Arbres

Alphabet $\Sigma = \Sigma_0 \cup \cdots \cup \Sigma_k$

- Σ_i : alphabet fini de symboles de rang i ($\Sigma_i \cap \Sigma_j \neq \emptyset$ possible).
- Un arbre t de rang k est défini par un ensemble (fini) $\mathrm{dom}(t) \subseteq \{1, \ldots, k\}^*$ clos par préfixe (domaine de t) : si $v, vi \in \mathrm{dom}(t)$, alors vi est le i-ème enfant de v.
- Un arbre t étiqueté dans Σ est une fonction $\ell(t) : \mathrm{dom}(t) \to \Sigma$ telle que si v a i enfants, alors $\ell(v) \in \Sigma_i$. On écrit aussi t à la place de $\ell(t)$.
- $T(\Sigma)$: ensemble des arbres étiquetés dans Σ.
- Racine, feuilles, nœuds internes, hauteur, prédécesseur, descendant, sous-arbre. Frontière $\mathrm{fr}(t)$: séquence des étiquettes des feuilles de t, lues de gauche à droite.

Exemples

Arbres binaires : $\Sigma = \Sigma_0 \cup \Sigma_2$. Termes LISP. Arbres des expressions arithmétiques (rang 2). Arbres de codes préfixe.

Automates d'arbre

Automates d'arbre montants (angl. bottom-up NTA - non-deterministic tree automaton)

Un automate d'arbre $\mathcal{A} = \langle Q, \Sigma, \delta, F \rangle$ sur l'alphabet $\Sigma = \cup_{i=0}^{k} \Sigma_i$ est composé d'un ensemble fini d'états Q, de relations de transitions $\delta_0, \ldots, \delta_k$, $\delta_i \subseteq Q^i \times \Sigma_i \times Q$, et d'un ensemble $F \subseteq Q$ d'états finaux (ou terminaux).

Si tous les $\delta_i : Q^i \times \Sigma_i \to Q$ sont des fonctions (partielles), on parle d'un automate déterministe (bottom-up DTA).

1. Contrairement aux NFA, on ne peut pas représenter les NTA par des graphes étiquetés...
2. Un calcul du NTA $\mathcal{A}$ sur un arbre t représente un étiquetage $\rho : \text{dom}(t) \to Q$ de t par des états (définition inductive) :
   - (Base : arbre réduit à une feuille a) Si $t = a \in \Sigma_0$ et $(a, q) \in \delta_0$, alors $\rho(\epsilon) = q$.
   - (Induction) Supposons que $t = a(t_1, \ldots, t_j)$ et $(q_1, \ldots, q_j, a, q) \in \delta_j$. Pour chaque $1 \leq i \leq j$, on considère un calcul $\rho_i : \text{dom}(t_i) \to Q$ sur t_i, qui étiquete la racine de t_i par q_i. Alors la fonction $\rho : \text{dom}(t) \to Q$ définie par $\rho(\epsilon) = q$ et $\rho(iv) = \rho_i(v)$ pour chaque $1 \leq i \leq j$, est un calcul de $\mathcal{A}$ sur t.

Definition, notations

1. Calcul acceptant : $\rho(\epsilon) \in F$. Langage reconnu $\mathcal{L}(\mathcal{A})$: ensemble des arbres ayant un calcul acceptant.
2. Rec$(T(\Sigma))$ ensemble des langages reconnaissables d'arbre.
3. NTA complet : pour tout $a \in \Sigma_i$ et $q_1, \ldots, q_i \in Q$ il existe un $q \in Q$ tel que $(q_1, \ldots, q_i, a, q) \in \delta_i$.
4. Soit t un arbre, q un état. On note par $t \xrightarrow{*} q$ l'existence d'un calcul ρ sur t tel que $\rho(\epsilon) = q$.
5. NTA réduit : pour tout état q il existe un arbre t tel que $t \xrightarrow{*} q$.

Proposition

Tout NTA $\mathcal{A}$ peut être transformé en un NTA équivalent, qui est complet et réduit. La construction préserve le déterminisme.

Proposition

Tout NTA peut être transformé en un DTA équivalent (déterminisation).

Automates

Automates d'arbre descendants (angl. top-down NTA)

Un automate d'arbre descendant $\mathcal{A} = \langle Q, \Sigma, \delta, I \rangle$ sur l'alphabet $\Sigma = \cup_{i=0}^{k} \Sigma_i$ est composé d'un ensemble fini d'états Q, de relations de transitions $\delta_0, \ldots, \delta_k$, $\delta_i \subseteq Q \times \Sigma_i \times Q^i$, et d'un ensemble I d'états initiaux.

Si tous les $\delta_i : Q \times \Sigma_i \to Q^i$ sont des fonctions (partielles) et $|I| = 1$, on parle d'un automate déterministe (top-down DTA).

Calcul

Un calcul de $\mathcal{A}$ sur t est un étiquetage $\rho : \text{dom}(t) \to Q$ tel que :

- $\rho(\epsilon) \in I$
- Soit $v \in dom(t)$ un nœud d'étiquette $a \in \Sigma_j$ et enfants $v1, \ldots, vj$. Alors $\rho(v) = q$ et $\rho(vi) = q_i$, pour $q, q_i \in Q$ $(1 \leq i \leq j)$ tels que $(q, a, q_1, \ldots, q_j) \in \delta$.

Un calcul ρ est acceptant, si pour toute feuille $v \in \text{dom}(t)$ d'étiquette a, on a $(\rho(v), a) \in \delta_0$.

Proposition

Les DTA descendants sont strictement plus faibles que les NTA descendants. Tout NTA descendant peut être transformé en un NTA montant équivalent.

Exemple

Le langage d'arbres (fini !) $\{c(a, b), c(b, a)\}$ est reconnu par un NTA descendant, mais pas par un DTA descendant.

Proposition

Rec$(T(\Sigma))$ *est fermé par les opérations booléennes et contient les singletons* $\{t\}$, $t \in T(\Sigma)$.

Expressions rationnelles

Produit, itération

- Soient $L, K \subseteq T(\Sigma)$ et $a \in \Sigma_0$. On définit $L \cdot_a K$ comme l'ensemble des arbres $t \in L$, dans lesquels on remplace chaque feuille d'étiquette a par un arbre de K.
- Soit $L \subseteq T(\Sigma)$, $a \in \Sigma_0$. On définit inductivement (pour $k \geq 0$) :

$$L^{0,a} = \{\epsilon\} \ (\epsilon = \text{ arbre vide}), \quad L^{k+1,a} = L^{k,a} \cdot_a L$$

- $L^{*,a} = \cup_{k \geq 0} L^{k,a}$

Definition

L'ensemble Rat($T(\Sigma)$) des expressions rationnelles sur $T(\Sigma)$ est défini par :

- $\emptyset$ et t (pour tout $t \in T(\Sigma)$) appartiennent à Rat($T(\Sigma)$)
- Si E, F sont dans Rat($T(\Sigma)$) et $a \in \Sigma_0$, alors $(E + F)$, $(E \cdot_a F)$ et $E^{*,a}$ aussi.

On identifie une expression rationnelle E et le langage $\mathcal{L}(E)$ décrit par E.

Kleene

Proposition

Pour tout alphabet $\Sigma = \Sigma_0 \cup \cdots \cup \Sigma_k$*, on a* $\mathit{Rec}(T(\Sigma)) = \mathit{Rat}(T(\Sigma))$.

Remarque

Une expression rationelle E peut utiliser plus de symboles que l'ensemble des étiquettes des arbres de $\mathcal{L}(E)$.

Exemple : Soit $L = \{f(a, f(a, \ldots (f(a, \underbrace{a) \ldots)}_{n}) \mid n \text{ pair}\}$.

On utilise l'alphabet $\Sigma = \Sigma_0 \cup \Sigma_2$, où $\Sigma_0 = \{a, c, d\}$, $\Sigma_2 = \{f\}$. Expression rationnelle pour L : $f(a, f(d, c))^{*,c} \cdot_c a \cdot_d a$.

NB : $f(a, f(d, c))^{*,c}$ peut se réecrire sans étoile (comme $(ab)^*$ sur les mots !)

Logique

Syntaxe

Soit $\text{Var}_1 = \{x, y, z, \ldots\}$, $\text{Var}_2 = \{X, Y, Z, \ldots\}$ les ensembles de variables du premier et du second ordre, et $\Sigma = \Sigma_0 \cup \cdots \cup S_k$ l'alphabet.

Les formules de MSOL sur les arbres de $T(\Sigma)$ sont définies inductivement :

- (Base) $a(x)$, $y = S_i(x)$ $(1 \leq i \leq k)$, $x \leq y$, $x \in X$ sont des formules.
- (Ind.) Si $\varphi_1, \varphi_2, \varphi$ sont des formules, alors

$$\varphi_1 \wedge \varphi_2, \quad \neg\varphi, \quad \exists x.\, \varphi, \quad \exists X.\, \varphi$$

sont des formules aussi.

Sémantique

Les variables de Var_1 sont interprétées par des nœuds, celles de Var_2 par des ensembles de nœuds.

La formule $y = S_i(x)$ est interprétée par "y est le i-ème enfant de x". La formule $x \leq y$ est interprétée par "y est descendant de x".

Logique et automates

Théorème de Thatcher/Wright

Pour tout NTA montant $\mathcal{A}$ il existe une formule MSOL $\psi_{\mathcal{A}}$ telle que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\psi_{\mathcal{A}})$, et réciproquement.

Algorithmes

1. Il existe un algorithme polynomial qui, étant donné un NTA $\mathcal{A}$, vérifie si $\mathcal{L}(\mathcal{A}) \neq \emptyset$.
2. Il existe un algorithme polynomial qui, étant donné un NTA $\mathcal{A}$, vérifie si l'ensemble $\mathcal{L}(\mathcal{A})$ est infini.