Un algorithme d'élimination de chemins infaisables et sa formalisation

Romain Aïssat
aissat@lri.fr
LRI, Univ. Paris-Sud, CNRS, Université Paris-Saclay
91405 Orsay, France

Réunion du groupe MTV2, Châtenay-Malabry, le 09/12/2015

Sommaire

Sommaire

Contexte

Sujet de thèse

- *Méthodes de test aléatoire pour des programmes C.*

Génération de cas de test

- boîte blanche : chemins complets du graphe de flot de contrôle (CFG),
- guidée par un critère, ex. : toutes ses instructions, tous les chemins de longueur k,
- aléatoire : uniforme sur les chemins.

Pourquoi aléatoire ?

- chaque chemin a la même chance d'être choisi,
- applicable à de très grands graphes (*Rukia*).

Problème : très improbable de tirer des chemins faisables pour de "vrais" programmes.

Aujourd'hui : comment éliminer des chemins infaisables dans le CFG ?

Détection de chemins infaisables

En pratique : réel problème pour le test de "vrais" programmes.

Idée

- élimination de chemins infaisables dans le CFG
- $\rightarrow$ nouveau CFG contenant moins de chemins infaisables, obtenu par transformation du CFG initial,
- trois opérateurs de transformation,
- tirage des chemins dans le nouveau CFG.

Détection des chemins infaisables

- indécidable en général,
- solution : heuristiques, ie. compromis précision vs efficacité.

Sommaire

Principes de l'algorithme

Algorithme de transformation de graphes

- basé sur l'exécution symbolique de tous les chemins.

Trois opérateurs de transformation

- exécution symbolique d'une transition,
- détection d'une subsomption,
- abstraction d'une configuration.

Nécessite : conservation de **tous** les chemins faisables.

Exécution symbolique

Exécution symbolique de tous les chemins

- permet de sur-approximer l'ensemble des chemins d'exécution,
- en conjonction avec solveurs de contraintes : peut être utilisée pour éliminer des chemins infaisables.

Langage de départ

- impératif, à la *C*,
- variables entières, tableaux,
- pas d'appel de fonctions.

Exécution symbolique :

- 3 types d'étiquettes
  - gardes : Assume *bexp*
  - affectations : Assign *var aexp*
  - Skip,
- produit des configurations : état symbolique $\times$ prédicat de cheminement.

Problème : boucles : peuvent être déroulées infiniment $\rightarrow$ arbre d'**ES** infini.

D'un arbre infini à un graphe fini : subsomption

ES classique

- en présence de boucles : dépliage des boucles $\rightarrow$ arbres d'ES potentiellement infinis,
- peut être résolu par la détection de subsomptions et l'abstraction d'états symboliques.

Subsomption : notée "$\sqsubseteq$"

- informellement : $c_2 \sqsubseteq c_1$ ssi c_2 est un cas particulier de c_1,
- formellement : $c_2 \sqsubseteq c_1$ ssi états représentés par $c_2 \subseteq$ états représentés par c_1,
- intérêt : pas besoin de visiter les successeurs d'une configuration subsumée,
- nota bene : subsomption $\leftrightarrow$ approximation,
- on se limite à des subsomptions entre deux configurations à un même point de contrôle, sur un même chemin (*descendant* $\sqsubseteq$ *ancêtre*).

Propriété de la subsomption : $c_2 \sqsubseteq c_1 \rightarrow$ inclusion des chemins faisables.

ES avec subsomptions : parcours en profondeur + détection de subsomptions.

Forcer les subsomptions : abstraction

Problème

- en général : pas toujours possible d'établir une subsomption,
- abstraction de configuration $\rightarrow$ forcer des subsomptions.

Abstraction

- généralisation de l'ensemble des états représentés par la configuration,
- affaiblissement de son prédicat de cheminement.

Subsomption en pratique : lorsqu'un chemin cyclique $\ldots \cdot c \cdot \ldots \cdot c'$ est produit, où c et c' désignent l'entrée d'une même boucle

- tant que $c' \not\sqsubseteq c$:
  - abstraction de c,
  - propagation dans le sous-arbre de racine c : abstraction de c',

Détection de subsomptions

- calcul d'un point fixe,
- synthèse d'un invariant ("local" et faible) de la boucle .

Exécution symbolique avec détection de subsomption

Exécution symbolique de tous les chemins :

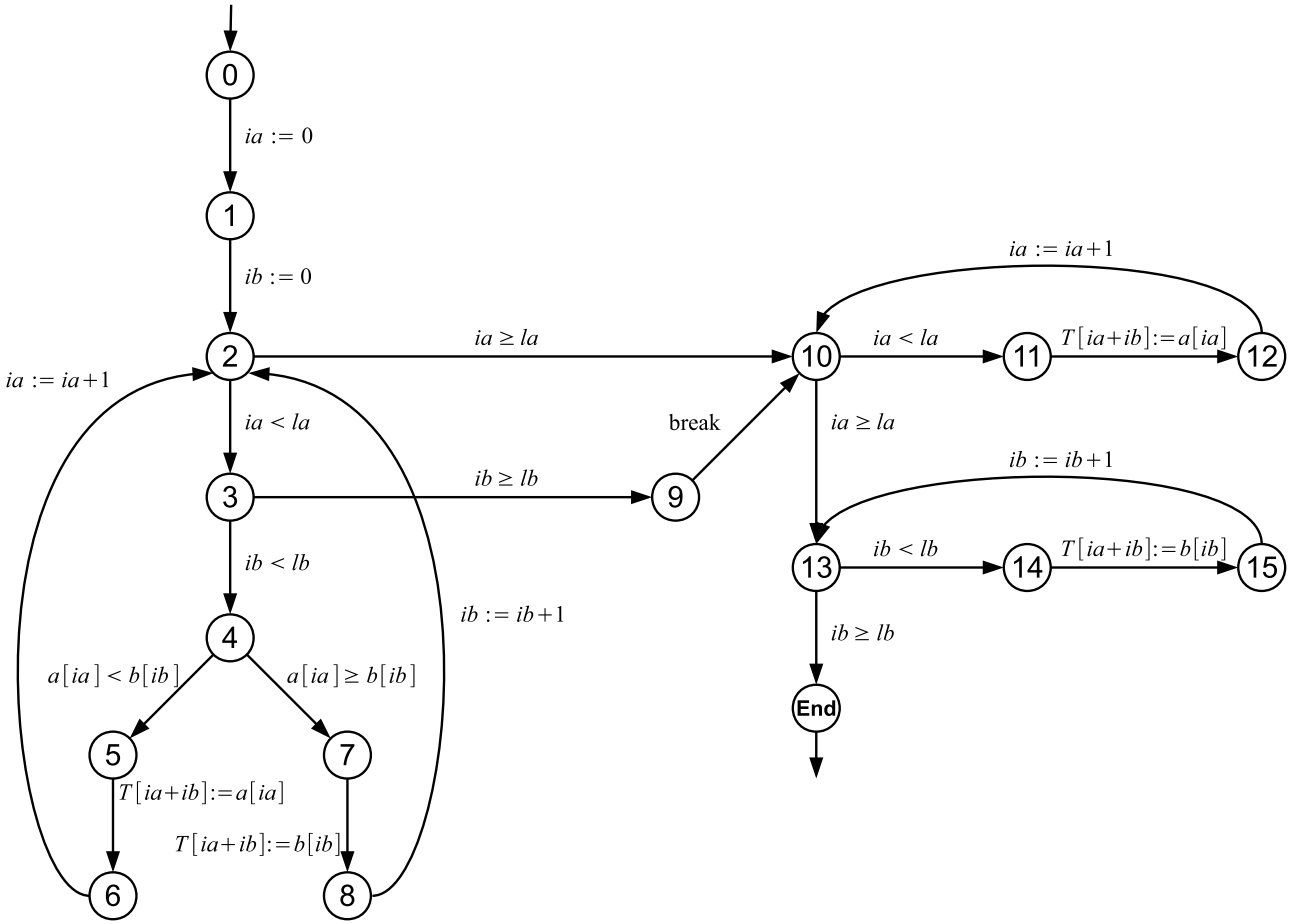
- sur-approximation de l'ensemble des chemins d'exécution,
- sous la forme d'un arbre potentiellement infini.

Détection de subsomptions :

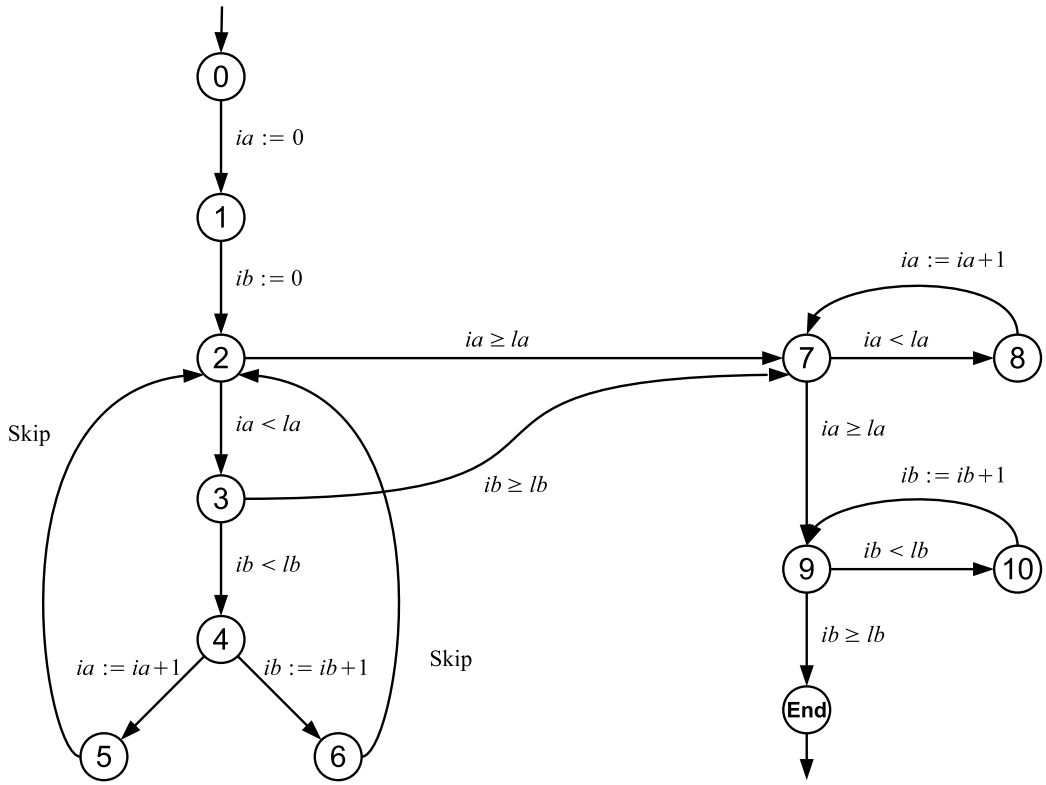
- force l'arrêt du dépliage des boucles,
- arbre potentiellement infini $\rightarrow$ graphe fini.

Sommaire

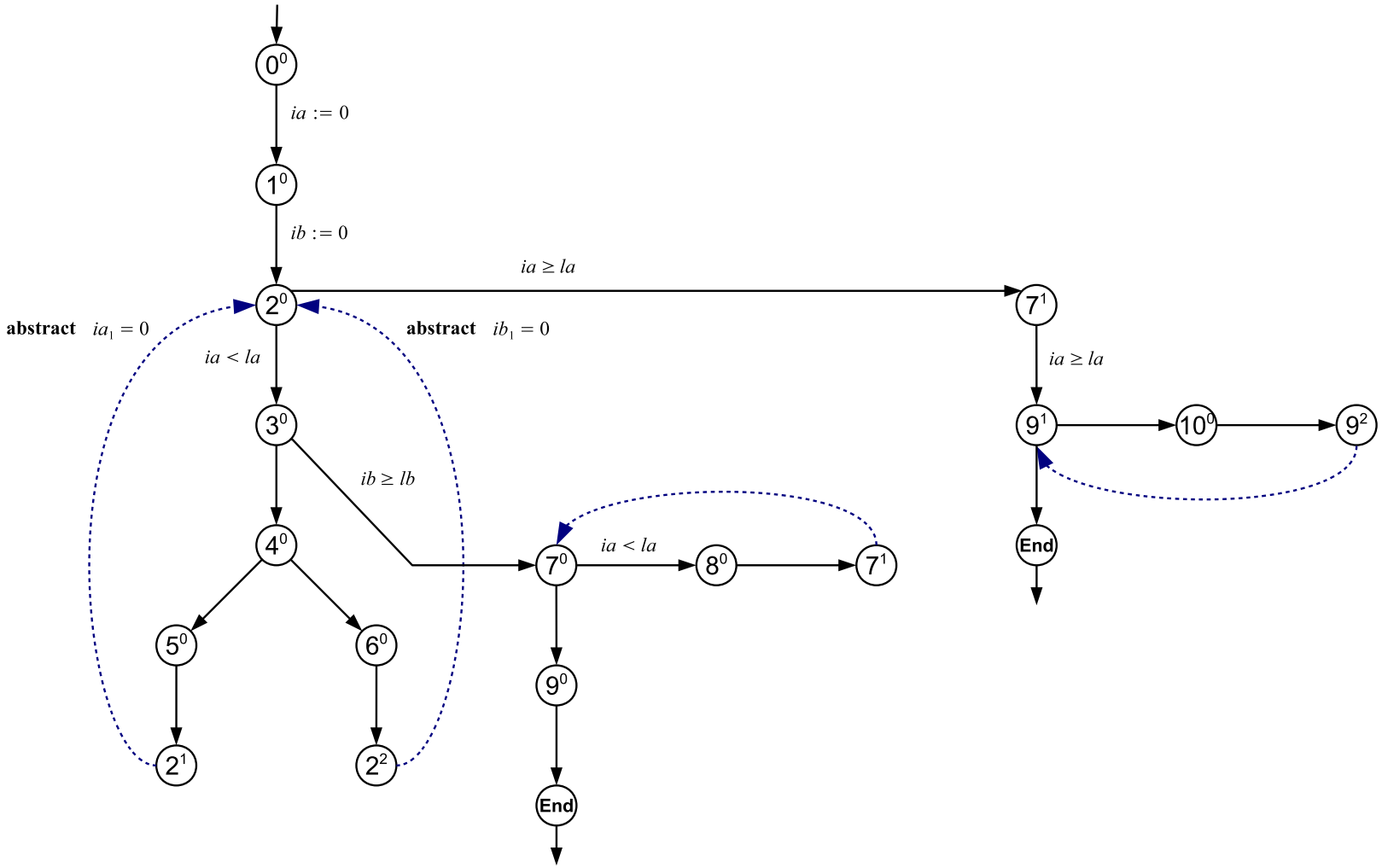
Exemple : fusion de tableaux triés



Exemple : fusion de tableaux triés



Exemple : fusion de tableaux triés



Refuser des subsomptions

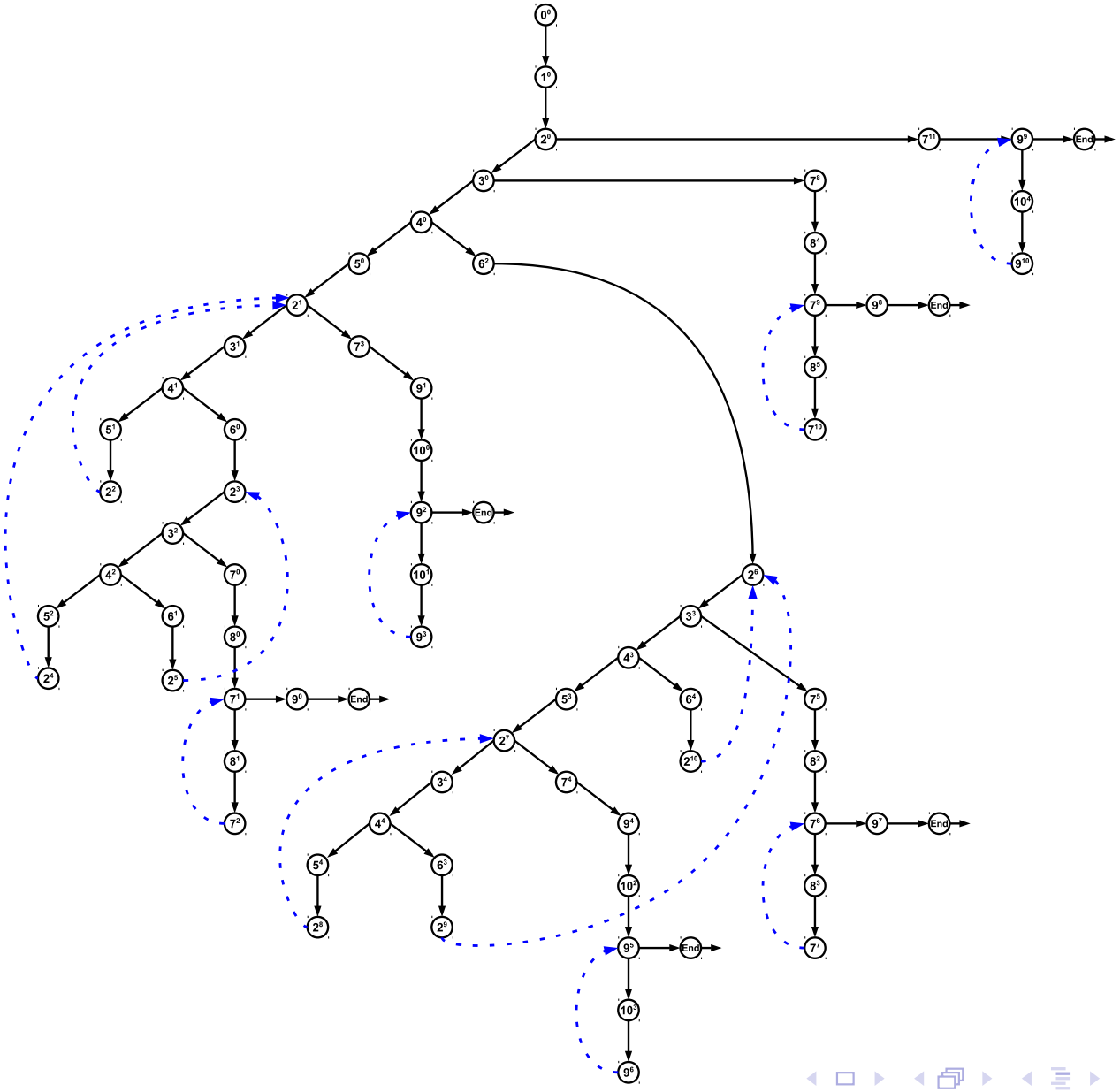
Subsomption : introduction potentielle de chemins infaisables dans le nouveau CFG.

Idéalement : $c_2 \sqsubseteq c_1$ ssi chemins faisables issus de c_1 $=$ chemins faisables issus de c_2.

Critère d'acceptation : ensembles des chemins faisables issus de c_1 et c_2 semblables jusqu'à une "certaine profondeur".

Fusion de tableaux triés : au maximum un dépliage par boucle.

Détection de subsomptions : exemple



Sommaire

Modélisation

Critère de correction : "**tous** les chemins faisables sont conservés".

Modélisation

- sous forme de dépliages/transformations de graphes,
- trois opérations "élémentaires" :
  - exécution symbolique,
  - détection de subsomption,
  - abstraction de configurations.

Théorème (informel) : les trois opérations conservent tous les chemins faisables.

Heuristiques

- dictent l'agencement des trois opérations élémentaires,
- pas d'influence sur la conservation des chemins faisables.

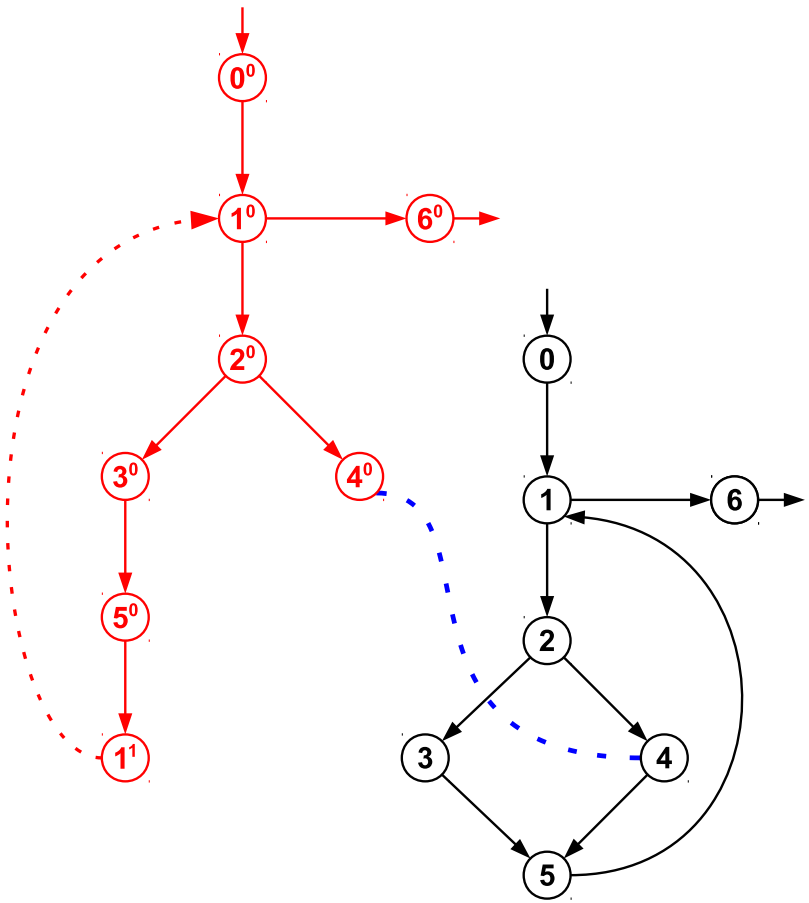
Systèmes de Transitions Etiquetés rouges-noirs

Preuve de correction

- partie explorée insuffisante : ensemble de préfixes des chemins faisables,
- $\rightarrow$ raisonner aussi sur la partie non-explorée.

STE rouge-noir : triplet

- partie noire : CFG initial,
- partie rouge : dépliage fini de la partie noire,
- relation de subsomption : entre sommets de la partie rouge.



Preuve de correction

Frontière rouge-noire : sommets rouges à partir desquels il existe un arc non-emprunté faisable.

Chemins rouges-noirs : ensemble des chemins rouges $\cup$ ensemble des chemins noirs possédant un préfixe rouge menant à la frontière.

Théorème de correction

Soit c la configuration en un sommet rouge (v, i), l'ensemble des chemins faisables noirs issus de v en partant de c est inclus dans l'ensemble des chemins rouges-noirs issus de (v, i).

Preuve

- par induction sur les trois opérateurs de transformation,
- en remarquant que :
  1. tout chemin de la partie rouge après ajout d'une subsomption peut se décomposer en un nombre fini de chemins de la partie rouge avant ajout,
  2. étant donnés un chemin p reliant c à c', on n'a pas $c' = \mathbf{ES}\ c\ p$ mais $\mathbf{ES}\ c\ p \sqsubseteq c'$.

Sommaire

Conclusion

Formalisation :

- assistant de preuve *Isabelle/HOL*,
- concepts :
  - expressions arithmétiques et booléennes (deep embedding),
  - états, états symboliques, configurations,
  - exécution symbolique, subsomption, abstraction,
  - graphes (munis d'une relation de subsomption), chemins, sous-chemins,
  - STE rouges-noirs, frontière, chemins rouges-noirs.
- 10k+ "loc",
- preuve de correction seule : 3k+ loc.

Problème commun à de nombreuses disciplines

- model-checking,
- estimation du pire cas de temps d'exécution,
- analyse statique,
- etc.

Perspectives :

- expérimentations : autres formes de subsomption, interpolation par WP,
- implémentation : en cours, intégration à *Frama-C*.

Merci !

Extension d'un STE rouge-noir par ES d'une transition

```
319
320 abbreviation se_extends ::
321   "('a,'b,'c) pre_RB_scheme ⇒
322    (('a × nat)) rgraph      ⇒
323    ('a × nat) arc           ⇒
324    'b conf                  ⇒
325    ('a,'b,'c) pre_RB_scheme ⇒ bool"
326 where
327   "se_extends prb rg' ra pc' prb' ≡
328      ui_arc ra ∈ arcs (black prb)
329    ∧ src ra ∉ subsumees (subs prb)
330    ∧ extends (red prb) ra rg'
331    ∧ se  (confs prb (src ra)) (labelling (black prb) (ui_arc ra)) pc'
332    ∧ sat pc'
333    ∧ prb' = ⦇ red       = rg',
334               black     = black prb,
335               subs      = subs prb,
336               init_conf = init_conf prb,
337               confs     = (confs prb) (tgt ra := pc'),
338               ...       = more prb ⦈"
339
```