n}}{< i, s, l > \rightarrow < i+1, n :: s, l >}$$

...

puis définition inductive des états atteignables

Domaines abstraits

Les états abstraits sont de la forme

$$State^\sharp = pc \rightarrow mem^\sharp$$

associant une propriété (sur les variables et le contenu de la pile) à chaque point de programme

Les valeurs des variables sont abstraites par des intervalles

$$l^\sharp \in Var \rightarrow Int_{\mathbb{Z}}$$

La pile d'opérandes est abstraite par une pile (liste) $s^\sharp$

Un état mémoire abstrait est un produit $(l^\sharp, s^\sharp)$

Domaines abstraits

Les états abstraits sont de la forme

$$State^\sharp = pc \to mem^\sharp$$

associant une propriété (sur les variables et le contenu de la pile) à chaque point de programme

Les valeurs des variables sont abstraites par des intervalles

$$l^\sharp \in Var \to Int_{\mathbb{Z}}$$

La pile d'opérandes est abstraite par une pile (liste) $s^\sharp$

Un état mémoire abstrait est un produit $(l^\sharp, s^\sharp)$

construire en Coq des treillis, par composition de treillis de base

Fonctions de concrétisation

Il est relativement aisé de définir une fonction de concrétisation

```
Definition gamma_locvar (L:LocvarLat.t) (l:locvar) : Prop :=
           ∀ x, gamma_val (LocvarLat.get L x) (l x).
```

La fonction `gamma_val` de concrétisation des valeurs numériques dépend elle-même de la concrétisation des intervalles.

Concrétisations paramétrées : si les variables locales peuvent contenir des références à des objets, la fonction de concrétisation sera paramétrée par le tas.

Fonctions de concrétisation

Il est relativement aisé de définir une fonction de concrétisation

$\gamma(\mathtt{L}) \ni \mathtt{l}$

```
Definition gamma_locvar (L:LocvarLat.t) (l:locvar) : Prop :=
          ∀ x, gamma_val (LocvarLat.get L x) (l x).
```

$\forall \mathtt{x},\ \gamma(\mathtt{L}(\mathtt{x})) \ni \mathtt{l}(\mathtt{x})$

La fonction `gamma_val` de concrétisation des valeurs numériques dépend elle-même de la concrétisation des intervalles.

Concrétisations paramétrées : si les variables locales peuvent contenir des références à des objets, la fonction de concrétisation sera paramétrée par le tas.

Fonctions de concrétisation

Il est relativement aisé de définir une fonction de concrétisation

$\gamma(\mathtt{L}) \ni \mathtt{l}$

```
Definition gamma_locvar (L:LocvarLat.t) (l:locvar) : Prop :=
          ∀ x, gamma_val (LocvarLat.get L x) (l x).
```

$\forall \mathtt{x},\ \gamma(\mathtt{L}(\mathtt{x})) \ni \mathtt{l}(\mathtt{x})$

La fonction `gamma_val` de concrétisation des valeurs numériques dépend elle-même de la concrétisation des intervalles.

Concrétisations paramétrées : si les variables locales peuvent contenir des références à des objets, la fonction de concrétisation sera paramétrée par le tas.

où sont passées les connexions de Galois ?

Spécification de l'analyse

Chaque instruction donne lieu à une ou plusieurs contraintes de la forme

$$F^\sharp(\sigma^\sharp(i)) \leq \sigma^\sharp(j)$$

avec i et j : points de programme avant et après l'instruction

```
Record cstr := { source : pc;
                 target : pc;
                 transfer : MemLat.t → MemLat.t }.
```

$mem^\sharp$

Traduction en prédicats Coq

```
Definition cstr2prop (S:StateLat.t) (c:cstr) : Prop :=
MemLat.order (c.(transfer) (StateLat.get S c.(source)))
             (StateLat.get S c.(target)).
```

$c \vdash S$

Il faut ensuite générer les contraintes pour chaque instruction

```
Definition gen_constraint (i:pc) (ins:instruction) : list cstr := ...
```

et les collecter pour l'ensemble du programme

```
Definition Spec (p:program) (S:StateLat.t) : Prop :=
```

$P \vdash S$

Correction de l'analyse

Repose sur le lemme suivant

```
Lemma step_correct :
∀ p S i ins l s s2,
instr_at p i = Some ins →
( ∀ c, In c (gen_constraint i ins) → cstr2prop S c) →
step (St i s l) s2 →
gamma p S (St i s l) →
gamma p S s2.
```

S vérifie les contraintes générées pour `ins`

si $< \mathtt{i}, \mathtt{s}, \mathtt{l} > \rightarrow \mathtt{s2}$ et $\gamma(\mathtt{S}) \ni < \mathtt{i}, \mathtt{s}, \mathtt{l} >$
alors $\gamma(\mathtt{S}) \ni \mathtt{s2}$

On montre alors que toute solution du système de contraintes est une sur-approximation de l'ensemble des états atteignables

comment calcule-t-on une solution ?

Questions de constructivité

Questions de constructivité

1. Connexions de Galois

Questions de constructivité

1. Connexions de Galois
   - Exemple : congruence modulo 2

Questions de constructivité

1. Connexions de Galois

- Exemple : congruence modulo 2

$$\begin{array}{rcl} \gamma(\bot) & = & \emptyset \\ \gamma(\overline{0}) & = & 2\mathbb{Z} \\ \gamma(\overline{1}) & = & 2\mathbb{Z}+1 \\ \gamma(\top) & = & \mathbb{Z} \end{array} \qquad \alpha(P) = \begin{cases} \bot & \text{if} \quad P = \emptyset \\ \overline{0} & \text{if} \quad P \neq \emptyset \text{ and } P \subseteq 2\mathbb{Z} \\ \overline{1} & \text{if} \quad P \neq \emptyset \text{ and } P \subseteq 2\mathbb{Z}+1 \\ \top & \text{if} \quad P \not\subseteq 2\mathbb{Z} \text{ and } P \not\subseteq 2\mathbb{Z}+1 \end{cases}$$

Questions de constructivité

1. Connexions de Galois

- Exemple : congruence modulo 2

$$\begin{array}{rcl} \gamma(\bot) & = & \emptyset \\ \gamma(\overline{0}) & = & 2\mathbb{Z} \\ \gamma(\overline{1}) & = & 2\mathbb{Z}+1 \\ \gamma(\top) & = & \mathbb{Z} \end{array} \qquad \alpha(P) = \begin{cases} \bot & \text{if} \quad P = \emptyset \\ \overline{0} & \text{if} \quad P \neq \emptyset \text{ and } P \subseteq 2\mathbb{Z} \\ \overline{1} & \text{if} \quad P \neq \emptyset \text{ and } P \subseteq 2\mathbb{Z}+1 \\ \top & \text{if} \quad P \not\subseteq 2\mathbb{Z} \text{ and } P \not\subseteq 2\mathbb{Z}+1 \end{cases}$$

La fonction α n'est pas calculable :

```
Definition alpha (P:𝒫(integer)) : {0,1} := ???
```

Questions de constructivité

1. Connexions de Galois

- Exemple : congruence modulo 2

$$\begin{array}{rcl} \gamma(\bot) & = & \emptyset \\ \gamma(\overline{0}) & = & 2\mathbb{Z} \\ \gamma(\overline{1}) & = & 2\mathbb{Z}+1 \\ \gamma(\top) & = & \mathbb{Z} \end{array} \qquad \alpha(P) = \begin{cases} \bot & \text{if} \quad P = \emptyset \\ \overline{0} & \text{if} \quad P \neq \emptyset \text{ and } P \subseteq 2\mathbb{Z} \\ \overline{1} & \text{if} \quad P \neq \emptyset \text{ and } P \subseteq 2\mathbb{Z}+1 \\ \top & \text{if} \quad P \not\subseteq 2\mathbb{Z} \text{ and } P \not\subseteq 2\mathbb{Z}+1 \end{cases}$$

La fonction α n'est pas calculable :

```
Definition alpha (P:𝒫(integer)) : {0,1} := ???
```

On se contente donc de fonctions de concrétisation

Questions de constructivité

1. Connexions de Galois

- Exemple : congruence modulo 2

$$\begin{array}{rcl} \gamma(\bot) & = & \emptyset \\ \gamma(\overline{0}) & = & 2\mathbb{Z} \\ \gamma(\overline{1}) & = & 2\mathbb{Z}+1 \\ \gamma(\top) & = & \mathbb{Z} \end{array} \qquad \alpha(P) = \begin{cases} \bot & \text{if} \quad P = \emptyset \\ \overline{0} & \text{if} \quad P \neq \emptyset \text{ and } P \subseteq 2\mathbb{Z} \\ \overline{1} & \text{if} \quad P \neq \emptyset \text{ and } P \subseteq 2\mathbb{Z}+1 \\ \top & \text{if} \quad P \not\subseteq 2\mathbb{Z} \text{ and } P \not\subseteq 2\mathbb{Z}+1 \end{cases}$$

La fonction α n'est pas calculable :

```
Definition alpha (P:𝒫(integer)) : {0,1} := ???
```

On se contente donc de fonctions de concrétisation

- monotones : composition et points fixes

Questions de constructivité

1. Connexions de Galois

- Exemple : congruence modulo 2

$$\begin{array}{rcl} \gamma(\bot) & = & \emptyset \\ \gamma(\overline{0}) & = & 2\mathbb{Z} \\ \gamma(\overline{1}) & = & 2\mathbb{Z}+1 \\ \gamma(\top) & = & \mathbb{Z} \end{array} \qquad \alpha(P) = \begin{cases} \bot & \text{if} \quad P = \emptyset \\ \overline{0} & \text{if} \quad P \neq \emptyset \text{ and } P \subseteq 2\mathbb{Z} \\ \overline{1} & \text{if} \quad P \neq \emptyset \text{ and } P \subseteq 2\mathbb{Z}+1 \\ \top & \text{if} \quad P \not\subseteq 2\mathbb{Z} \text{ and } P \not\subseteq 2\mathbb{Z}+1 \end{cases}$$

La fonction α n'est pas calculable :

```
Definition alpha (P:𝒫(integer)) : {0,1} := ???
```

On se contente donc de fonctions de concrétisation

- monotones : composition et points fixes
- morphisme d'intersection binaire : combinaison de deux approximations correctes

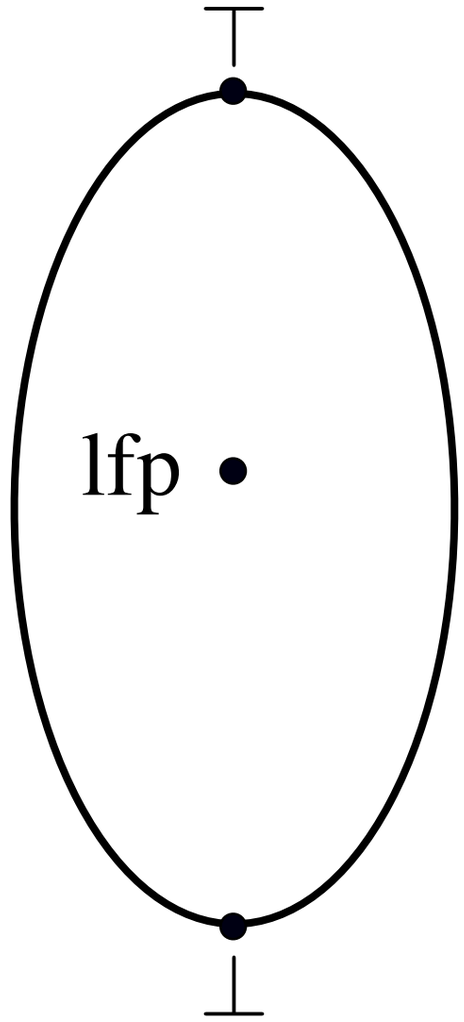
Questions de constructivité

2. Calculs de points fixes : terminaison

$$\bot, F^\sharp(\bot), \ldots, F^{\sharp n}(\bot), \ldots$$

- condition de chaîne ascendante
  - pas de preuve constructive en général
  - on remplace la condition classique par une notion d'ordre bien fondé

- opérateurs d'élargissement
  - même technique, par ordre bien fondé
  - définition équivalente à l'opérateur « classique »



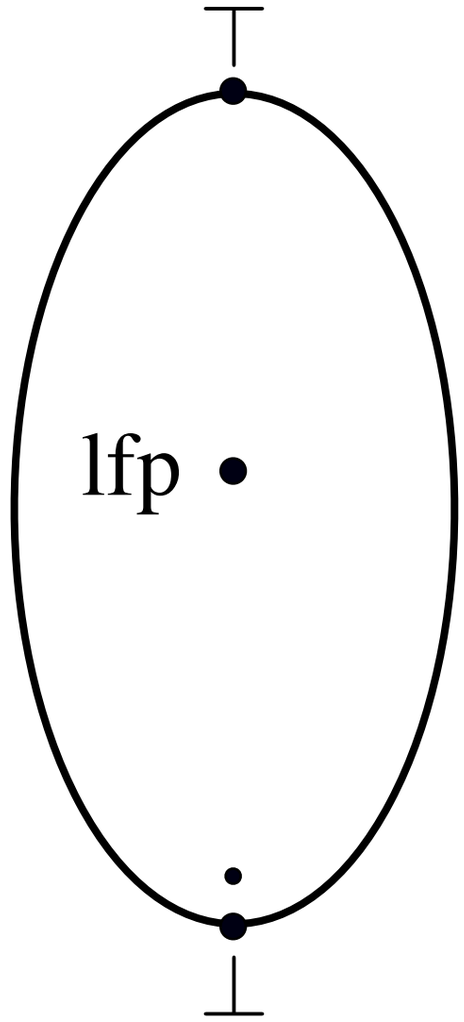
Questions de constructivité

2. Calculs de points fixes : terminaison

$$\bot, F^\sharp(\bot), \ldots, F^{\sharp n}(\bot), \ldots$$

- condition de chaîne ascendante
  - pas de preuve constructive en général
  - on remplace la condition classique par une notion d'ordre bien fondé

- opérateurs d'élargissement
  - même technique, par ordre bien fondé
  - définition équivalente à l'opérateur « classique »



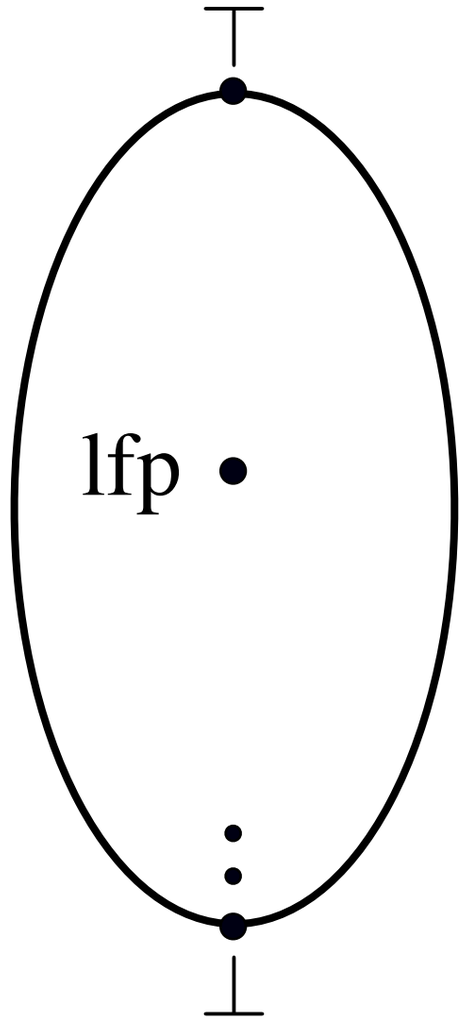
Questions de constructivité

2. Calculs de points fixes : terminaison

$$\bot, F^\sharp(\bot), \ldots, F^{\sharp n}(\bot), \ldots$$

- condition de chaîne ascendante
  - pas de preuve constructive en général
  - on remplace la condition classique par une notion d'ordre bien fondé

- opérateurs d'élargissement
  - même technique, par ordre bien fondé
  - définition équivalente à l'opérateur « classique »



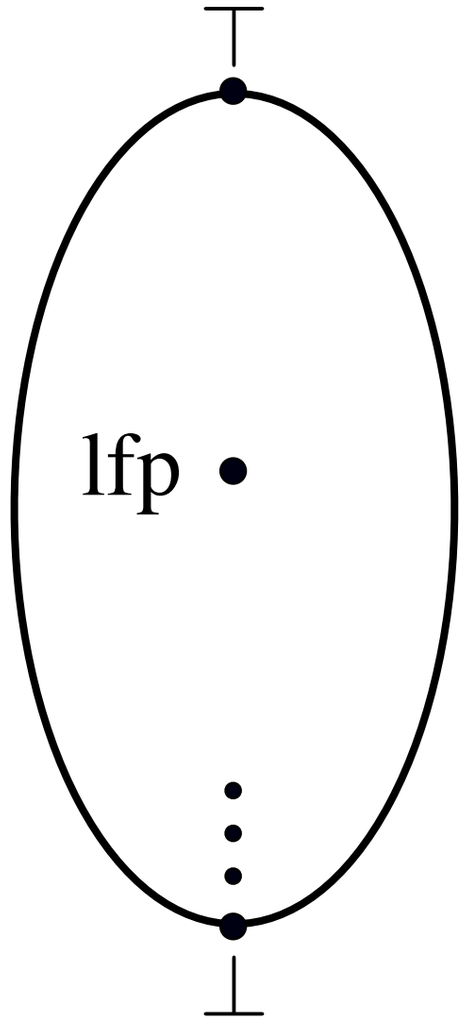
Questions de constructivité

2. Calculs de points fixes : terminaison

$$\bot, F^\sharp(\bot), \ldots, F^{\sharp n}(\bot), \ldots$$

- condition de chaîne ascendante
  - pas de preuve constructive en général
  - on remplace la condition classique par une notion d'ordre bien fondé

- opérateurs d'élargissement
  - même technique, par ordre bien fondé
  - définition équivalente à l'opérateur « classique »



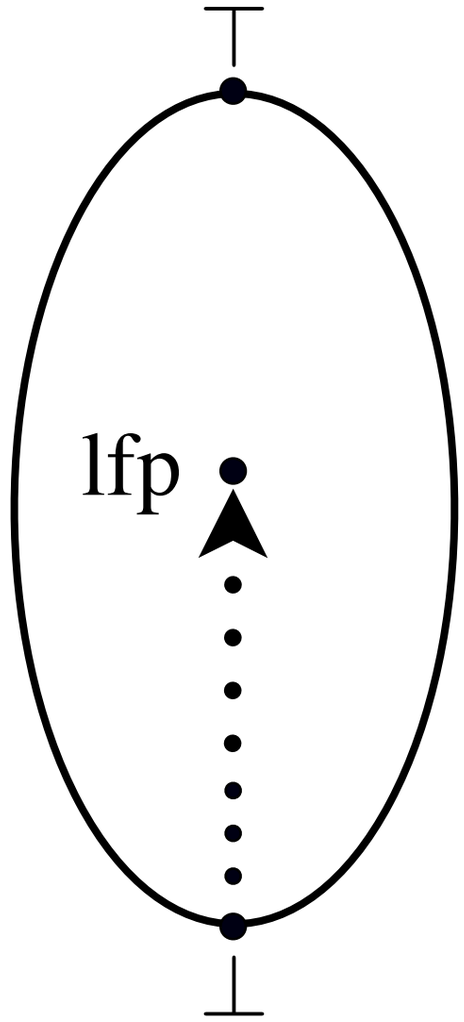
Questions de constructivité

2. Calculs de points fixes : terminaison

$$\bot, F^\sharp(\bot), \ldots, F^{\sharp n}(\bot), \ldots$$

- condition de chaîne ascendante
  - pas de preuve constructive en général
  - on remplace la condition classique par une notion d'ordre bien fondé

- opérateurs d'élargissement
  - même technique, par ordre bien fondé
  - définition équivalente à l'opérateur « classique »



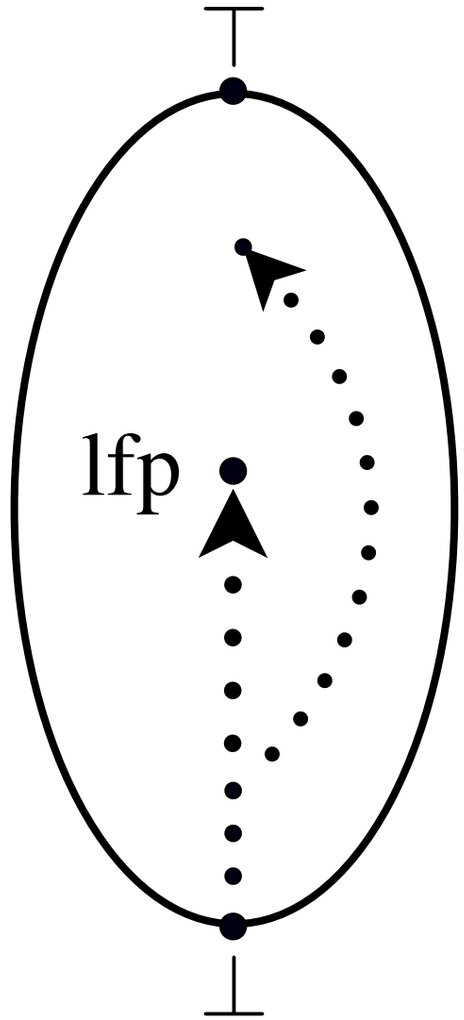
Questions de constructivité

2. Calculs de points fixes : terminaison

$$\bot, F^\sharp(\bot), \ldots, F^{\sharp n}(\bot), \ldots$$

- condition de chaîne ascendante
  - pas de preuve constructive en général
  - on remplace la condition classique par une notion d'ordre bien fondé

- opérateurs d'élargissement
  - même technique, par ordre bien fondé
  - définition équivalente à l'opérateur « classique »



Questions de constructivité

3. Construction des domaines abstraits (treillis)

Des modules Coq, munis de paramètres

- usuels : *order, join, meet*
- preuves : la relation *order* est antisymétrique (etc.) et calculable

Combinaison par foncteurs

- treillis de base (signes, intervalles, etc.)
- produit, fonctions de domaine fini, tableaux à indices bornés, etc.

Construction aisée de treillis complexes

- avec opérateurs d'élargissement
- et fonctions de calcul de (post) point fixes

Utilisation de *Map* (arbres binaires) pour une complexité raisonnable

Études de cas

- Langage de byte code à la Javacard
  - motivation historique : logiciel embarqué
  - langage assez riche pour démontrer les possibilités du cadre
- Analyse générique de type « flot de données »
  - interprocédurale
  - objets abstraits par leur classe
- Analyse de consommation mémoire
  - algorithmique simple de graphes
  - évaluation de l'efficacité des structures de données

ENS CACHAN BRETAGNE

Bilan

Contributions

- identifier une partie de l'interprétation abstraite compatible avec la logique constructive de Coq
- construction d'analyseurs statiques certifiés
  - par extraction à partir de leur preuve
  - avec un souci d'efficacité
- application à des cas d'étude « réalistes »

Publications

- **Extracting a Data Flow Analyser in Constructive Logic**. *Theoretical Computer Science*, 2005. Avec Thomas Jensen, David Pichardie et Vlad Rusu.
- **Certified Memory Usage Analysis**. FM 2005. Avec Thomas Jensen, David Pichardie et Gerardo Schneider.
- **Proof Pearl: A Certified Denotational Abstract Interpreter.** *ITP 2010*. Avec David Pichardie.
- **Comparing techniques for certified static analysis.** *NASA Formal Methods Symposium , 2009*. Avec David Pichardie.

Analyses statiques quantitatives

Motivation et feuille de route

Motivation et feuille de route

Analyse statique « classique » : réponse qualitative

$$[\![P]\!] \subseteq Safe$$

Motivation et feuille de route

Analyse statique « classique » : réponse qualitative

$$[\![P]\!] \subseteq Safe$$

Traiter des informations de nature quantitative

Motivation et feuille de route

Analyse statique « classique » : réponse qualitative

$$[\![P]\!] \subseteq Safe$$

Traiter des informations de nature quantitative

- consommation de ressources

Motivation et feuille de route

Analyse statique « classique » : réponse qualitative

$$[\![P]\!] \subseteq Safe$$

Traiter des informations de nature quantitative

- consommation de ressources
- quantifier la précision d'une analyse

Motivation et feuille de route

Analyse statique « classique » : réponse qualitative

$$[\![P]\!] \subseteq Safe$$

Traiter des informations de nature quantitative

- consommation de ressources
- quantifier la précision d'une analyse

- Introduire des notions quantitatives dans la sémantique

Motivation et feuille de route

Analyse statique « classique » : réponse qualitative

$$[\![P]\!] \subseteq Safe$$

Traiter des informations de nature quantitative

- consommation de ressources
- quantifier la précision d'une analyse

- Introduire des notions quantitatives dans la sémantique
- En tirer des informations quantitatives sur le déroulement d'un programme

Motivation et feuille de route

Analyse statique « classique » : réponse qualitative

$$[\![P]\!] \subseteq Safe$$

Traiter des informations de nature quantitative

- consommation de ressources
- quantifier la précision d'une analyse

- Introduire des notions quantitatives dans la sémantique
- En tirer des informations quantitatives sur le déroulement d'un programme
- Trouver un moyen d'abstraire la sémantique

Motivation et feuille de route

Analyse statique « classique » : réponse qualitative

$$[\![P]\!] \subseteq Safe$$

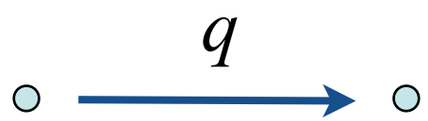
Traiter des informations de nature quantitative

- consommation de ressources
- quantifier la précision d'une analyse

- Introduire des notions quantitatives dans la sémantique
- En tirer des informations quantitatives sur le déroulement d'un programme
- Trouver un moyen d'abstraire la sémantique
- Définir une notion d'approximation correcte

Sémantique avec coûts

Système de transitions étiqueté



Le coût peut modéliser plusieurs types de ressources

temps, mémoire, énergie...

Il peut être nécessaire d'enrichir la structure des états

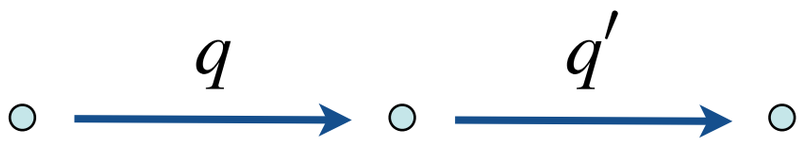
ex : tenir compte du contenu du cache dans la sémantique

Composer les transitions, donc les coûts

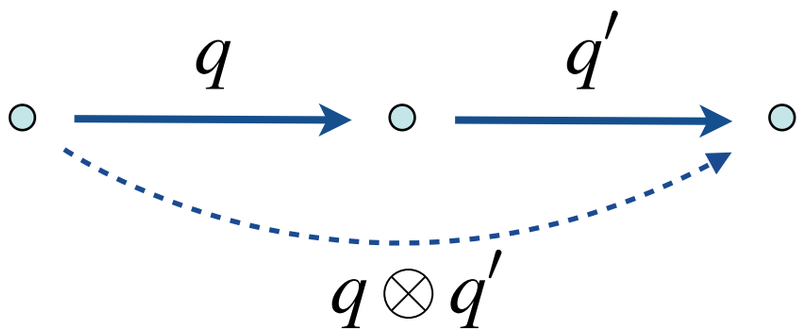
opérateurs algébriques sur les coûts

Dioïdes de coût

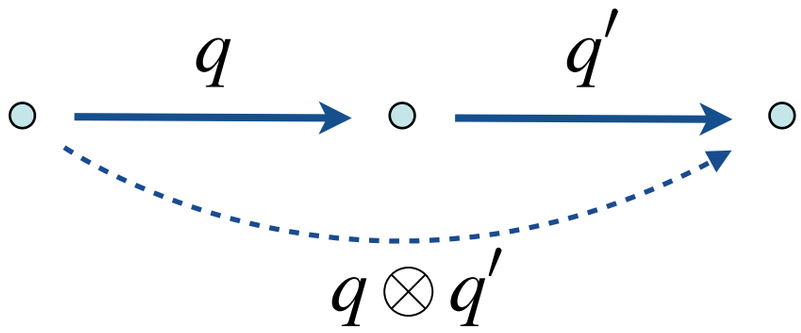
Dioïdes de coût

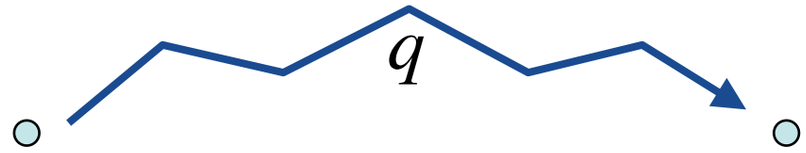


Dioïdes de coût

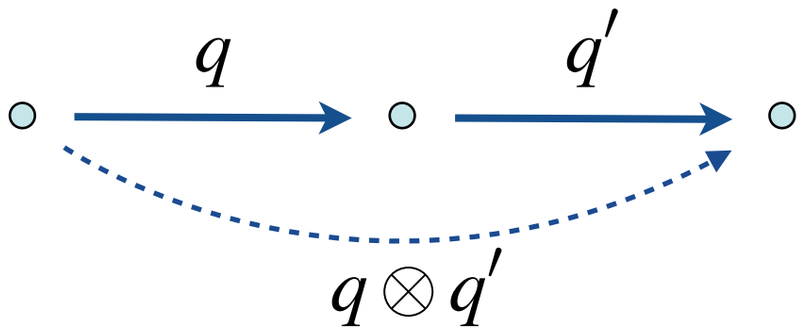


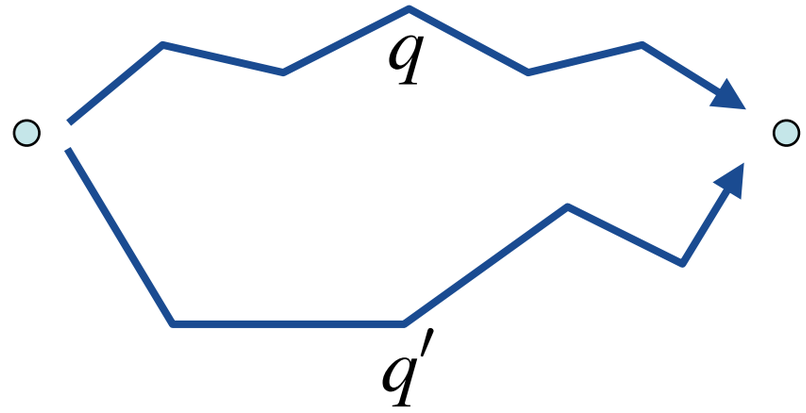
Dioïdes de coût



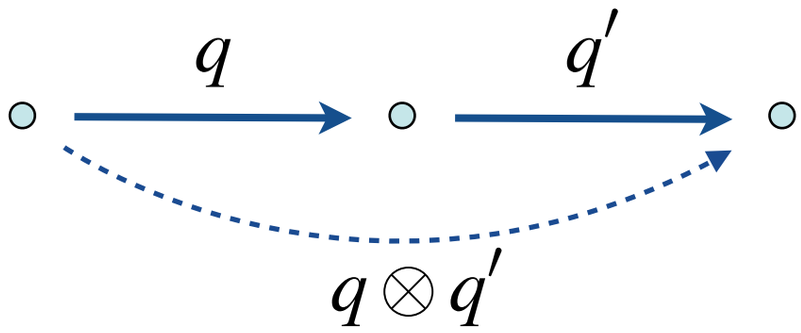


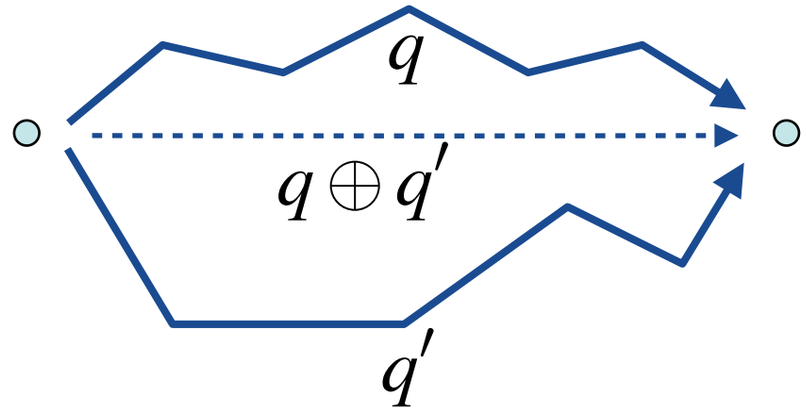
Dioïdes de coût



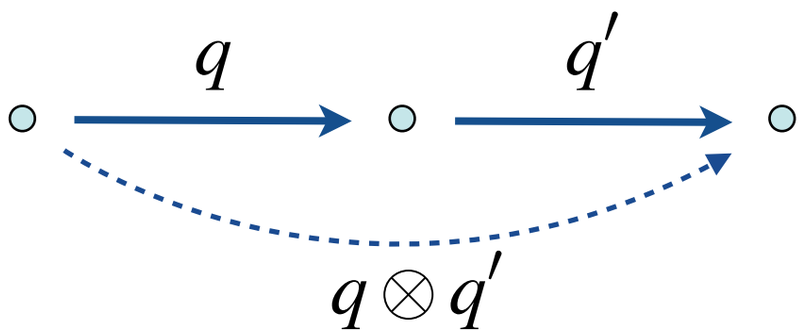


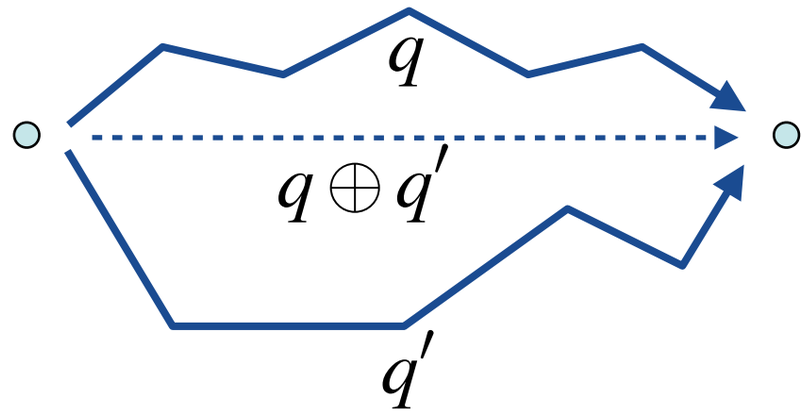
Dioïdes de coût





Dioïdes de coût





Structure de semi-anneau commutatif

- associativité, distributivité...
- éléments particuliers e et $\perp$

Idempotence : $q \oplus q = q$

Notion d'ordre : $a \leq_{\oplus} b \Leftrightarrow \exists c : a \oplus c = b$

\+ : comparer les coûts

\- : incompatible avec la notion d'anneau

Dioïdes de coût

Dioïde complet : pouvoir parler de sommes infinies

$$\bigoplus_{q \in S} q$$

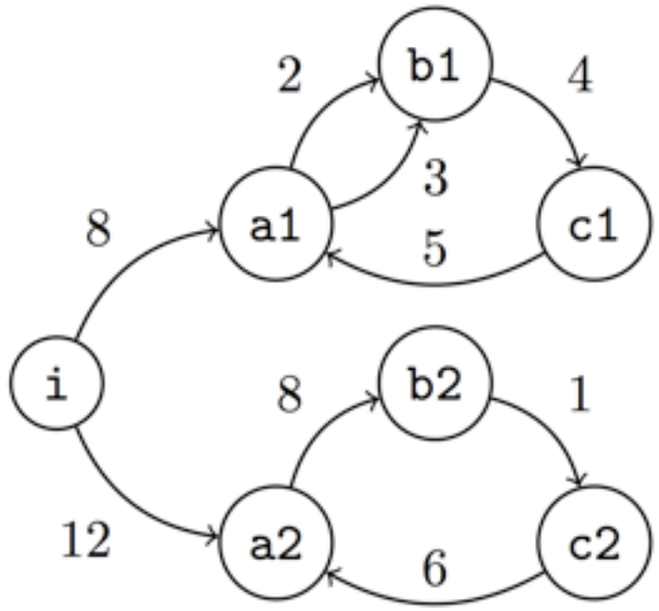
Existence d'une racine n-ième pour tout n : coût moyen

$$\sqrt[3]{q \otimes q \otimes q} = q$$

Exemple type :

$$(\mathbb{R} \cup \{+\infty, -\infty\}, \max, +)$$

Sémantique concrète : opérateur linéaire



$$
\begin{array}{c} \\ \texttt{a1} \\ \\ \\ \texttt{a2} \\ \\ \\ \end{array}
\begin{array}{c} \texttt{i} \\ \begin{pmatrix} \bot & \bot & \bot & \bot & \bot & \bot & \bot \\ 8 & \bot & \bot & 5 & \bot & \bot & \bot \\ \bot & 3 & \bot & \bot & \bot & \bot & \bot \\ \bot & \bot & 4 & \bot & \bot & \bot & \bot \\ 12 & \bot & \bot & \bot & \bot & \bot & 6 \\ \bot & \bot & \bot & \bot & 8 & \bot & \bot \\ \bot & \bot & \bot & \bot & \bot & 1 & \bot \end{pmatrix} \end{array}
$$

état $\texttt{a1} \rightsquigarrow (\bot, e, \bot, \bot, \bot, \bot, \bot)^t$

Matrice M : un pas d'exécution de la sémantique étiquetée

Produit de matrices : M^n contient les coûts des chemins de longueur n

Ordre sur les vecteurs et matrices : point-à-point, induit par $\leq_{\oplus}$

Calculs de coût : coût global

Calculs de coût : coût global

Coût « maximum » de tous les chemins joignant états initiaux I et finaux F

Calculs de coût : coût global

Coût « maximum » de tous les chemins joignant états initiaux I et finaux F

Puissances successives de la matrice M

$$M^{+} = \bigoplus_{i=1}^{\infty} M^{i}$$

Calculs de coût : coût global

Coût « maximum » de tous les chemins joignant états initiaux I et finaux F

Puissances successives de la matrice M

$$M^{+} = \bigoplus_{i=1}^{\infty} M^{i}$$

$$gc(P) = \bigoplus \{M^{+}_{i,f} | i \in I, f \in F\}$$

Calculs de coût : coût global

Coût « maximum » de tous les chemins joignant états initiaux I et finaux F

Puissances successives de la matrice M

$$M^+ = \bigoplus_{i=1}^{\infty} M^i$$

$$gc(P) = \bigoplus \{M^+_{i,f} | i \in I, f \in F\}$$

Boucle de coût non nul : coût global infini

Calculs de coût : coût global

Coût « maximum » de tous les chemins joignant états initiaux I et finaux F

Puissances successives de la matrice M

$$M^+ = \bigoplus_{i=1}^{\infty} M^i$$

$$gc(P) = \bigoplus \{M^+_{i,f} | i \in I, f \in F\}$$

Boucle de coût non nul : coût global infini

Systèmes réactifs : pas vraiment d'état final

Calculs de coût : coût *long-run*

Calculs de coût : coût *long-run*

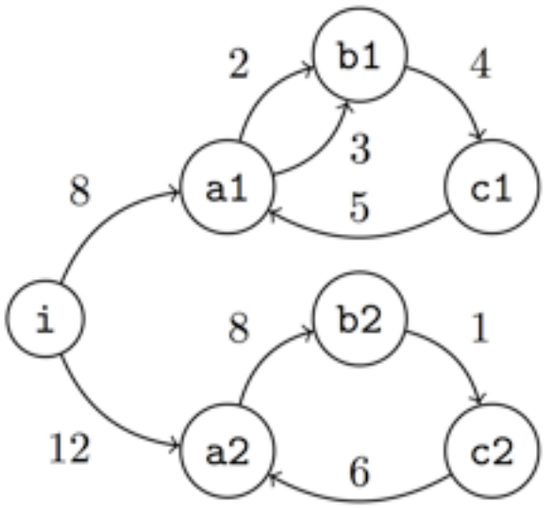
Notion inspirée des systèmes à événement discrets

Coût moyen « maximal » d'un cycle

Calculs de coût : coût *long-run*

Notion inspirée des systèmes à événement discrets

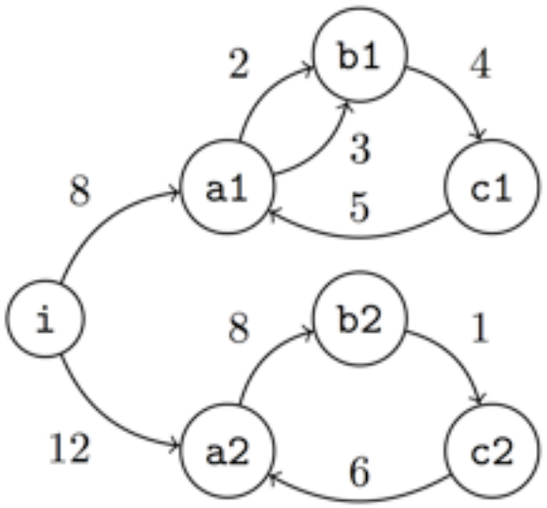
Coût moyen « maximal » d'un cycle



Calculs de coût : coût *long-run*

Notion inspirée des systèmes à événement discrets

Coût moyen « maximal » d'un cycle

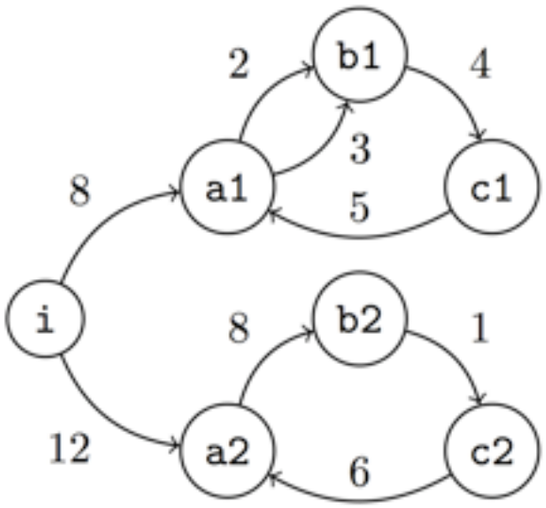


$$\left.\begin{array}{l} \sqrt[3]{(2 \oplus 3) \otimes 4 \otimes 5} = 4 \\ \sqrt[3]{8 \otimes 1 \otimes 6} = 5 \end{array}\right\} 4 \oplus 5 = 5$$

Calculs de coût : coût *long-run*

Notion inspirée des systèmes à événement discrets

Coût moyen « maximal » d'un cycle



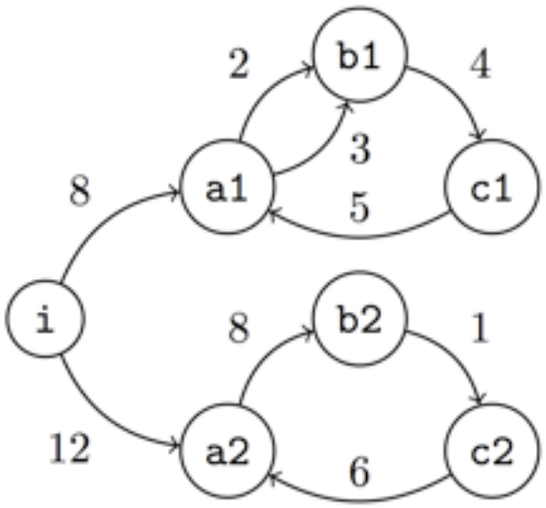
$$\left.\begin{array}{l} \sqrt[3]{(2 \oplus 3) \otimes 4 \otimes 5} = 4 \\ \sqrt[3]{8 \otimes 1 \otimes 6} = 5 \end{array}\right\} \; 4 \oplus 5 = 5$$

$$\rho(M) = \bigoplus_k \sqrt[k]{tr\, M^k} \quad \text{avec} \quad tr\, A = \bigoplus_i A_{i,i}$$

Calculs de coût : coût *long-run*

Notion inspirée des systèmes à événement discrets

Coût moyen « maximal » d'un cycle



$$\left.\begin{array}{l} \sqrt[3]{(2 \oplus 3) \otimes 4 \otimes 5} = 4 \\ \sqrt[3]{8 \otimes 1 \otimes 6} = 5 \end{array}\right\} \; 4 \oplus 5 = 5$$

$$\rho(M) = \bigoplus_k \sqrt[k]{tr\, M^k} \quad \text{avec} \quad tr\, A = \bigoplus_i A_{i,i}$$

Th : ρ coïncide avec le coût moyen maximal d'une exécution arbitrairement longue

Abstraction

Ensemble d'états trop grand, voire infini

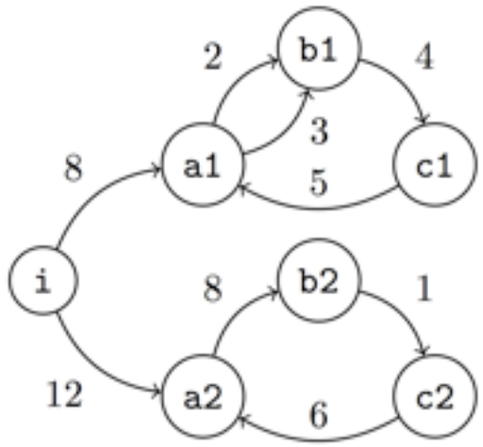
- on retrouve les problèmes classiques de non-calculabilité
- analyses statiques ?

Défi : définir une méthode d'approximation de la sémantique qui tienne compte

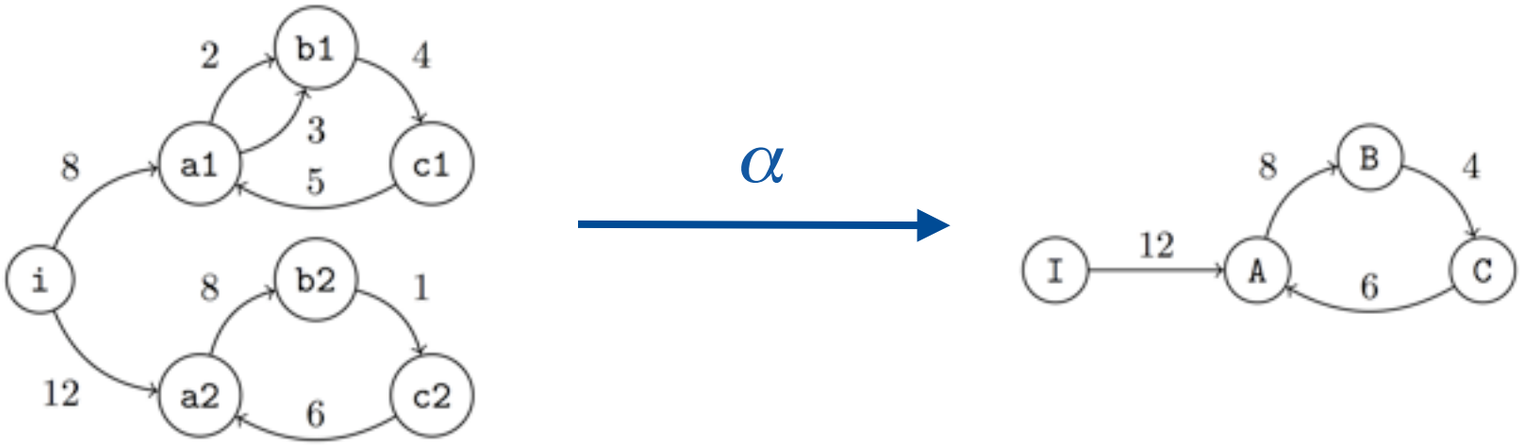
- des états : abstraire le domaine des états
- des coûts : approximation par rapport à $\leq_{\oplus}$

NB : pas d'abstraction des coûts, on garde le même dioïde

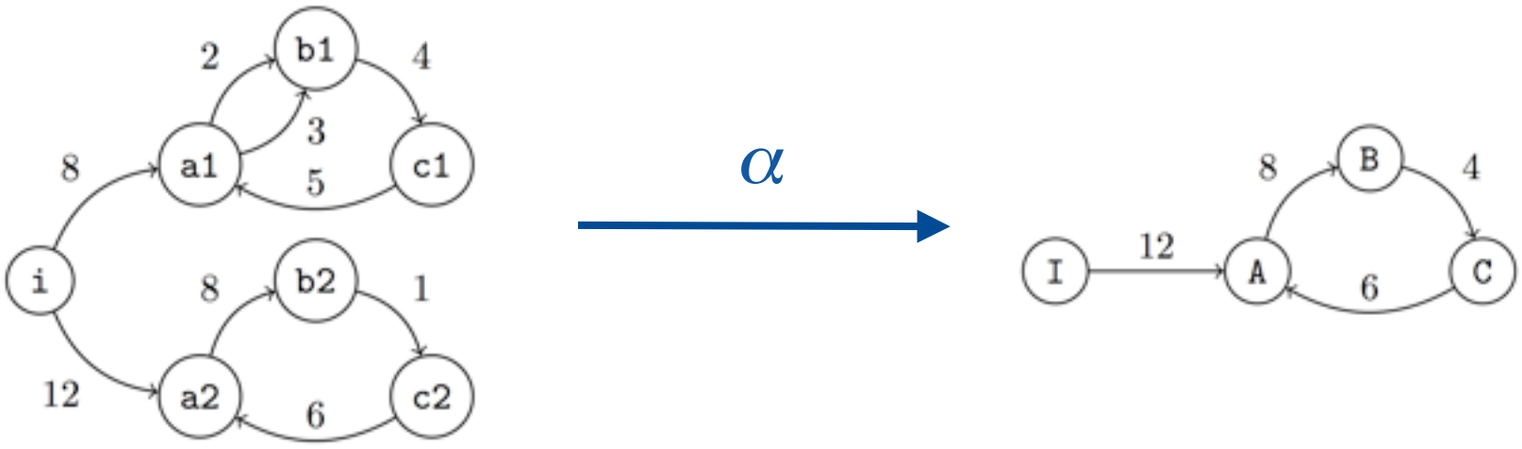
Abstraction par partition



Abstraction par partition



Abstraction par partition

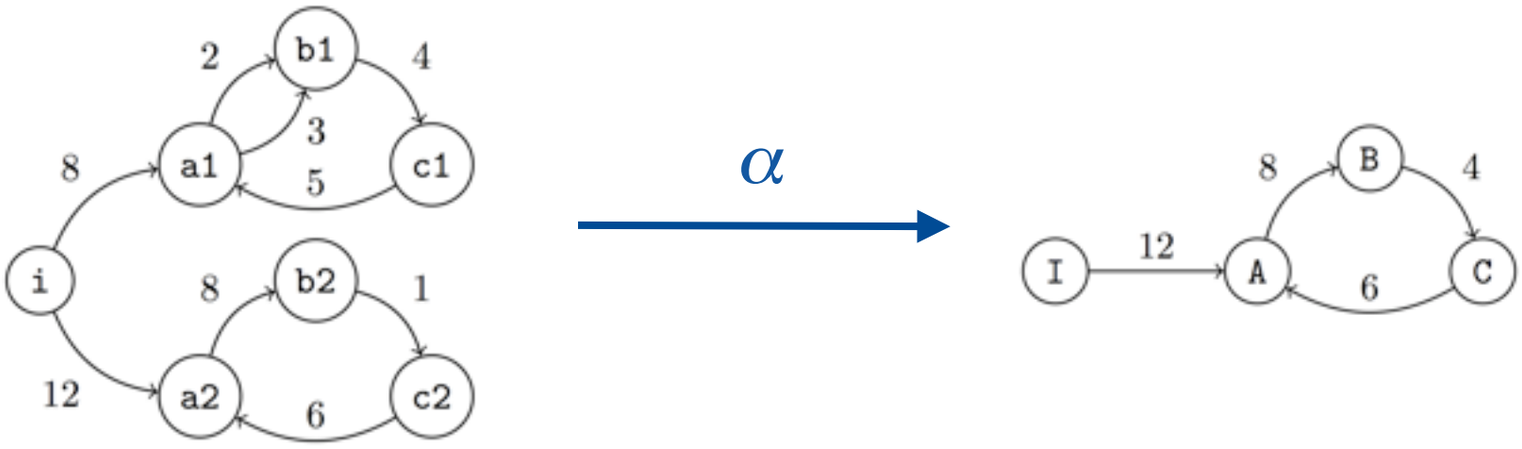


L'abstraction s'exprime aisément sous forme matricielle

$$\overline{\alpha} = \begin{pmatrix} e & \bot & \bot & \bot & \bot & \bot & \bot \\ \bot & e & \bot & \bot & e & \bot & \bot \\ \bot & \bot & e & \bot & \bot & e & \bot \\ \bot & \bot & \bot & e & \bot & \bot & e \end{pmatrix} \begin{matrix} \mathtt{I} \\ \mathtt{A} \\ \mathtt{B} \\ \mathtt{C} \end{matrix}$$

$$\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathtt{c1} \qquad\qquad \mathtt{c2}$$

Abstraction par partition



L'abstraction s'exprime aisément sous forme matricielle

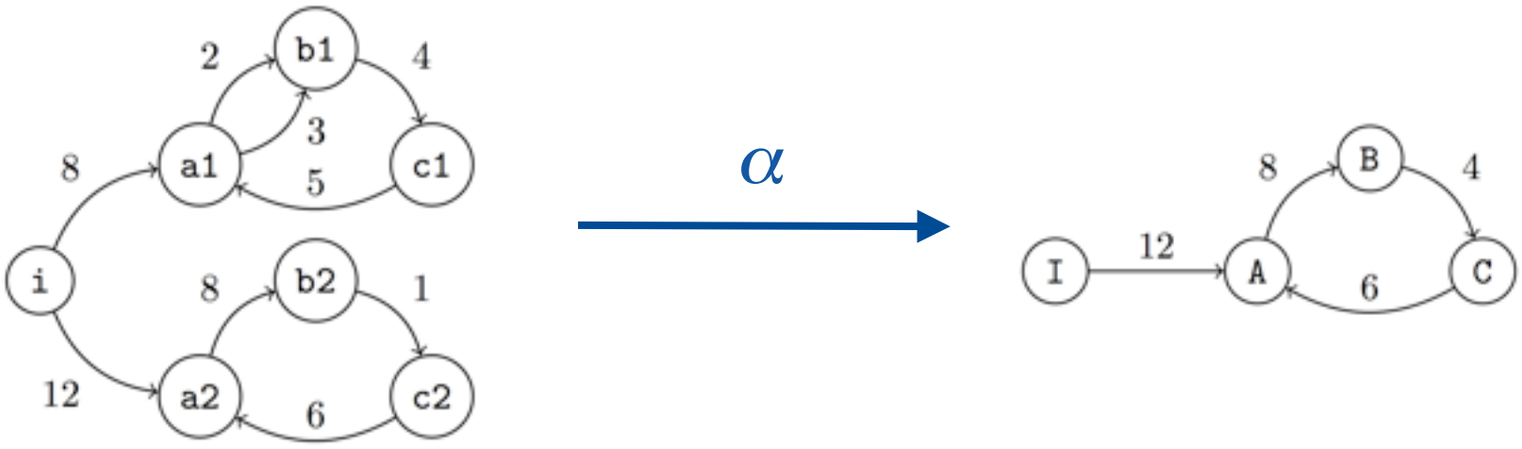
$$\overline{\alpha} = \begin{pmatrix} e & \bot & \bot & \bot & \bot & \bot & \bot \\ \bot & e & \bot & \bot & e & \bot & \bot \\ \bot & \bot & e & \bot & \bot & e & \bot \\ \bot & \bot & \bot & e & \bot & \bot & e \end{pmatrix}$$

Il existe une matrice $\overline{\gamma}$ telle que

$$\overline{\alpha} \circ \overline{\gamma} \leq_{\oplus} Id$$

$$Id \leq_{\oplus} \overline{\gamma} \circ \overline{\alpha}$$

Abstraction par partition



L'abstraction s'exprime aisément sous forme matricielle

$$\overline{\alpha} = \begin{pmatrix} e & \bot & \bot & \bot & \bot & \bot & \bot \\ \bot & e & \bot & \bot & e & \bot & \bot \\ \bot & \bot & e & \bot & \bot & e & \bot \\ \bot & \bot & \bot & e & \bot & \bot & e \end{pmatrix}$$

Il existe une matrice $\overline{\gamma}$ telle que

$$\overline{\alpha} \circ \overline{\gamma} \leq_{\oplus} Id$$

$$Id \leq_{\oplus} \overline{\gamma} \circ \overline{\alpha}$$

On peut définir $M^{\sharp} = \overline{\alpha} \circ M \circ \overline{\gamma}$ comme sémantique abstraite

composition = produit de matrices

Correction des abstractions

Critère de correction

$$\overline{\alpha} \circ M \leq_{\oplus} M^{\sharp} \circ \overline{\alpha}$$

sous cette hypothèse, on a

Th : la sémantique abstraite permet de calculer une approximation des coûts concrets

$$gc(M) \leq_{\oplus} gc(M^{\sharp})$$
$$\rho(M) \leq_{\oplus} \rho(M^{\sharp})$$

La matrice définie par $M^{\sharp} = \overline{\alpha} \circ M \circ \overline{\gamma}$

- vérifie naturellement cette condition
- est la meilleure (plus petite au sens de $\oplus$) abstraction

Limitations

Adapté à des cas simples

fusionner des états = oublier des informations

mais

- gourmand en espace : 1 dimension par élément concret ou abstrait
- ne peut rendre compte des abstractions usuelles

ex : les états contiennent des variables entières → on voudrait abstraire par des intervalles

Limitations

Adapté à des cas simples

fusionner des états = oublier des informations

mais

- gourmand en espace : 1 dimension par élément concret ou abstrait
- ne peut rendre compte des abstractions usuelles

ex : les états contiennent des variables entières → on voudrait abstraire par des intervalles

sur $\{1,2,3\}$:

Limitations

Adapté à des cas simples

fusionner des états = oublier des informations

mais

- gourmand en espace : 1 dimension par élément concret ou abstrait
- ne peut rendre compte des abstractions usuelles

ex : les états contiennent des variables entières → on voudrait abstraire par des intervalles

sur $\{1,2,3\}$: $\begin{pmatrix} e \\ \bot \\ \bot \\ \bot \\ \bot \\ \bot \\ \bot \end{pmatrix}$ $[]$

Limitations

Adapté à des cas simples

fusionner des états = oublier des informations

mais

- gourmand en espace : 1 dimension par élément concret ou abstrait
- ne peut rendre compte des abstractions usuelles

ex : les états contiennent des variables entières → on voudrait abstraire par des intervalles

sur $\{1,2,3\}$: $\begin{pmatrix} \bot \\ e \\ \bot \\ \bot \\ \bot \\ \bot \\ \bot \end{pmatrix}$ $[1]$

Limitations

Adapté à des cas simples

fusionner des états = oublier des informations

mais

- gourmand en espace : 1 dimension par élément concret ou abstrait
- ne peut rendre compte des abstractions usuelles

ex : les états contiennent des variables entières → on voudrait abstraire par des intervalles

sur $\{1,2,3\}$: $\begin{pmatrix} \bot \\ \bot \\ e \\ \bot \\ \bot \\ \bot \\ \bot \end{pmatrix}$ [2]

Limitations

Adapté à des cas simples

fusionner des états = oublier des informations

mais

- gourmand en espace : 1 dimension par élément concret ou abstrait
- ne peut rendre compte des abstractions usuelles

ex : les états contiennent des variables entières → on voudrait abstraire par des intervalles

sur $\{1,2,3\}$: $\begin{pmatrix} \bot \\ \bot \\ \bot \\ e \\ \bot \\ \bot \\ \bot \end{pmatrix}$ [3]

Limitations

Adapté à des cas simples

fusionner des états = oublier des informations

mais

- gourmand en espace : 1 dimension par élément concret ou abstrait
- ne peut rendre compte des abstractions usuelles

ex : les états contiennent des variables entières → on voudrait abstraire par des intervalles

sur $\{1,2,3\}$:

$$\begin{pmatrix} \bot \\ \bot \\ \bot \\ \bot \\ e \\ \bot \\ \bot \end{pmatrix} \quad [1,2]$$

Limitations

Adapté à des cas simples

fusionner des états = oublier des informations

mais

- gourmand en espace : 1 dimension par élément concret ou abstrait
- ne peut rendre compte des abstractions usuelles

ex : les états contiennent des variables entières → on voudrait abstraire par des intervalles

sur $\{1,2,3\}$:

$$\begin{pmatrix} \bot \\ \bot \\ \bot \\ \bot \\ \bot \\ e \\ \bot \end{pmatrix} \quad [2,3]$$

Limitations

Adapté à des cas simples

fusionner des états = oublier des informations

mais

- gourmand en espace : 1 dimension par élément concret ou abstrait
- ne peut rendre compte des abstractions usuelles

ex : les états contiennent des variables entières → on voudrait abstraire par des intervalles

sur $\{1,2,3\}$: $\begin{pmatrix} \bot \\ \bot \\ \bot \\ \bot \\ \bot \\ \bot \\ e \end{pmatrix}$ $[1,3]$

Limitations

Adapté à des cas simples

fusionner des états = oublier des informations

mais

- gourmand en espace : 1 dimension par élément concret ou abstrait
- ne peut rendre compte des abstractions usuelles

ex : les états contiennent des variables entières → on voudrait abstraire par des intervalles

sur $\{1,2,3\}$: $\begin{pmatrix} \bot \\ \bot \\ \bot \\ \bot \\ \bot \\ \bot \\ e \end{pmatrix}$ $[1,3]$

L'ordre sur les vecteurs est induit par l'ordre sur les coûts

on a $[1] \subseteq [1,3]$ $\begin{pmatrix} \bot \\ e \\ \bot \\ \bot \\ \bot \\ \bot \\ \bot \end{pmatrix} \leq_{\oplus} \begin{pmatrix} \bot \\ \bot \\ \bot \\ \bot \\ \bot \\ \bot \\ e \end{pmatrix}$

mais pas

Limitations

Adapté à des cas simples

fusionner des états = ou

codage compact des treillis en espaces de vecteurs

mais

- gourmand en espace : 1 dimension par élément concret ou abstrait
- ne peut rendre compte des abstractions usuelles

ex : les états contiennent des variables entières → on voudrait abstraire par des intervalles

sur $\{1,2,3\}$: $\begin{pmatrix} \bot \\ \bot \\ \bot \\ \bot \\ \bot \\ \bot \\ e \end{pmatrix}$ $[1,3]$

L'ordre sur les vecteurs est induit par l'ordre sur les coûts

on a $[1] \subseteq [1,3]$ $\begin{pmatrix} \bot \\ e \\ \bot \\ \bot \\ \bot \\ \bot \\ \bot \end{pmatrix} \leq_{\oplus} \begin{pmatrix} \bot \\ \bot \\ \bot \\ \bot \\ \bot \\ \bot \\ e \end{pmatrix}$

mais pas

Limitations

Adapté à des cas simples

fusionner des états = ou

codage compact des treillis en espaces de vecteurs

mais

- gourmand en espace : 1 dimension par élément concret ou abstrait
- ne peut rendre compte des abstractions usuelles

ex : le[illegible]iables entières → on voudrait abstraire par des intervalles

qui tienne compte des deux relations d'ordre

[illegible]re sur les vecteurs est induit par l'ordre sur les couts

sur $\{1,2,3\}$:

$$\begin{pmatrix} \bot \\ \bot \\ \bot \\ \bot \\ \bot \\ e \end{pmatrix} \quad [1,3]$$

on a $[1] \subseteq [1,3]$

mais pas

$$\begin{pmatrix} \bot \\ e \\ \bot \\ \bot \\ \bot \\ \bot \end{pmatrix} \leq_{\oplus} \begin{pmatrix} \bot \\ \bot \\ \bot \\ \bot \\ \bot \\ e \end{pmatrix}$$

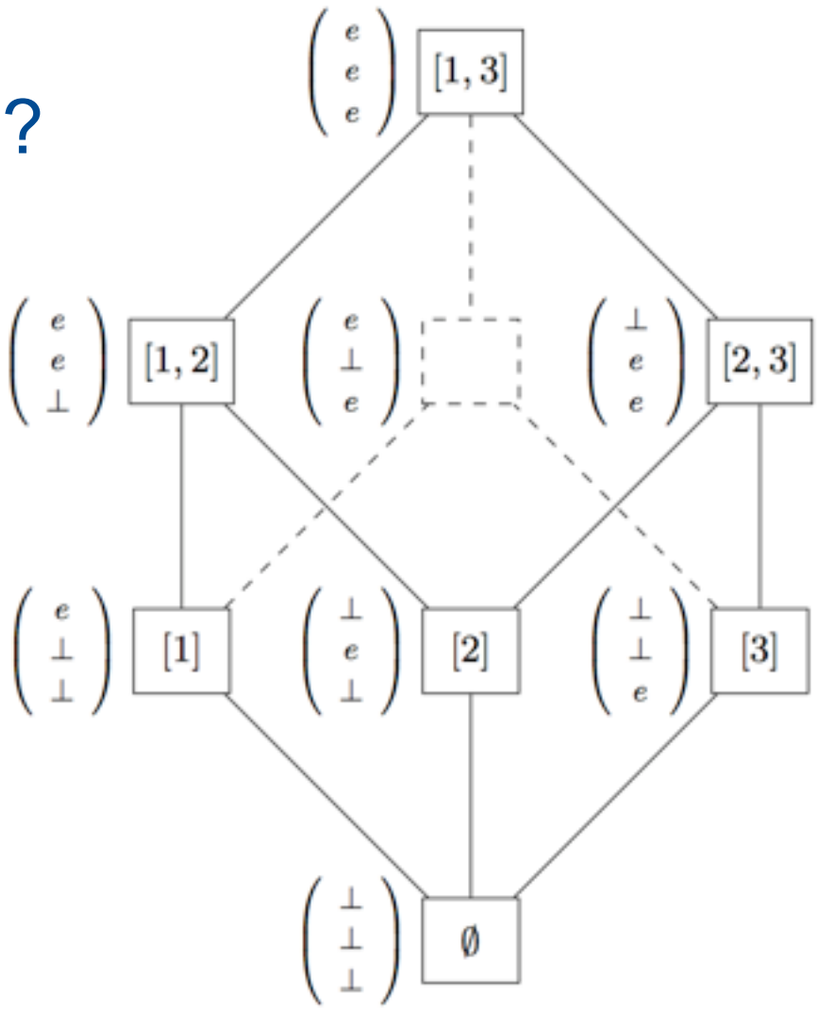
Codage des treillis

Le treillis des parties d'un ensemble de n éléments est naturellement représenté par un espace de dimension n

Ex. avec $\{1,2,3\} : \{1,3\}$ → $\begin{pmatrix} e \\ \perp \\ e \end{pmatrix}$

Pour un treillis complet quelconque ?

Ex. intervalles



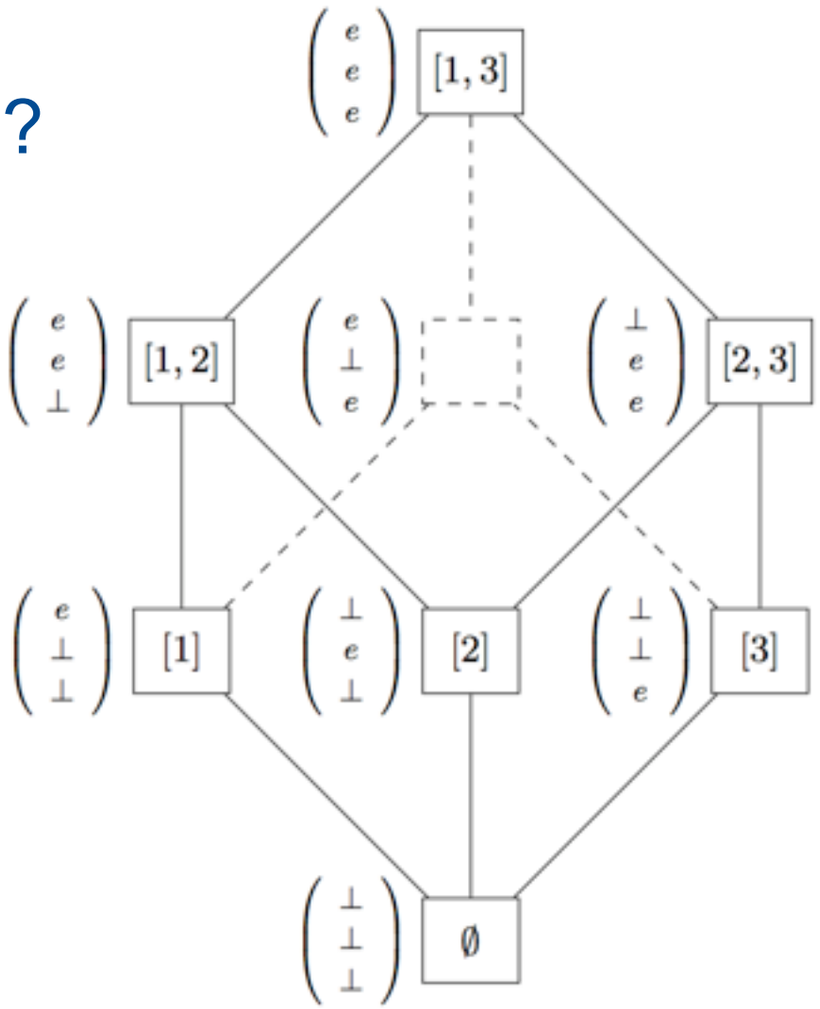
Codage des treillis

Le treillis des parties d'un ensemble de n éléments est naturellement représenté par un espace de dimension n

Ex. avec $\{1,2,3\} : \{1,3\}$ → $\begin{pmatrix} e \\ \perp \\ e \end{pmatrix}$

Pour un treillis complet quelconque ?

Ex. intervalles



généralisation à tout treillis complet

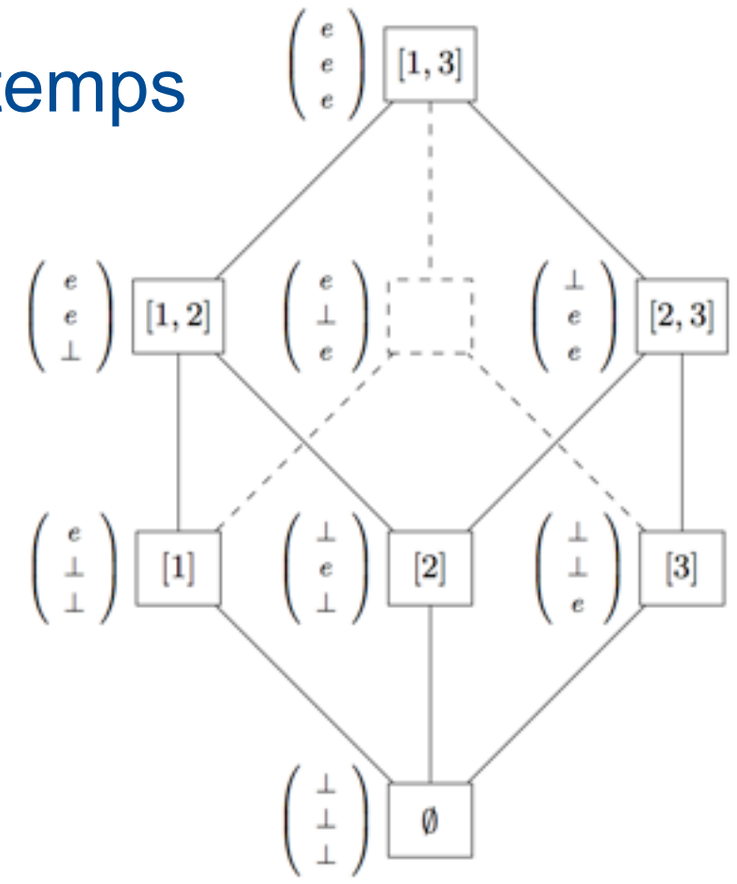
Abstraction revisitée

$$\alpha(\{1,3\}) = [1,3] \rightarrow \overline{\alpha}(\begin{pmatrix} e \\ \bot \\ e \end{pmatrix}) = \begin{pmatrix} e \\ e \\ e \end{pmatrix}$$

Abstraction revisitée

$$\alpha(\{1,3\}) = [1,3] \rightarrow \overline{\alpha}\left(\begin{pmatrix} e \\ \bot \\ e \end{pmatrix}\right) = \begin{pmatrix} e \\ e \\ e \end{pmatrix}$$

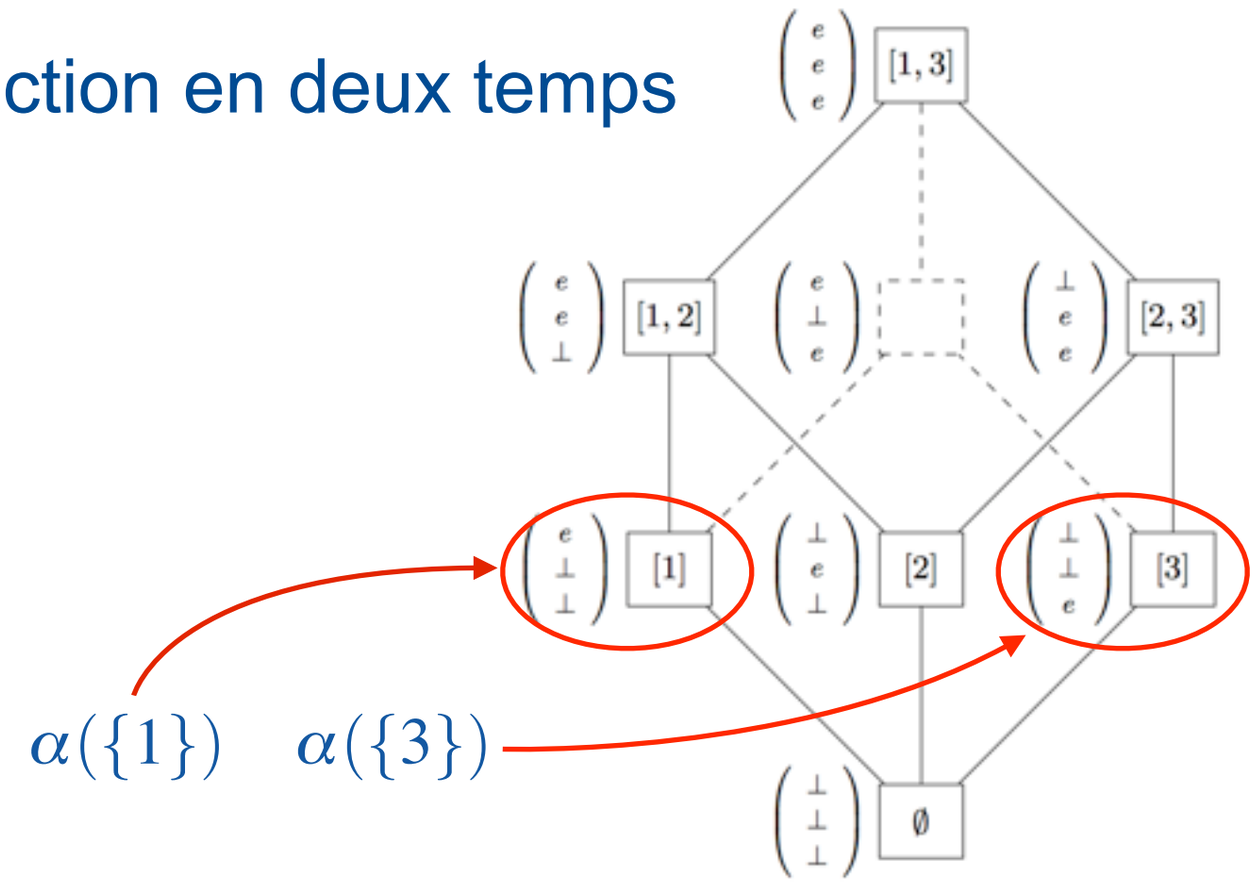
Abstraction en deux temps



Abstraction revisitée

$$\alpha(\{1,3\}) = [1,3] \rightarrow \overline{\alpha}\begin{pmatrix} e \\ \bot \\ e \end{pmatrix} = \begin{pmatrix} e \\ e \\ e \end{pmatrix}$$

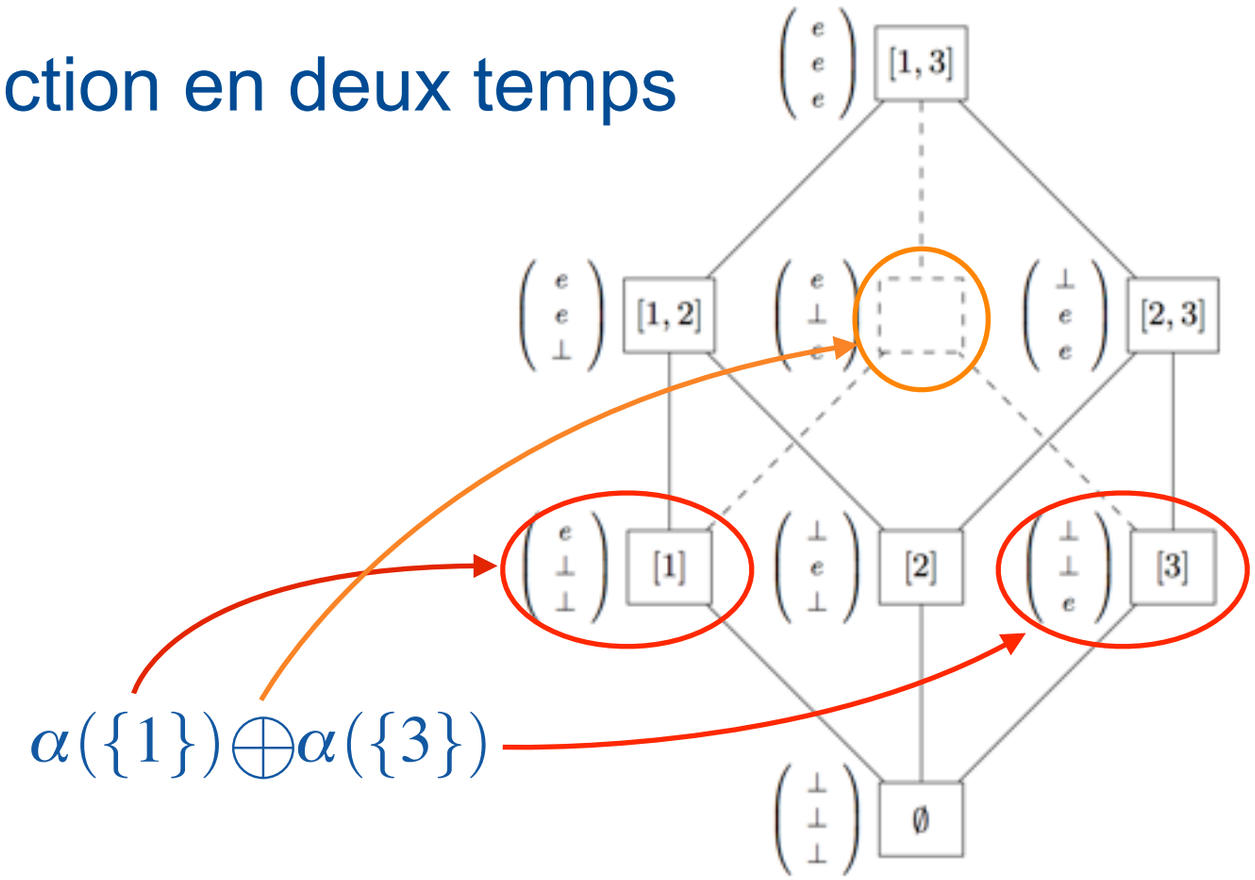
Abstraction en deux temps



Abstraction revisitée

$$\alpha(\{1,3\}) = [1,3] \longrightarrow \overline{\alpha}\left(\begin{pmatrix} e \\ \bot \\ e \end{pmatrix}\right) = \begin{pmatrix} e \\ e \\ e \end{pmatrix}$$

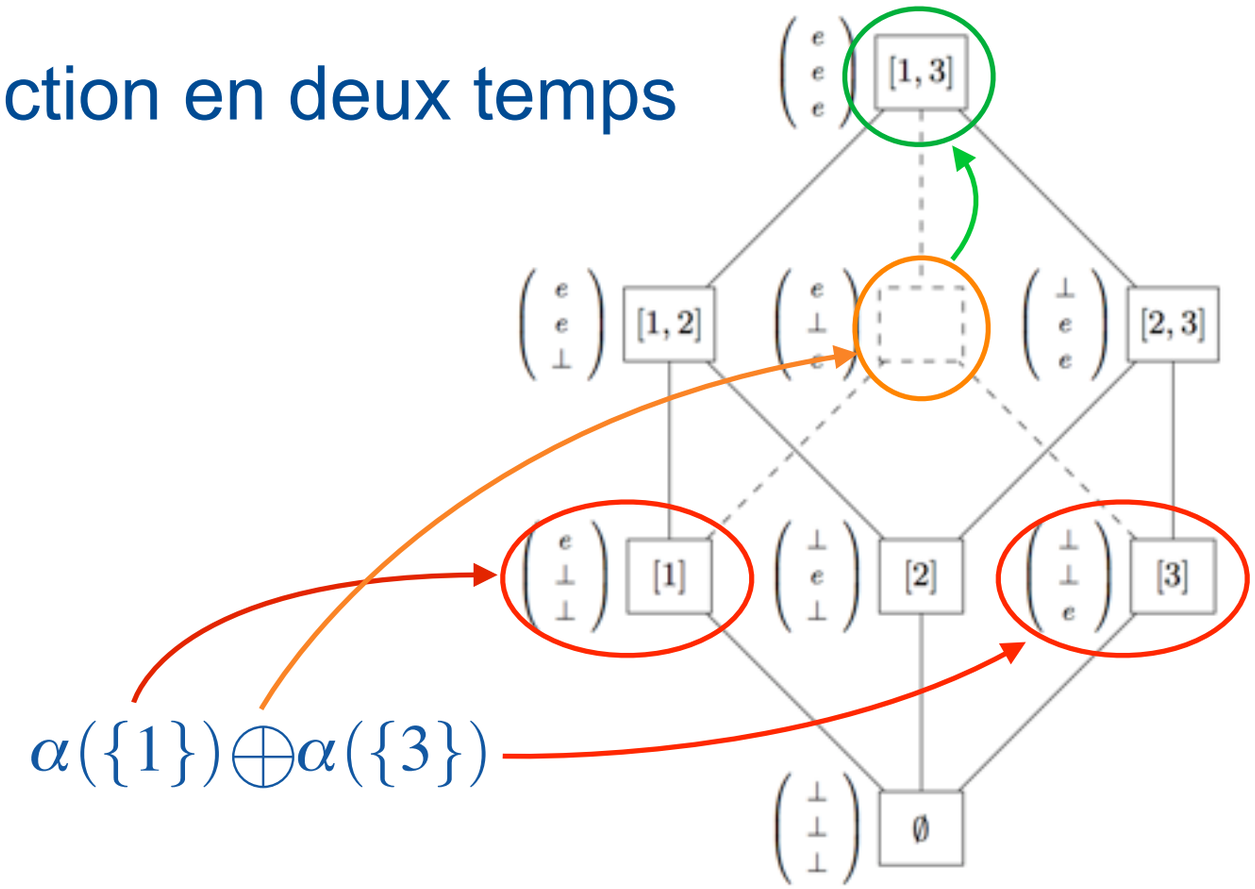
Abstraction en deux temps



Abstraction revisitée

$$\alpha(\{1,3\}) = [1,3] \rightarrow \overline{\alpha}\left(\begin{pmatrix} e \\ \bot \\ e \end{pmatrix}\right) = \begin{pmatrix} e \\ e \\ e \end{pmatrix}$$

Abstraction en deux temps

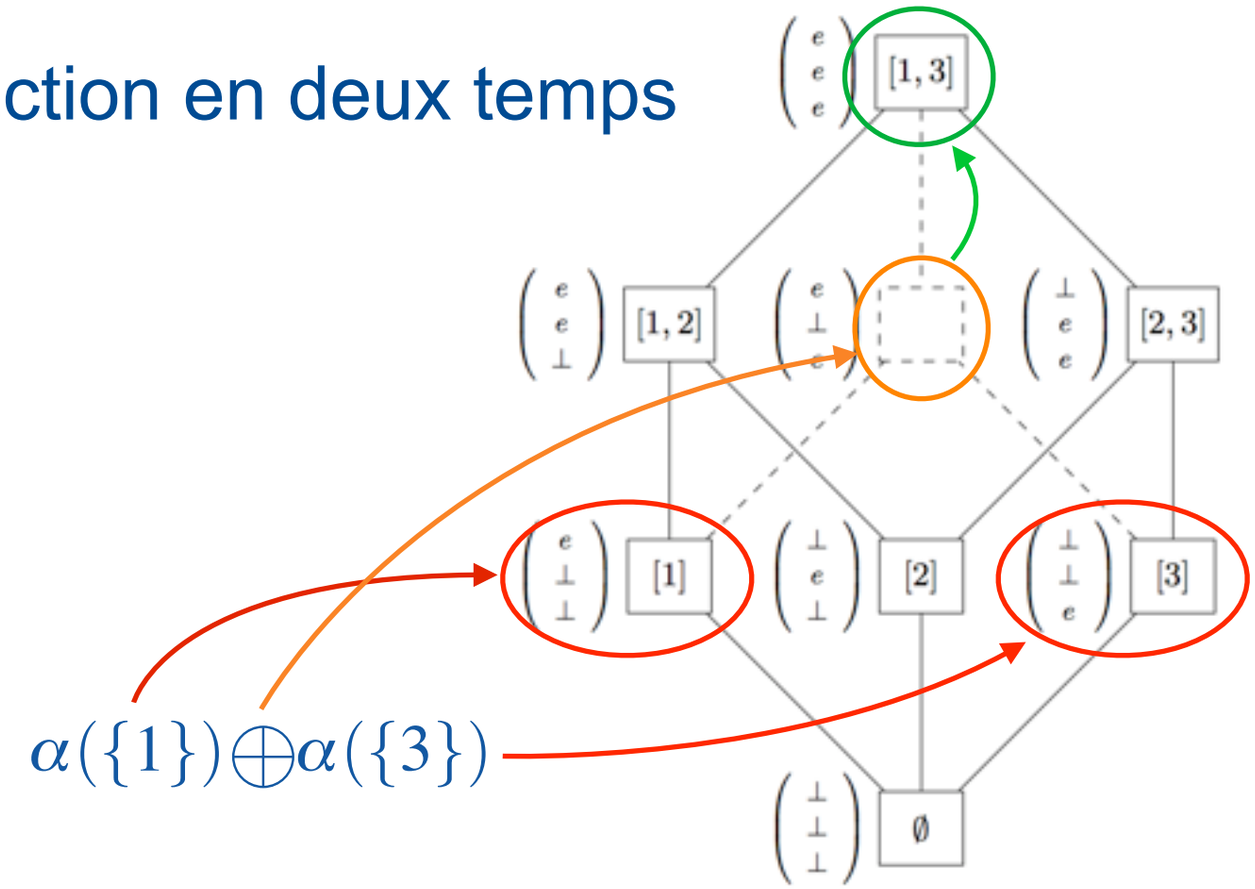


$\overline{\alpha_1}$ p

Abstraction revisitée

$$\alpha(\{1,3\}) = [1,3] \rightarrow \overline{\alpha}\left(\begin{pmatrix} e \\ \bot \\ e \end{pmatrix}\right) = \begin{pmatrix} e \\ e \\ e \end{pmatrix}$$

Abstraction en deux temps



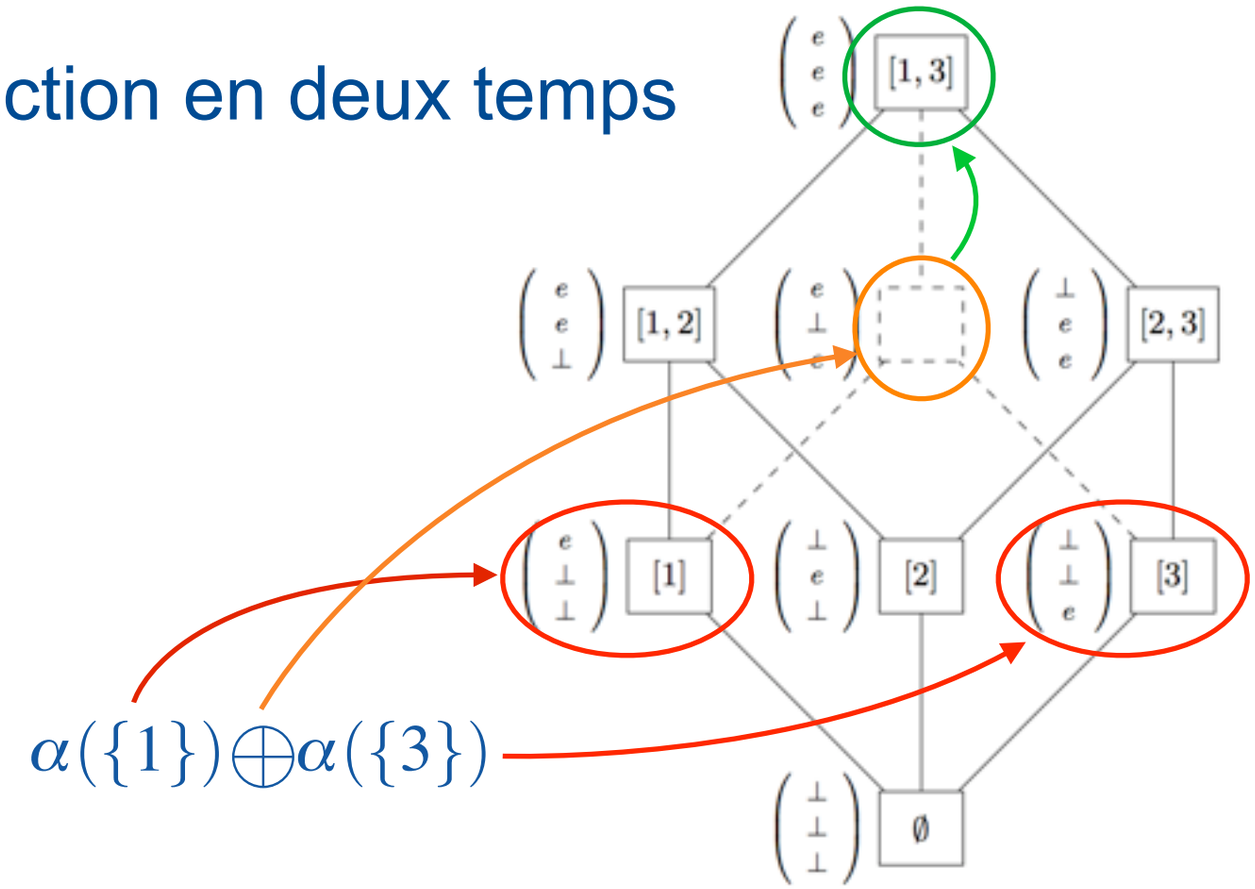
$\overline{\alpha_1}$ p

concrétisation

Abstraction revisitée

$$\alpha(\{1,3\}) = [1,3] \rightarrow \overline{\alpha}\left(\begin{pmatrix} e \\ \bot \\ e \end{pmatrix}\right) = \begin{pmatrix} e \\ e \\ e \end{pmatrix}$$

Abstraction en deux temps



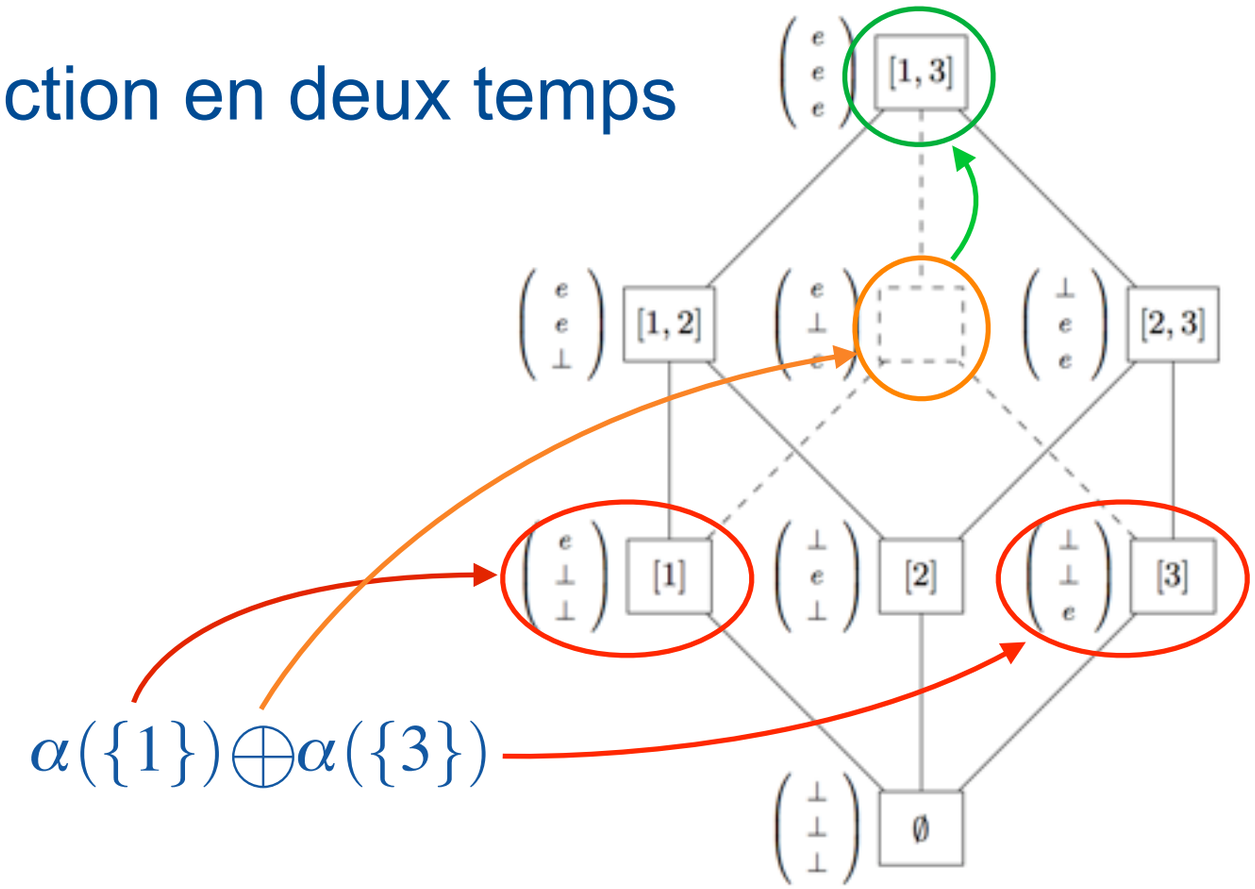
$\overline{\alpha_1}$ p ι

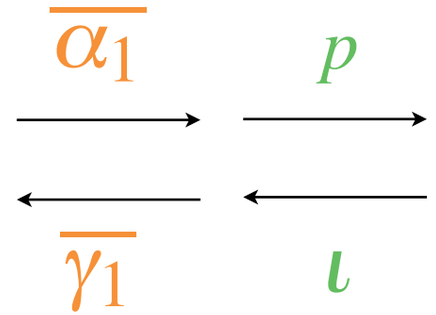
concrétisation

Abstraction revisitée

$$\alpha(\{1,3\}) = [1,3] \rightarrow \overline{\alpha}\left(\begin{pmatrix} e \\ \bot \\ e \end{pmatrix}\right) = \begin{pmatrix} e \\ e \\ e \end{pmatrix}$$

Abstraction en deux temps



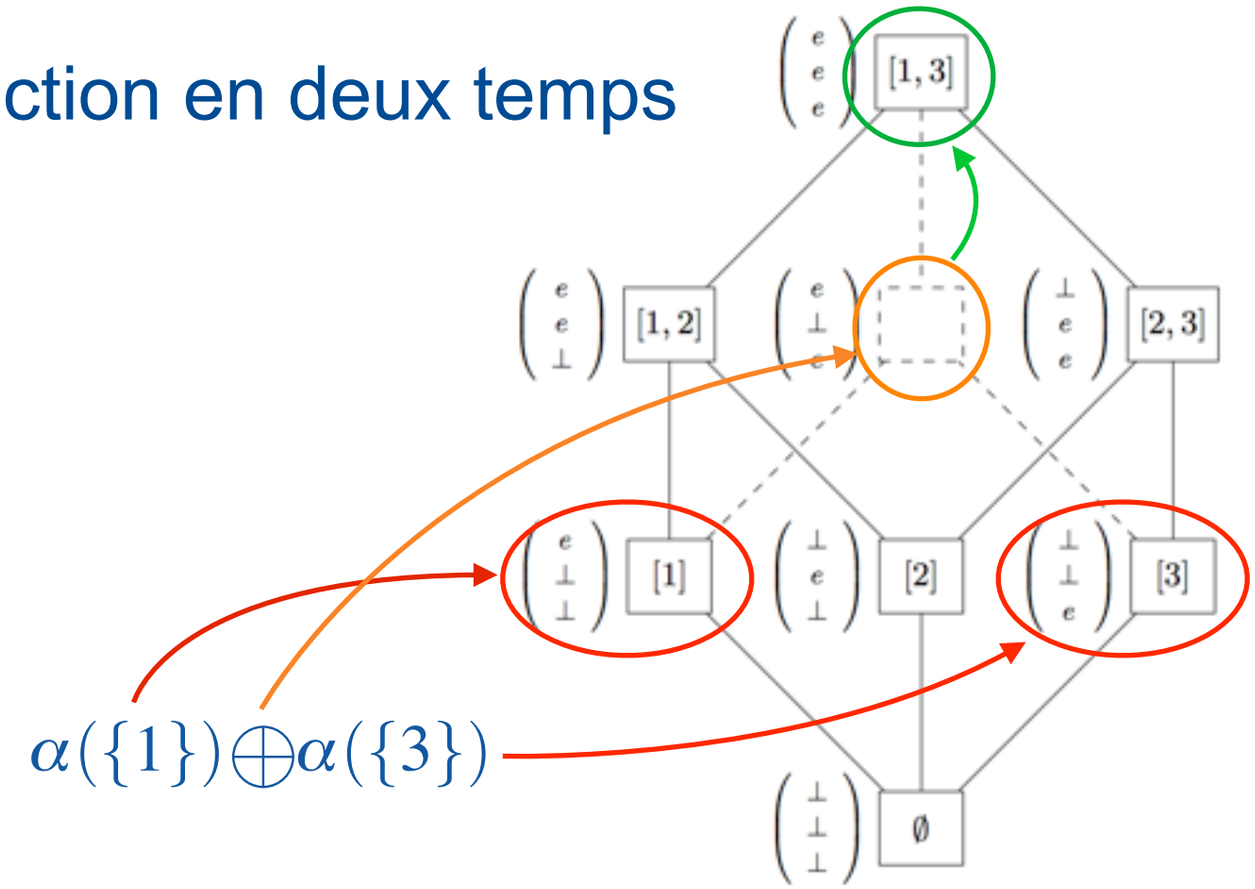


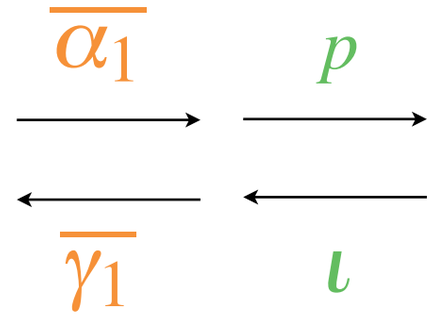
concrétisation

Abstraction revisitée

$$\alpha(\{1,3\}) = [1,3] \rightarrow \overline{\alpha}\left(\begin{pmatrix} e \\ \bot \\ e \end{pmatrix}\right) = \begin{pmatrix} e \\ e \\ e \end{pmatrix}$$

Abstraction en deux temps





concrétisation

Abstraction et concrétisation forment une paire résiduée

En résumé

On a obtenu un cadre qui permet

- de définir une sémantique quantitative sous forme d'opérateur linéaire
- de tenir compte de deux ordres : sur les états et sur les coûts
- de retrouver les connexions de Galois
- de définir une sémantique abstraite
  - correcte par construction
  - la plus précise
- de calculer une approximation des coûts globaux et *long-run*

Études de cas

Analyse de défauts de cache

- langage de bytecode (cf. analyse statique certifiée)
- prise en compte du contenu du cache dans la sémantique
- coûts : compter les défauts de cache
- abstraction : par partition
  - insensible au contexte
  - ne retient qu'une partie du cache

Analyse de consommation d'énergie

- langage impératif avec tableaux
- plusieurs modes d'énergie
- la consommation induite par une opération dépend du mode
- interfaçage avec une analyse existante (domaines polyédriques)

Bilan

Publications

- **Long-run cost analysis by approximation of linear operators over dioids.** Mathematical Structures in Computer Science, 2010. Avec Thomas Jensen, Arnaud Jobin et Pascal Sotin.
- **Quantitative Static Analysis over semirings: analysing cache behaviour for Java Card**. *QAPL 2006*. Avec Thomas Jensen et Pascal Sotin.
- **Injecting Abstract Interpretations into Linear Cost Models.** *QAPL 2010*. Avec Arnaud Jobin.

Développement de prototype

- utilise la librairie max-plus de Scilab

Conclusion

Vérification dans le modèle polyédrique

- utilisation d'assistants de preuve
- technique de preuve spécifique

Analyses statiques certifiées

- parti d'une approche pragmatique : montrer la faisabilité
- cadre générique : interprétation abstraite « simplifiée »
- domaine maintenant mature

Analyses statiques quantitatives

- plus prospectif
- modèle général et original
  - structures mathématiques inhabituelles
  - réconcilier avec le cadre standard

Perspectives

Analyses certifiées

- vers la certification d'analyseurs à la Astrée
  - code C (restreint), flottants...
- connexions de Galois : optimalité
- aspects probabilistes dans la vérification

Aspects quantitatifs

- marier notre cadre avec des résultats plus classiques
  - analyse du nombre d'itérations
- explorer les possibilités du modèle
  - dioïdes de fonctions
- aspects probabilistes
- quantifier la précision

Questions ?