Vérification de programmes

Representations symboliques d'ensembles infinis d'états

$\{\phi\}$ **P** $\{\psi\}$

Entiers	Arithmetiques de premier ordre
Vecteurs	Logiques, machines à compteurs
Arbres	Automates d'arbre etendus
Listes	Shape graphs
Graphes	Logique de séparation

Vérification de programmes

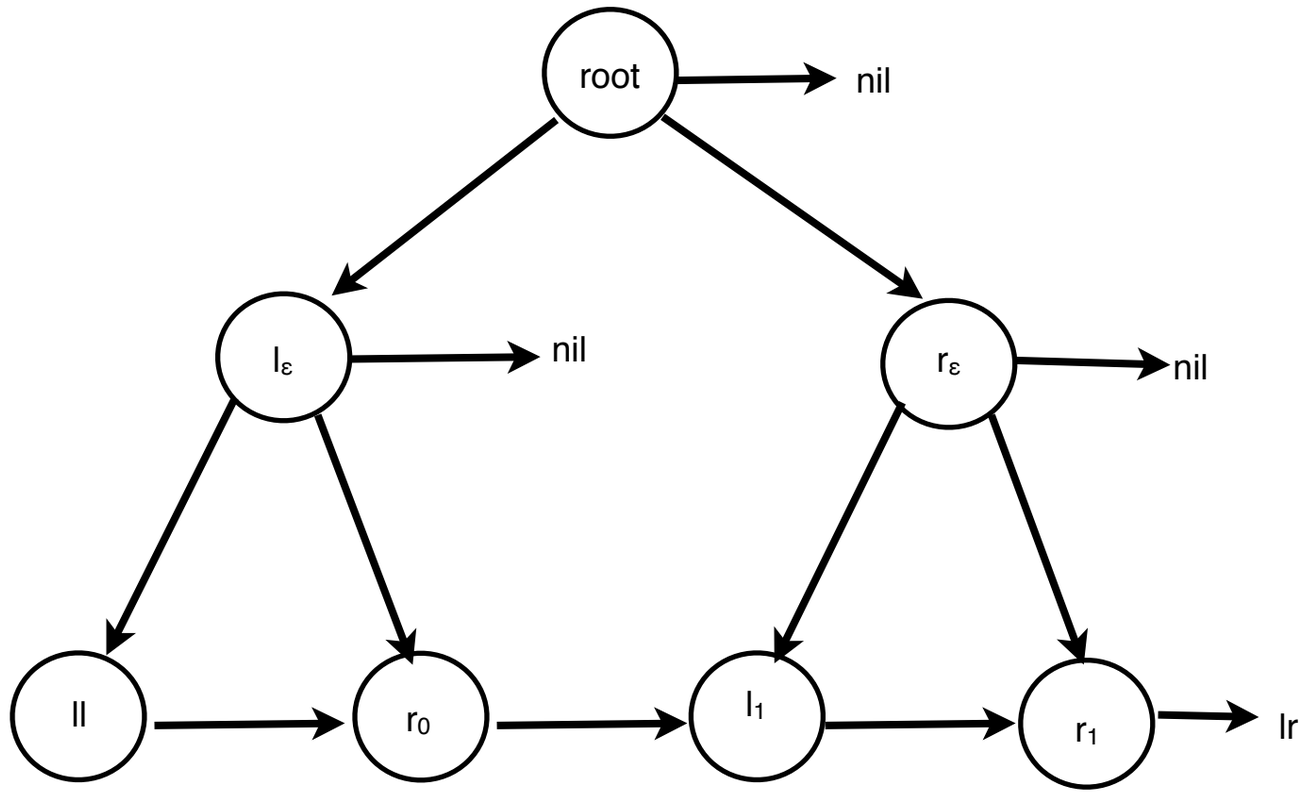
Sémantique du programme

$\{\phi\}$ **P** $\{\psi\}$

Entiers	Machines à compteurs
Vecteurs	Transformateurs de formules, m.à.c.
Arbres	Transformateurs d'automates, m.à.c.
Listes	Machines à compteurs
Graphes	Transformateurs de formules

Structures de graphe récursives

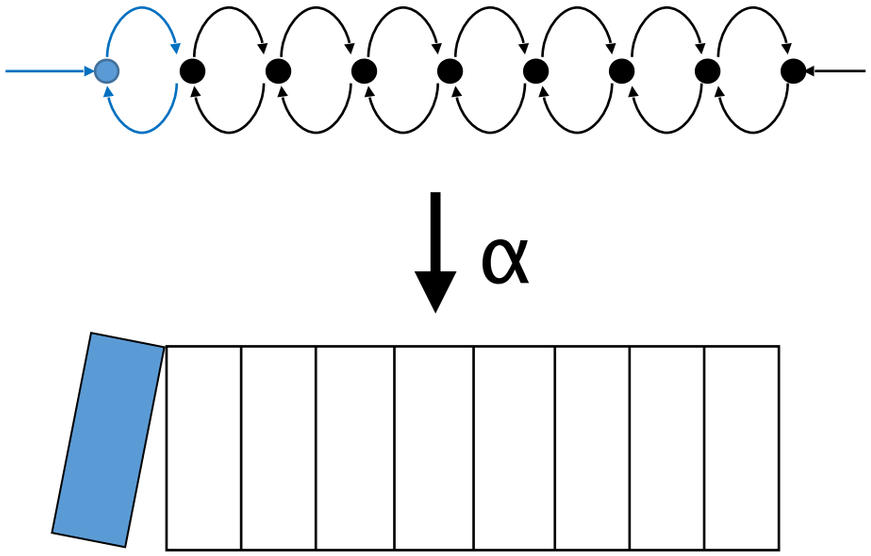
$$\mathrm{tll}(\mathrm{root},\mathrm{ll},\mathrm{lr}) \equiv \mathrm{root}\rightarrow(\mathrm{nil},\mathrm{nil},\mathrm{lr}) \wedge \mathrm{root}=\mathrm{ll}$$
$$\vee\ \exists \mathrm{l},\mathrm{r},\mathrm{z}\ .\ \mathrm{root}\rightarrow(\mathrm{l},\mathrm{r},\mathrm{nil}) * \mathrm{tll}(\mathrm{l},\mathrm{ll},\mathrm{z}) * \mathrm{tll}(\mathrm{r},\mathrm{z},\mathrm{lr})$$



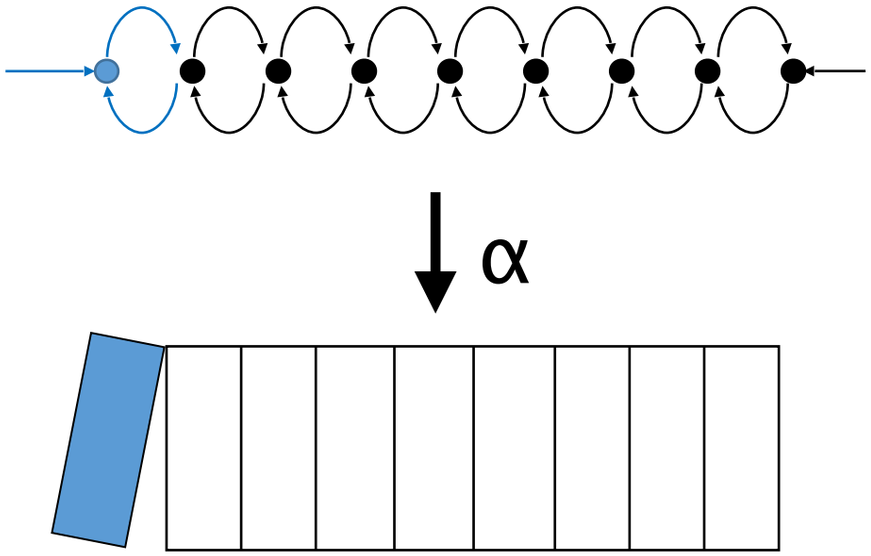
Fragment inductif avec problème d'implication décidable:

- reduction au problème de satisfiabilité dans MSO avec largeur arborescente bornée
- borne inférieure EXPTIME-dûr
- sous-fragment "local" EXPTIME-complet

Relations de raffinement pour ADT



Relations de raffinement pour ADT



∀x:Element ∀s:Stack . pop(push(x,s))

{φ(hd) ∧ α(M,hd)} hd = remove(insert(x,hd)) {φ(hd) ∧ α(M,hd)}

Preuves par descente infinie et SMT

$$acls(x,y) \equiv emp \wedge x=y \vee x \neq y \wedge \exists z \,.\, x \rightarrow z * acls(x,z)$$
$$lseg(x,y) \equiv emp \wedge x=y \vee \exists u \,.\, x \rightarrow u * lseg(u,y)$$

Preuves par descente infinie et SMT

$acls(x,y) \models lseg(x,y)$

$acls(x,y) \equiv emp \wedge x=y \vee x \neq y \wedge \exists z \,.\, x \rightarrow z * acls(x,z)$

$lseg(x,y) \equiv emp \wedge x=y \vee \exists u \,.\, x \rightarrow u * lseg(u,y)$

Preuves par descente infinie et SMT

$$\frac{x \neq y \wedge \exists z \,.\, x \rightarrow z * \mathrm{acls}(x,z) \vDash \exists u \,.\, x \rightarrow u * \mathrm{lseg}(u,y)}{\mathrm{acls}(x,y) \vDash \mathrm{lseg}(x,y)}$$

$$\mathrm{acls}(x,y) \equiv \mathrm{emp} \wedge x = y \vee x \neq y \wedge \exists z \,.\, x \rightarrow z * \mathrm{acls}(x,z)$$
$$\mathrm{lseg}(x,y) \equiv \mathrm{emp} \wedge x = y \vee \exists u \,.\, x \rightarrow u * \mathrm{lseg}(u,y)$$

Preuves par descente infinie et SMT

$$\frac{\mathsf{acls}(x,z) \vDash \mathsf{lseg}(z,y)}{x{\neq}y \wedge \exists z\,.\, x{\rightarrow}z * \mathsf{acls}(x,z) \vDash \exists u\,.\, x{\rightarrow}u * \mathsf{lseg}(u,y)}$$

$x{\neq}y \wedge x{\rightarrow}z \vDash \exists u\,.\, x{\rightarrow}u$
par substitution $u \leftarrow z$

$$\mathsf{acls}(x,y) \vDash \mathsf{lseg}(x,y)$$

$$\mathsf{acls}(x,y) \equiv \mathsf{emp} \wedge x{=}y \vee x{\neq}y \wedge \exists z\,.\, x{\rightarrow}z * \mathsf{acls}(x,z)$$
$$\mathsf{lseg}(x,y) \equiv \mathsf{emp} \wedge x{=}y \vee \exists u\,.\, x{\rightarrow}u * \mathsf{lseg}(u,y)$$

Preuves par descente infinie et SMT

$$\dfrac{\dfrac{}{\mathsf{acls}(x,z) \vDash \mathsf{lseg}(z,y)}}{x{\neq}y \wedge \exists z\,.\, x{\rightarrow}z * \mathsf{acls}(x,z) \vDash \exists u\,.\, x{\rightarrow}u * \mathsf{lseg}(u,y)}\quad x{\neq}y \wedge x{\rightarrow}z \vDash \exists u\,.\, x{\rightarrow}u \text{ par substitution } u \leftarrow z$$

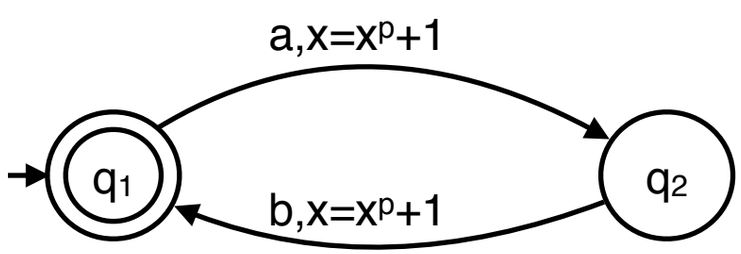
$$\mathsf{acls}(x,y) \vDash \mathsf{lseg}(x,y)$$

$$\mathsf{acls}(x,y) \equiv \mathsf{emp} \wedge x{=}y \vee x{\neq}y \wedge \exists z\,.\, x{\rightarrow}z * \mathsf{acls}(x,z)$$
$$\mathsf{lseg}(x,y) \equiv \mathsf{emp} \wedge x{=}y \vee \exists u\,.\, x{\rightarrow}u * \mathsf{lseg}(u,y)$$

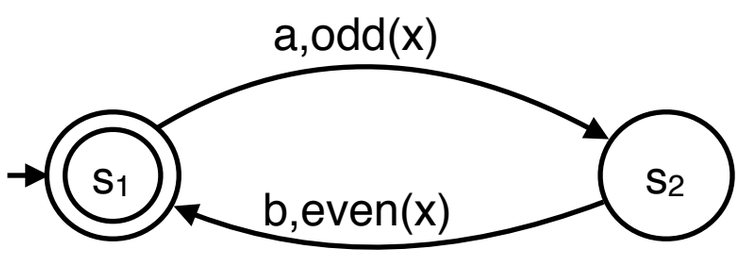
Automates alternants modulo théories

Mot avec données (a,1) (b,2) (a,3) (b,4) (a,5) …

Automates avec données



A



B

L(A) ⊆ L(B) ?

Idee: construire un automate alternant ***A*** tel que L(***A***) = L(A) ∖ L(B) et tester le vide

Défis:

- le problème du vide étant indécidable, on doit définir un semi-algorithme
- raffinement de l'abstraction par contre-exemple (CEGAR) et interpolation de Craig