Automates d'arbres

http://list.mpri.master.univ-paris7.fr/wws/info/cours-1-18

cours-1-18@mpri.master.univ-paris7.fr

Cours: Florent Jacquemard
TD: Ștefan Ciobâcă

INRIA Saclay & LSV (CNRS-ENS Cachan)

florent.jacquemard@inria.fr
ciobaca@lsv.ens-cachan.fr

27 janvier 2010

Automates d'arbres comme ensembles de clauses de Horn

Définition des automates d'arbres comme ensembles de clauses du premier ordre. Langages = modèles de Herbrand.

+ un formalisme uniforme pour représenter plusieurs classes d'automates (automates alternants, à contraintes...)
+ utilisation de techniques et outils de déduction automatiques pour résoudre les problèmes de décision classiques
− complexité
− approche pas constructive

Clauses : syntaxe

- termes de $\mathcal{T}(\Sigma, \mathcal{X})$ sur signature Σ (Σ_n : symboles d'arité n)
- ensemble $\mathcal{P}$ de symboles de prédicats P, Q,... (notation $\mathcal{P}_n$)
  on ne considérera que des prédicats d'arité 1 ou 0.
- litéraux positifs : $P(t)$, noté $+P(t)$
  négatifs : $\neg P(t)$, noté $-P(t)$
- clause : disjonction de litéraux $\pm P_1(t_1) \vee \ldots \vee \pm P_k(t_k)$
  clause vide ($k = 0$), notée $\bot$.
- clause de Horn : au plus un litéral positif
  $-P_1(t_1) \vee \ldots \vee -P_k(t_k) \vee +P(t)$, notée
  $P_1(t_1), \ldots, P_k(t_k) \Rightarrow P(t)$.
- but = clause négative
  $-P_1(t_1) \vee \ldots \vee -P_k(t_k)$, notée $P_1(t_1), \ldots, P_k(t_k) \Rightarrow \bot$.

Clauses : sémantique

- structure $\mathcal{M}$ de domaine D :
  $\mathcal{M} = \langle D, Q^{\mathcal{M}} \subseteq D^n \mid Q \in \mathcal{P}_n, f^{\mathcal{M}} : D^n \to D \mid f \in \Sigma_n \rangle$
  pour $Q \in \mathcal{P}_0$, $Q^{\mathcal{M}} \in \{true, false\}$.
- interprétation $\rho : \mathcal{X} \to D$
- valuation des termes $[\![\_]\!]_{\mathcal{M},\rho} : \mathcal{T}(\Sigma, \mathcal{X}) \to D$
  - $[\![x]\!]_{\mathcal{M},\rho} := \rho(x)$,
  - $[\![f(t_1, \ldots, t_n)]\!]_{\mathcal{M},\rho} := f^{\mathcal{M}}([\![t_1]\!]_{\mathcal{M},\rho}, \ldots, [\![t_1]\!]_{\mathcal{M},\rho})$
- $\mathcal{M}, \rho \models C$ ssi
  - il existe $+P(t) \in C$ tel que $[\![t]\!]_{\mathcal{M},\rho} \in P^{\mathcal{M}}$
    ou bien
  - il existe $-P(t) \in C$ tel que $[\![t]\!]_{\mathcal{M},\rho} \notin P^{\mathcal{M}}$
- $\mathcal{M}$ est un *modèle* d'un ensemble de clauses S (noté $\mathcal{M} \models S$)
  ssi pour toute $C \in S$, pour toute interprétation ρ, $\mathcal{M}, \rho \models C$.
- S est dit *satisfiable* ssi il a un modèle.

Modèles de Herbrand

- une structure de Herbrand $\mathcal{H}$ a pour domaine $\mathcal{T}(\Sigma)$ et fonctions $f^{\mathcal{H}}(t_1, \ldots, t_n) := f(t_1, \ldots, t_n)$ (terme clos avec symbole f en tête).
- $\mathcal{H}$ peut-être définie par l'ensemble des atomes clos $P(t)$ tels que $\mathcal{H} \models P(t)$.

Theorem :

Un ensemble de clauses S est satisfiable ssi il admet un modèle de Herbrand.

Plus petit modèle de Herbrand

Theorem :

Tout ensemble satisfiable de clauses de Horn S admet un plus petit modèle de Herbrand $\mathcal{H}_S$ (pour l'inclusion).

un ensemble de clauses de Horn S définit l'opérateur T_S sur les ensembles d'atomes clos par :

$$\begin{array}{rcl} T_S(L) & = & \left\{ P(t\sigma) \;\middle|\; \begin{array}{l} t\sigma \text{ clos}, P_1(t_1), \ldots, P_n(t_n) \Rightarrow P(t) \in S, \\ P_1(t_1\sigma), \ldots, P_n(t_n\sigma) \in L \end{array} \right\} \\ & \cup & \{\bot\} \text{ si } \begin{array}{l} P_1(t_1), \ldots, P_n(t_n) \Rightarrow \bot \in S, \\ P_1(t_1\sigma), \ldots, P_n(t_n\sigma) \in L \end{array} \end{array}$$

Le plus petit point fixe de T_S est $\bigcup_{n\geq 1} T_S^n(\emptyset)$,

- si il contient $\bot$, S est insatisfiable,
- sinon, il est le plus petit modèle de Herbrand de S.

Langages et automates

Le *langage* d'un ensemble S satisfiable de clauses de Horn pour un prédicat Q est :

$$L(S, Q) = \{t \mid Q(t) \in \mathcal{H}_S\}$$

Soit $\mathcal{A} = (\Sigma, \{q_1, \ldots, q_k\}, F, \Delta)$ un automate d'arbre ascendant.
Soit $\mathcal{P} = \{Q_1, \ldots, Q_k\}$ ensemble de prédicats unaires.
On associe à $\mathcal{A}$ l'ensemble (satisfiable) de clauses de Horn

$$S_{\mathcal{A}} := \left\{ \begin{array}{ll} & Q_1(x_1), \ldots, Q_n(x_n) \Rightarrow Q\big(f(x_1, \ldots, x_n)\big) \\ \mid & f(q_1, \ldots, q_n) \rightarrow q \in \Delta \end{array} \right\}$$

Lemma :

Pour tout état q, $L(\mathcal{A}, q) = L(S_{\mathcal{A}}, Q)$.

Clauses/classes d'automates

clauses d'automate standard ($x_1, \ldots, x_n$ 2 à 2 distinctes)

$$Q_1(x_1), \ldots, Q_n(x_n) \Rightarrow Q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(reg)}$$

ε-transitions

$$Q_1(x) \Rightarrow Q(x) \qquad (\varepsilon)$$

clauses alternantes

$$Q_1(x), \ldots, Q_n(x) \Rightarrow Q(x) \qquad \text{(alt)}$$

clauses bidirectionelles ($x_1, \ldots, x_n$ 2 à 2 distinctes)

$$Q\big(f(x_1, \ldots, x_n)\big) \Rightarrow Q_i(x_i) \qquad \text{(bidi)}$$

Problèmes de décision et satisfiabilité

Soit S un ensemble satisfiable de clauses de Horn et Q un prédicat.

- le terme clos $t \in L(S, Q)$ ssi $S \cup \{Q(t) \Rightarrow \bot\}$ est insatisfiable.
- il existe σ telle que $t\sigma \in L(S, Q)$ ssi $S \cup \{Q(t) \Rightarrow \bot\}$ est insatisfiable.
- $L(S, Q) \neq \emptyset$ ssi $S \cup \{Q(x) \Rightarrow \bot\}$ est insatisfiable.
- $L(S, Q_1) \cap \ldots \cap L(S, Q_p) \neq \emptyset$ ssi $S \cup \{Q_1(x), \ldots, Q_p(x) \Rightarrow \bot\}$ est insatisfiable.

$\Rightarrow$ on s'intéresse aux techniques pour décider la satisfiabilité quand S représente un automate.

Résolution

$$\frac{C \vee +Q(s) \quad D \vee -Q(t)}{C\sigma \vee D\sigma}$$

où σ est l'unificateur le plus général (*mgu*) de s et t.

clauses de Horn :

$$\frac{P_1(s_1), \ldots, P_m(s_m) \Rightarrow Q_1(s) \quad Q_1(t_1), \ldots, Q_n(t_n) \Rightarrow Q(t)}{P_1(s_1\sigma), \ldots, P_m(s_m\sigma), Q_2(t_2\sigma), \ldots, Q_n(t_n\sigma) \Rightarrow Q(t\sigma)}$$

où σ *mgu* de s et t_1.

Theorem : correction, complétude

Un ensemble S de clauses de Horn est insatisfiable ssi on peut dériver $\perp$ à partir de S par résolution.

Terminaison de la résolution

L'application de la règle de résolution à des clauses d'automates (reg) ne termine pas.

$$\frac{P_1(x_1), \ldots, P_m(x_m) \Rightarrow Q_1(g(\overline{x})) \quad Q_1(y_1), \ldots, Q_n(y_n) \Rightarrow Q(f(\overline{y}))}{P_1(x_1), \ldots, P_m(x_m), Q_2(y_2), \ldots, Q_n(y_n) \Rightarrow Q\big(f(g(\overline{x}), y_2, \ldots, y_n)\big)}$$

Stratégies complètes pour la résolution

$$\frac{\begin{array}{cc} C & D \\ P_1(s_1), \ldots, P_m(s_m) \Rightarrow Q_1(s) & Q_1(t_1), \ldots, Q_n(t_n) \Rightarrow Q(t) \end{array}}{P_1(s_1\sigma), \ldots, P_m(s_m\sigma), Q_2(t_2\sigma), \ldots, Q_n(t_n\sigma) \Rightarrow Q(t\sigma)}$$

résolution ordonnée pour $\succ$:

- $Q_1(s)$ maximal pour $\succ$ dans C,
- $Q_1(t_1)$ maximal pour $\succ$ dans D.

résolution ordonnée avec sélection :
fonction de sélection : clause $\mapsto$ sous-ensemble de litéraux négatifs.

- aucun litéral sélectionné dans C,
- $Q_1(s)$ maximal pour $\succ$ dans C,
- $Q_1(t_1)$ sélectionné dans D ou bien
- aucun litéral sélectionné dans D et $Q_1(t)$ maximal dans D.

Complétude de la résolution

Theorem :

La résolution ordonnée avec sélection est complète pour les clauses de Horn : à partir de tout ensemble insatisfiable S, on dérive $\bot$.

Transformation des automates alternants

Proposition :

Etant donné un automate d'arbres alternant $\mathcal{A}$ sur Σ, on peut construire en temps exponentiel un automate d'arbres ascendant déterministe $\mathcal{A}'$ reconnaissant le même langage.

Construction par application de la résolution ordonnée avec sélection, suivant une stratégie appropriée.

Choix d'une stratégie ordonnée avec sélection

- ordre t.q. $P(s) \succ Q(t)$ ssi $s > t$ pour l'ordre $>$ de sous-terme.
- fonction de sélection sel :
  - litéraux négatifs $-Q(t)$ où t n'est pas une variable.

Lemma :

Tout automate d'arbres (ensemble de clauses (reg)) est saturé par résolution ordonnée avec $\succ$.

Transformation des automates alternants

pr.: (proposition automate alternant $\rightarrow$ automate ascendant déterministe).

- on part d'un ensemble $\mathcal{A}$ de clauses de la forme (reg) et (alt).
- on sature par résolution ordonnée (par $\succ$) avec sélection (par sel).
- toutes les clauses produites appartiennent à un type contenant un nombre exp. de clauses
- $\rightarrow$ saturation termine avec $\mathcal{A}''$
- déduction par résolution de $\mathcal{A}'' \cup \{Q(t) \Rightarrow \bot\}$ (pour t clos) n'implique que des clauses de la forme (reg).
- donc pour tout Q, $L(\mathcal{A}'', Q) = L(\mathcal{A}''|_{\mathsf{reg}}, Q)$.

Transformation des automates bidirectionnels alternants

Pour généraliser le résultat précédent (reg + alt $\longrightarrow$ reg)
à reg + alt + bidi $\longrightarrow$ reg, on utilise le même principe avec

- d'autres ordre et fonction de sélection pour définir la stratégie de résolution,
- et une règle supplémentaire de ε-splitting.

Proposition :

Etant donné un automate d'arbres bidirectionnel alternant $\mathcal{A}$ sur Σ, on peut construire en temps exponentiel un automate d'arbres ascendant déterministe $\mathcal{A}'$ reconnaissant le même langage.

ε-splitting

Un ε-bloc $B(x)$ est un ensemble de litéraux négatifs $-P_1(x) \vee \ldots \vee -P_n(x)$.

$$\frac{B(x), Q_1(t_1), \ldots, Q_n(t_n) \Rightarrow Q(t)}{B(x) \Rightarrow q_B \quad q_B, Q_1(t_1), \ldots, Q_n(t_n) \Rightarrow Q(t)}$$

- q_B est un prédicat 0-aire associé de manière unique à $B(x)$,
- $x \notin Q_1(t_1), \ldots, Q_n(t_n), Q(t)$.

Theorem :

La résolution ordonnée avec sélection et ε-splitting est complète pour les clauses de Horn.

Choix d'une autre stratégie ordonnée avec sélection

- ordre t.q. $P(s) \succ Q(t)$ ssi $s > t$ pour un ordre $>$ de sous-terme fixé et $P(s) \succ q_B$ p.t. P et q_B.
- fonction de sélection sel (par ordre de priorité) :
  - litéraux de splitting (q_B).
  - litéraux négatifs $-Q(t)$ où t n'est pas une variable.

Transformation des automates bidirectionnels alternants

Proposition :

Etant donné un automate d'arbres bidirectionnel alternant $\mathcal{A}$ sur Σ, on peut construire en temps exponentiel un automate d'arbres ascendant déterministe $\mathcal{A}'$ reconnaissant le même langage.

pr.: même principe que pour les automates alternants, avec le ε-splitting en plus de la résolution.

Décision appartenance généralisée

Theorem :

L'application de la résolution ordonnée (par $\succ$) avec sélection (par sel) et ε-splitting termine sur l'union d'un ensemble de clauses (reg) et d'une clause but $Q(t) \Rightarrow \bot$.

pr.: La résolution ne produit que des clauses de ces 2 types :

$$P_1(s_1), \ldots, P_m(s_m), q_1, \ldots, q_k \Rightarrow [\, q \,] \qquad \text{(gs)}$$

où $m, k \geq 0$, et $s_1, \ldots, s_m$ sont des sous-termes de t.

$$P_1(y_{i_1}), \ldots, P_k(y_{i_k}), \underline{P'_1(f(y_1, \ldots, y_n))}, \ldots, \underline{P'_m(f(\overline{y}))} \Rightarrow [\, q \,] \quad \text{(gf)}$$

où $k, m \geq 0$, $k + m > 0$, $i_1, \ldots, i_k \leq n$, et $y_1, \ldots, y_n$ distinctes.

Automates d'arbres à contraintes égalitaires

Tests d'égalité entre frères

clauses d'automate standard ($x_1, \ldots, x_n$ 2 à 2 distinctes)

$$Q_1(x_1), \ldots, Q_n(x_n) \Rightarrow Q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(reg)}$$

avec partage de variables dans

$$Q_1(x_1), \ldots, Q_n(x_n) \Rightarrow Q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(brother)}$$

on exprime des égalité entre les sous termes frères.

exemple :

$$\begin{array}{rcl} & \Rightarrow & Q(a), \\ Q(x_1), Q(x_2) & \Rightarrow & Q\big(f(x_1, x_2)\big), \\ Q(x), Q(x) & \Rightarrow & Q_{\mathsf{f}}\big(f(x, x)\big) \end{array}$$

Tests d'égalité entre frères

expressivité $\supsetneq$ automates d'arbres ascendants.

Theorem :

Le vide des automates à tests d'égalité entre frères est EXPTIME-complet.

Automates de Bogaert-Tison

Tests $=$ et $\neq$ entre frères (chapitre 4 TATA).

- déterminizables en temps exponentiel
- clotures Booléennes
- vide PTIME pour déterministes
- vide EXPTIME-complet pour non-déterministes

Tests d'égalité arbitraires

clauses avec égalité

$$Q_1(x_1), \ldots, Q_n(x_n), u_1 = v_1, \ldots, u_k = v_k \Rightarrow Q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(test)}$$

où $k \geq 0$, $u_1, v_1, \ldots, u_k, v_k \in \mathcal{T}\big(\Sigma, \{x_1, \ldots, x_n\}\big)$.

sans restrictions, le vide est indécidable.

Tests d'égalité arbitraires (décidable)

On distingue des prédicats *test*, et on suppose un ordre partiel $\succ$ sur les prédicats tel que pour tout Q test et Q_0 non-test, $Q \succ Q_0$.

$$Q_1(x_1), \ldots, Q_n(x_n), u_1 = v_1, \ldots, u_k = v_k \Rightarrow Q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(test)}$$

où Q prédicat test, et pour tout Q_i qui est test, $Q \succ Q_i$.

$$Q_1(x_1), \ldots, Q_n(x_n) \Rightarrow Q\big(f(x_1, \ldots, x_n)\big) \qquad \text{(reg}')$$

ou bien $Q, Q_1, \ldots, Q_n$ pas test
ou bien Q test et au plus un $Q_i = Q$, les autres sont non-test.

Tests d'égalité arbitraires (décidable)

exemple : listes à bégaiement

$$\begin{array}{rlrrl} \Rightarrow & Q_0(0) & & Q_0(x) & \Rightarrow \quad Q_0(s(x)) \\ \Rightarrow & Q_1([\,]) & Q_0(x), Q_1(y) & \Rightarrow & Q_1(cons(x,y)) \\ & & Q_0(x), Q_2(y) & \Rightarrow & Q_2(cons(x,y)) \end{array}$$

$$Q_0(x), Q_1(y), y = cons(x, y') \Rightarrow Q_2(cons(x, y))$$

Tests d'égalité arbitraires (décidable)

Theorem :

La satisfiabilité d'un ensemble de clauses (test) et (reg') et d'une clause but $Q(t) \Rightarrow \bot$ est décidable.

pr.: Saturation par paramodulation ordonnée avec sélection et ϵ-splitting.

Extension aux langages modulo des théories équationelles, par ajout de clauses $\Rightarrow u = v$.

Automates d'arbres à mémoire

Automates à pile

Extension des automates de mots finis (NFA) avec une mémoire non bornée = pile.

$\mathcal{A} = (\Sigma, \Gamma, Q, Q^{\mathsf{i}}, Q^{\mathsf{f}}, \delta)$.

- Σ : alphabet d'entrée,
- Γ : alphabet de pile.

$$\delta \subseteq \underbrace{Q \times \Sigma \times Q \times \Gamma}_{(push)} \cup \underbrace{Q \times \Sigma \times \Gamma \times Q}_{(pop)} \cup \underbrace{Q \times \Sigma \cup \{\varepsilon\} \times Q}_{(internal)}$$

Example :

$\Sigma = \{a, b\}$, $\Gamma = \{\alpha\}$.

$$\begin{aligned} push : &\quad q^{\mathsf{i}} \xrightarrow{a} q^{\mathsf{i}}, \alpha \\ internal : &\quad q^{\mathsf{i}} \xrightarrow{b} q^{\mathsf{f}} \\ pop : &\quad q^{\mathsf{f}} \xrightarrow{a,\alpha} q^{\mathsf{f}} \end{aligned}$$

reconnaît : $\{a^n b a^n \mid n \geq 0\}$.

Automates à pile (propriétés)

- expressivité $\supsetneq$ NFA
- même expressivité que les grammaires hors-contexte ($N := N_1 N_2$, $N := a$)
- vide décidable
- pas clos par intersection, complément
- la restriction *visibly* : $\Sigma = \Sigma_{push} \uplus \Sigma_{pop} \uplus \Sigma_{int}$ a les clôtures Booléennes.

Lemma :

Le langage des piles accessibles est régulier.

Automates d'arbres à pile

Automates d'arbres à pile [Guessarian 83] :

- Σ : signature d'entrée ; en lecture : termes de $\mathcal{T}(\Sigma)$,
- Γ : alphabet de pile ; mémoire auxiliaire = pile de Γ^*,
- transitions descendantes :

$$\begin{array}{lrcl}
push & q(y) & \to & f\big(q_1(e_1(y)), \ldots, q_n(e_n(y))\big) \\
pop & q(e(y)) & \to & f\big(q_1(y), \ldots, q_n(y)\big) \\
pop & q(\bot) & \to & f\big(q_1(\bot), \ldots, q_n(\bot)\big) \\
int & q(y) & \to & f\big(q_1(y), \ldots, q_n(y)\big) \\
\varepsilon & q(y) & \to & q'(y)
\end{array}$$

- même expressivité que les grammaires d'arbres hors-contexte.

Automates d'arbres à une mémoire

Automates ascendants avec une mémoire auxiliaire contenant un arbre. $\mathcal{A} = (\Sigma, \Gamma, Q, Q^{\mathsf{f}}, \Delta)$.

- Σ : signature d'entrée ; en lecture : termes de $\mathcal{T}(\Sigma)$,
- Γ : signature de pile ; mémoire auxiliaire = terme de $\mathcal{T}(\Gamma)$.
- Δ : transitions de la forme

$$
\begin{array}{lllcl}
push & a & & \to & q(c) \\
push & f\big(q_1(y_1), & q_2(y_2)\big) & \to & q\big(h(y_1, y_2)\big) \\
pop_{11} & f\big(q_1(h(y_{11}, y_{12})), & q_2(y_2)\big) & \to & q(y_{11}) \\
 & f\big(q_1(\bot), & q_2(y_2)\big) & \to & q(\bot) \\
pop_{12} & f\big(q_1(h(y_{11}, y_{12})), & q_2(y_2)\big) & \to & q(y_{12}) \\
 & f\big(q_1(\bot), & q_2(y_2)\big) & \to & q(\bot) \\
 & \vdots & & & \\
int_0 & a & & \to & q(\bot) \\
int_1 & f\big(q_1(y_1), & q_2(y_2)\big) & \to & q(y_1) \\
int_2 & f\big(q_1(y_1), & q_2(y_2)\big) & \to & q(y_2)
\end{array}
$$

Automates d'arbres à une mémoire (propriétés)

- expressivité $\supsetneq$ automates d'arbres ascendants
- généralise les automates (de mots) à pile
- vide décidable
- pas clos par intersection ni complément
- la restriction *visibly* a toutes les clôtures Booléennes.

$$\Sigma = \Sigma_{push} \uplus \Sigma_{pop_{11}} \uplus \Sigma_{pop_{12}} \uplus \Sigma_{pop_{21}} \uplus \Sigma_{pop_{22}} \uplus \Sigma_{int_0} \uplus \Sigma_{int_1} \uplus \Sigma_{int_2}$$

Décision du vide

Theorem :

Le vide est décidable en temps polynomial pour les automates à 1 mémoire.

- $L(\mathcal{A}, q) := \left\{ t \;\middle|\; \exists m \in \mathcal{T}(\Gamma), t \xrightarrow[\Delta]{*} q(m) \right\}$
- $M(\mathcal{A}, q) := \left\{ m \;\middle|\; \exists t \in \mathcal{T}(\Sigma), t \xrightarrow[\Delta]{*} q(m) \right\}$.

Lemma :

Pour tous $\mathcal{A}$, q, $L(\mathcal{A}, q) = \emptyset$ ssi $M(\mathcal{A}, q) = \emptyset$.

Lemma :

Pour tous $\mathcal{A}$, q, $M(\mathcal{A}, q)$ est un langage d'automate bidirectionnel.

Décision du vide (2)

$$
\begin{array}{lrcll}
push & f\big(q_1(y_1), q_2(y_2)\big) & \longrightarrow & q\big(h(y_1, y_2)\big) & \\
 & Q_1(y_1), Q_2(y_2) & \Rightarrow & Q\big(h(y_1, y_2)\big) & (\mathsf{reg}) \\
pop_{11} & f\big(q_1(h(y_{11}, y_{12})), q_2(y_2)\big) & \longrightarrow & q(y_{11}) & \\
 & Q_1\big(h(y_{11}, y_{12})\big), Q_2(y_2) & \Rightarrow & Q(y_{11}) & \\
split: & Q_1\big(h(y_{11}, y_{12})\big), q_{Q_2} & \Rightarrow & Q(y_{11}) & (bidi) \\
 & Q_2(y_2) & \Rightarrow & q_{Q_2} & \\
int_1 & f\big(q_1(y_1), q_2(y_2)\big) & \longrightarrow & q(y_1) & \\
 & Q_1(y_1), Q_2(y_2) & \Rightarrow & Q(y_1) & \\
split: & Q_1(y_1), q_{Q_2} & \Rightarrow & Q(y_1) & (\varepsilon) \\
 & Q_2(y_2) & \Rightarrow & q_{Q_2} &
\end{array}
$$

Automates d'arbres à une mémoire et contraintes

Version étendue avec contraintes $=$ et $\neq$ entre mémoires dans les transitions int.

$$
\begin{array}{lll}
int_1^{=} & f\big(q_1(y_1), q_2(y_2)\big) & \xrightarrow{y_1=y_2} \quad q(y_1) \\
int_1^{\neq} & f\big(q_1(y_1), q_2(y_2)\big) & \xrightarrow{y_1\neq y_2} \quad q(y_1) \\
int_2^{=} & f\big(q_1(y_1), q_2(y_2)\big) & \xrightarrow{y_1=y_2} \quad q(y_2) \\
int_2^{\neq} & f\big(q_1(y_1), q_2(y_2)\big) & \xrightarrow{y_1\neq y_2} \quad q(y_2)
\end{array}
$$

Automates d'arbres à une mémoire et contraintes : exemples

- arbres binaires équilibrés
- powerlists
- arbres (binaires) rouge-noir
  1. chaque noeud est noir ou rouge,
  2. la racine est noire,
  3. toutes les feuilles sont noires,
  4. les deux fils d'un noeud rouge sont noirs,
  5. tous les chemins contiennent le même nombre de noeuds noirs.

Automates d'arbres à une mémoire et contraintes : décision

Theorem :

Le vide est décidable en temps exponentiel pour les automates à 1 mémoire et contraintes.

pr.: Résolution ordonnée avec sélection et ε-splitting.