# Technique d'automates d'arbre et applications

Stefan Schwoon

2020 – 2021

## Table des matières



# Évaluation

Il y aura du contrôle continu et un examen. La note sera moitié exam moitié CC.

# 1 Préliminaires

## 1.1 Arbres et termes

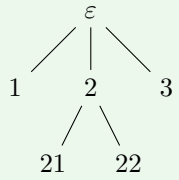
En informatique on a toujours des arbres partout. On va donc voir des algorithmes et des notions pour les gérer.

On ne considère dans ce cours que des arbres finis et ordonnés.

**Définition 1.1** Arbre fini ordonné

Un arbre est un sous-ensemble Pos de $\mathbb{N}^*$ clos par préfixe.

**Exemple 1.1 — Exemple d'arbre**




On va également classer les nœuds.

**Définition 1.2 — Symboles classés**

On a des $\mathcal{F}_i$ ensembles disjoints de symboles d'arité $i$. On note $\mathcal{F} = \bigcup_i \mathcal{F}_i$.

**Notation 1.1 — Ensemble de variables $\mathcal{X}$**

On note $\mathcal{X}$ un ensemble de variables (disjoint des autres symboles).

**Définition 1.3 — Arbre classé**

Un arbre classé est une fonction $t : \text{Pos} \to (\mathcal{F} \cup \mathcal{X})$ telle que :
— Pos est un arbre
— $\forall p \in \text{Pos}, t(p) \in \mathcal{F}_n, n > 1 \implies \text{Pos} \cap p\mathbb{N} = [\![1, n]\!]$
— $\forall p \in \text{Pos}, t(p) \in \mathcal{X} \cup \mathcal{F}_0 \implies \text{Pos} \cap p\mathbb{N} = \emptyset$.

On peut également voir les arbres comme des termes.

**Définition 1.4 — Terme**

L'ensemble des termes $\mathcal{T}(\mathcal{F}, \mathcal{X})$ est le plus petit ensemble tel que :
— $\mathcal{X} \cup \mathcal{F}_0 \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$
— si $t_1, ..., t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ et $f \in \mathcal{F}_n$ alors $f(t_1, ..., t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$.

**Définition 1.5 — Terme linéaire**

Un terme est linéaire s'il contient au plus une occurrence de chaque variable.

On définit alors la taille des termes/arbres par induction.

**Définition 1.6 — Sous-terme**

Si $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ et $p \in \text{Pos}$ alors $t_{|_p} : \text{Pos}_p \to \mathcal{T}(\mathcal{F}, \mathcal{X})$ est l'arbre classé défini par :
— $\text{Pos}_p = \{q \mid pq \in \text{Pos}\}$
— $t_{|_p}(q) = t(pq)$.

## 1.2 Substitutions et contextes

**Définition 1.7 — Substitution**

Une substitution $\sigma$ est une fonction de $\mathcal{X}$ dans $\mathcal{T}(\mathcal{F}, \mathcal{X})$.
On note $\sigma = \{x_1 \leftarrow t_1, ..., x_n \leftarrow t_n\}$ la substitution telle que $\sigma(x) = x$ si $x \neq x_1, ..., x_n$.

On étend la définition de substitution aux termes par induction et on note $t\sigma$ pour $\sigma(t)$.

**Définition 1.8 — Contexte**

Un contexte $C$ est un terme linéaire avec comme variables $x_1, ..., x_n$.
On note $C[t_1, ..., t_n] = C\{x_1 \leftarrow t_1, ..., x_n \leftarrow t_n\}$.

**Notation 1.2 — Ensemble des contextes**

On note $\mathcal{C}^n(\mathcal{F})$ l'ensemble des contextes à $n$ variables et $\mathcal{C}(\mathcal{F}) = \mathcal{C}^1(\mathcal{F})$.

**Notation 1.3** Itéré d'un contexte

Soit $C \in \mathcal{C}(\mathcal{F})$.
On note $C^0 = x_1$ et $C^{n+1} = C^n[C]$.

# 2 Automates d'arbre

L'idée va être d'étendre la définition d'automate des mots aux arbres.
On peut déjà voir les mots comme un arbre avec chaque lettre comme un symbole unaire.

## 2.1 Automate d'arbre ascendant (bottom-up) et descendant (top-bottom)

**Définition 2.1** Automate d'arbre ascendant (NFTA)

Un automate d'arbre ascendant $\mathcal{A}$ est un quadruplet $(Q, \mathcal{F}, G, \Delta)$ où :
— $Q$ est un ensemble fini d'états
— $\mathcal{F}$ est un alphabet classé
— $G \subseteq Q$ est l'ensemble des états finaux
— $\Delta$ est un ensemble fini de règles de la forme $f(q_1, ..., q_n) \to q$.

**Définition 2.2** Relation de mouvement

Soit $\mathcal{A}$ un automate d'arbre ascendant.
On a $t \to_{\mathcal{A}} t'$ ssi
— $t = C[f(q_1, ..., q_n)]$ pour un certain contexte $C$
— $t' = C[q]$ avec une règle $f(q_1, ..., q_n) \to q \in \Delta$.

**Définition 2.3** Calcul

Soit $\mathcal{A}$ un automate d'arbre et $t : \text{Pos} \to \mathcal{F}$ un arbre.
Un calcul de $\mathcal{A}$ sur $t$ c'est trouver un $t' : \text{Pos} \to Q$ compatible avec $\Delta$, c'est-à-dire que si $t(p) = f$, $t'(p) = q$ et $t'(pj) = q_j$ pour $j \in \text{Pos} \cap p\mathbb{N}$, alors $f(q_1, ..., q_n) \to q \in \Delta$.

**Définition 2.4** Langage d'arbre régulier

Un arbre $t$ est accepté par un automate d'arbre $\mathcal{A}$ ssi $t \to^*_{\mathcal{A}} q \in G$.

**Définition 2.5** Automate d'arbre avec $\varepsilon$-move

Un automate est à $\varepsilon$-move s'il a des règles de la forme $q \to q' \in \Delta$.

**Proposition 2.1** Équivalence de $\varepsilon$-NFTA

Tout automate avec $\varepsilon$-move est équivalent à un automate sans.

**Définition 2.6** Automate déterministe, complet et réduit

— Un automate est déterministe si deux règles ne peuvent pas avoir le même membre de gauche.
— Un automate est complet si pour $f \in \mathcal{F}_n$ et $q_1, ..., q_n \in Q$ il existe au moins une règle $f(q_1, ..., q_n) \to q \in \Delta$.
— Un automate est réduit si tous les états sont accessibles (il existe $t \to^*_{\mathcal{A}} q$.

Un automate déterministe a au plus un calcul sur un arbre. Un automate déterministe complet a exactement un calcul sur un arbre.

**Lemme 2.2** Lemme de l'étoile

Soit $L$ reconnaissable.
Alors il existe une constante $k$ telle que pour tout $t \in L$ tel que $\mathcal{H}(t) > k$ il existe des contextes $C, D \in \mathcal{C}(\mathcal{F})$ et $u \in \mathcal{T}(\mathcal{F})$ satisfaisant :
— $D$ est non trivial (pas une variable)
— $t = C[D[u]]$
— $\forall n \in \mathbb{N}, C[D^n[u]] \in L$.

**Définition 2.7** Automate d'arbre descendant (T-NFTA)

Un automate d'arbre descendant $\mathcal{A}$ est un quadruplet $(Q, \mathcal{F}, I, \Delta)$ où :
— $Q$ et $\mathcal{F}$ sont définis comme dans l'automate ascendant
— $I \subseteq Q$ contient les états initiaux
— $\Delta$ contient des règles de la forme $q(f) \to (q_1, ..., q_n)$.

**Définition 2.8** Calcul dans un automate descendant

On a $t \to t'$ ssi
— $t = C[q(f(t_1, ..., t_n))]$ pour un certain contexte $C$ et fonction $f \in \mathcal{F}_n$ et $t_1, ..., t_n \in \mathcal{T}(\mathcal{F})$
— $t' = C[f(q_1(t_1), ..., q_n(t_n))]$ avec une règle $q(f) \to (q_1, ..., q_n)$.

On définit de la même manière le déterminisme et la complétude.

Là y'a le cours 2 mais j'ai rien suivi.

## 2.2 Isomorphismes d'automates d'arbres

## 2.3 Chemin de langages

**Définition 2.9** Langage chemin

Soit $t \in \mathcal{T}(\mathcal{F})$ un terme. Le langage chemin $\pi(t)$ est défini par induction :
— si $t = a \in \mathcal{F}_0$ alors $\pi(t) = \{a\}$
— si $t = f(t_1, ..., t_n)$ alors $\pi(t) = \{fiw \mid w \in \pi(t_i)\}$

**Définition 2.10** Clôture de chemin

Soit $L \subseteq \mathcal{T}(\mathcal{F})$ un langage d'arbre.
— La clôture de chemin de $L$ est $\text{pc}(L) = \{t \mid \pi(t) \subseteq \pi(L)\}$
— $L$ est dit clos par chemin si $L = \text{pc}(L)$.

**Lemme 2.3**

Soit $L \subseteq \mathcal{T}(\mathcal{F})$ un langage d'arbres reconnaissable. Alors
— $\pi(L)$ est un langage reconnaissable.
— $\text{pc}(L)$ est un langage d'arbres reconnaissable.

**Théorème 2.4**

Soit $L \subseteq \mathcal{T}(\mathcal{F})$ un langage d'arbres reconnaissable.
$L$ est clos par chemin ssi il est reconnu par un T-DFTA.

## 2.4 Congruences d'arbres

**Définition 2.11** Congruence

Soit $\equiv$ une relation d'équivalence sur $\mathcal{T}(\mathcal{F})$.
$\equiv$ est une congruence si pour tout $n \in \mathbb{N}$ et $f \in \mathcal{F}_n$, si $u_1 \equiv v_1$, ..., $u_n \equiv v_n$ alors

$$f(u_1, ..., u_n) \equiv f(v_1, ..., v_n).$$

On dit que $\equiv$ sature si $u \equiv v$ implique $u \in L \iff v \in L$.

**Théorème 2.5** Théorème de Myhill-Nerode pour les arbres

Les propositions suivantes sont équivalentes
1. $L \subseteq \mathcal{T}(\mathcal{F})$ est reconnaissable
2. $L$ est saturé par une congruence d'index fini
3. $\equiv_L$ est d'index fini

## 2.5 Problème d'intersection

**Théorème 2.6**

Le problème suivant est **PSPACE**-complet :
**Données :** Des automates d'arbre $\mathcal{A}_1, ..., \mathcal{A}_n$.
**Question :** Est-ce que $\mathcal{L}(\mathcal{A}_1) \cap ... \cap \mathcal{L}(\mathcal{A}_n) = \emptyset$ ?

**Preuve du théorème 2.6.**

On simule une machine de Turing alternante en espace linéaire.

*Voilà !*

## 2.6 Automate à pile visiblement

**Définition 2.12** Automate visiblement à pile

Un automate à pile $\mathcal{A} = (Q, \Sigma, \Gamma, T, q_0 z_0, F)$ est dit visiblement à pile (VPA) si
— $\Sigma = \Sigma_0 \uplus \Sigma_1 \uplus \Sigma_2$
— $T \subseteq \bigcup_{i=0}^{2} (Q \times \Gamma) \times \Sigma_i \times (Q \times \Gamma^i)$

**Propriété 2.7** Propritété de clôture

Les langages acceptés par un VPA sont clos par opérations booléennes.

**Proposition 2.8**

Soit $L \subseteq \mathcal{T}(\mathcal{F})$ in langage d'arbres reconnaissable.
Alors $L$, vu comme un langage de mots de termes, est accepté par un VPA.

# 3 Logique sur les arbres

## 3.1 Rappels et définitions

**Définition 3.1** Logique du premier ordre

Soit $\sigma = ((R_i)_{1 \leqslant i \leqslant n})$ une signature et $\mathcal{X} = \{x_1, ...\}$ un ensemble de variables.
L'ensemble des formules du premier ordre $\mathrm{FO}(\sigma)$ est défini par la grammaire suivante :

$$R_i(x_{j_1}, ..., x_{j_i}) \mid x = x' \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \exists x.\varphi$$

On va définir également une logique du deuxième ordre, monadique, c'est-à-dire que l'on ne peut quantifier que sur les ensembles.

**Définition 3.2** Logique monadique du deuxième ordre

Soit $\sigma = ((R_i)_{1 \leqslant i \leqslant n})$ une signature, et $\mathcal{X}_1 = \{x_1, ...\}$ et $\mathcal{X}_2 = \{X_1, ...\}$ deux ensembles de variables (du premier et du deuxième rordre.
L'ensemble des formules du deuxième ordre $\mathrm{MSO}(\sigma)$ est défini par :

$$R_i(x_{j_1}, ..., x_{j_i}) \mid x = x' \mid x \in X \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \exists x.\varphi \mid \exists X.\varphi$$

On définit également une logique du second ordre faible, où l'on ne peut quantifier que sur des ensembles finis.

**Définition 3.3** Logique monadique faible du second ordre avec $k$ successeurs

On définit WS$k$S par WS$k$S $= \mathrm{MSO}(<_1, ..., <_k)$.

**Définition 3.4** Sémantique de MSO

Soit $\mathfrak{M}$ un domaine, $\sigma$ une signature et $\nu$ une valuation telle que
— $\nu(x) \in \mathfrak{M}$ pour $x \in \mathcal{X}_1$
— $\nu(X) \subseteq \mathfrak{M}$ pour $X \in \mathcal{X}_2$.

| | |
|---|---|
| $\mathfrak{M}, \sigma, \nu \models R_i(x_{j_1}, ..., x_{j_i})$ | si $(\nu(x_{j_1}), ..., \nu(x_{j_i})) \in R_i$ |
| $\mathfrak{M}, \sigma, \nu \models x = x'$ | si $\nu(x) = \nu(x')$ |
| $\mathfrak{M}, \sigma, \nu \models x \in X$ | si $\nu(x) \in \nu(X)$ |
| $\mathfrak{M}, \sigma, \nu \models \neg\varphi$ | si $\mathfrak{M}, \sigma, \nu \not\models \varphi$ |
| $\mathfrak{M}, \sigma, \nu \models \varphi \wedge \varphi'$ | si $\mathfrak{M}, \sigma, \nu \models \varphi$ et $\mathfrak{M}, \sigma, \nu \models \varphi'$ |
| $\mathfrak{M}, \sigma, \nu \models \exists x.\varphi$ | si il existe $m \in \mathfrak{M}, \mathfrak{M}, \sigma, \nu[x \mapsto m] \models \varphi$ |
| $\mathfrak{M}, \sigma, \nu \models \exists X.\varphi$ | si il existe $M \subseteq \mathfrak{M}, \mathfrak{M}, \sigma, \nu[X \mapsto M] \models \varphi$ |

## 3.2 Lien avec les arbres

On prend $\mathfrak{M} = \mathbb{N}^*$ et $p <_i q$ ssi $\exists p'.q = pip'$. On définit aussi $<= \bigcup_{i=1}^{k} <_i$ et $\leqslant$.

**Définition 3.5** Codage d'un arbre

Soit $t \in \mathcal{T}(\mathcal{F})$ un arbre et $k$ l'arité maximale de $\mathcal{F}$.
On note $C(t) = (S, (S_f)_{f \in \mathcal{F}}$ où
— $S = \bigcup_{f \in \mathcal{F}} S_f$
— pour tout $f \in \mathcal{F}$, $S_f = \{p \in \mathrm{Pos}_t \mid t(p) = f\}$.
On dit que $C(t)$ encode un arbre si la formule suivante est vérifiée :
flemme

**Définition 3.6** Valuation encodée

Soit $\mathcal{F}' = \mathcal{F} \times 2^{\mathcal{X}_1 \cup \mathcal{X}_2}$. L'arité de $(f, \tau)$ est $n$ si $f \in \mathcal{F}_n$.
Soit $t \in \mathcal{T}(\mathcal{F})$ et $\nu$ une valuation. Le tuple $\langle t, \nu \rangle$ est codé par un arbre $t' \in \mathcal{T}(\mathcal{F}')$ comme suit.
Pour tout $p \in \mathrm{Pos}$ et $t'(p) = (f, \tau)$ :
— si $x \in \mathcal{X}_1$ alors $\tau(x) = 1$ ssi $p = \nu(x)$
— si $X \in \mathcal{X}_2$ alors $\tau(X) = 1$ ssi $p \in \nu(X)$.
Un arbre $t' \in \mathcal{T}(\mathcal{F}')$ est valide s'il encode un certain $\langle t, \nu \rangle$, et on note $t \in \mathcal{T}_v(\mathcal{F}')$.

**Définition 3.7** Sémantique de WS$k$S

Soit $\varphi$ une formule de WS$k$S et $V \subseteq (\mathcal{X}_1 \cup \mathcal{X}_2) \uplus (\{S\} \cup \{S_f \mid f \in \mathcal{F}\})$ ses variables libres.

$$\mathcal{L}(\varphi) = \{\langle t, \nu \rangle \in \mathcal{T}_v(\mathcal{F}') \mid \nu[S, (S_f)_{f \in \mathcal{F}} \mapsto C(t)] \models \varphi\}$$

**Théorème 3.1**

Un arbre de langages $L \subseteq \mathcal{T}(\mathcal{F})$ est reconnaissable ssi $L = \mathcal{L}(\varphi)$ avec $\varphi$ une formule close de WS$k$S.

**Preuve du théorème 3.1.**

Pour le sens direct, on prend un DFTA $\mathcal{A}$ et on construit une formule $\varphi$ qui
— vérifie que la structure est un arbre
— devine un calcul de $\mathcal{A}$, c'est-à-dire partitionne $S$ en états
— vérifie que le calcul est localement correct
— vérifie que la racine est labellisée par un état acceptant
Pour le sens réciproque, on montre par récurrence sur $\varphi$ que toutes les sous-formules sont reconnaissables.

*Youpi !*

# 4 Arbres non classés

## 4.1 Définition

On considère maintenant des arbres non classés : ils sont toujours ordonnés (les nœuds internes ont des enfants ordonnés de 1 à $n$), mais ils peuvent avoir un nombre arbitraire fini d'enfants.

C'est le cas des documents XML par exemple.

**Définition 4.1** Automate couvrant ascendant (NHA)

Un automate couvrant ascendant est un quadruplet $\mathcal{A} = (Q, \Sigma, G, \Delta)$ où
— $Q$ est un ensemble fini d'états
— $\Sigma$ est un alphabet fini
— $G \subseteq Q$ est un ensemble d'états finaux
— $\Delta$ est un ensemble fini de règles de la forme $a(R) \to q$ pour $a \in \Sigma$, $q \in Q$ et $R$ un langage régulier (de mots) sur $Q$.

**Exemple 4.1**

Avec
— $Q = \{q_\times, q_h, q_b, q_p\}$
— $\Sigma = \{\times, h, b, p\}$
— $G = \{q_\times\}$
— $\Delta = \{\times(q_h^? q_b) \to q_\times \quad h(\varepsilon) \to q_h \quad b(q_p^*) \to q_b \quad p(\varepsilon) \to q_p\}$
On accepte les arbres de la forme $\times(h, b(p, ..., p))$ et $\times(b(p, ..., p))$.

**Définition 4.2** Calcul d'un NHA

Soit $t \in \mathcal{T}(\Sigma)$ un arbre. Un calcul de $\mathcal{A}$ sur $t$ est un arbre $t' \in \mathcal{T}(Q)$, c'est-à-dire que pour tout $p \in \text{Pos}$, si
— $t(p) = a \in \Sigma$
— $t'(p) = a \in Q$
— $\text{Pos} \cap pN = \{p1, ..., pn\}$
alors il existe $a(R) \to q \in \Delta$ tel que $t'(p1) \cdots t'(pn) \in R$.

**Définition 4.3** NHA complet, plein, réduit, déterministe

Un NHA est
— complet si $\forall t \in \mathcal{T}(\Sigma), t \to_{\mathcal{A}}^* q$ pour un certain $q$
— plein si $\forall a \in \Sigma, \forall q \in Q, \exists a(R) \to q \in \Delta$
— réduit si $a(R_1) \to q, a(R_2) \to q \in \Delta$ implique $R_1 = R_2$
— déterministe (DHA) si $a(R_1) \to q_1, a(R_2) \to q_2 \in \Delta$ impliquent $R_1 \cap R_2 = \emptyset$ ou $q_1 = q_2$.

## 4.2 Résultats

**Proposition 4.1** Déterminisation d'un NHA

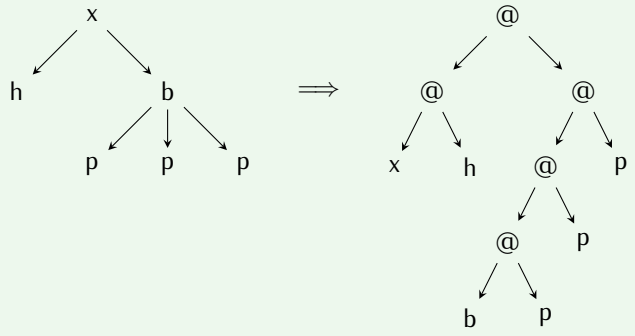
Soit $\mathcal{A} = (Q, \Sigma, G, \Delta)$ un NHA complet, plein et réduit.
Il est équivalent au DHA $\mathcal{A}' = (2^Q, \Sigma, G', \Delta')$ où
— $G' = \{S \subseteq Q \mid S \cap G \neq \emptyset\}$
— Soit $R_{a,q}$ le langage (unique) tel que $a(R_{a,q}) \to q \in \Delta$
— $R'_{a,q} = R_{a,q}[q' \to (S \cup \{q'\}) \mid q' \in Q, S \subseteq S]$
— Pour tout $a \in \Sigma, S \subseteq Q$, on a que $a(R_{a,S}) \to S \in \Delta'$

**Proposition 4.2** Bijection entre les arbres classés et non classés

Soit $\Sigma$ un alphabet et $\mathcal{F}_\Sigma = \{@(2)\} \cup \{a(0) \mid a \in \Sigma\}$.
On définit le codage $C_@(t) \in \mathcal{T}(\mathcal{F}_\Sigma)$ de $t \in \mathcal{T}(\Sigma)$ par

$$C_@(a(t_1, ..., t_n)) = \underbrace{@(@(...(@}_{n}(a, C_@(t_1)), C_@(t_2)), ...), C_@(t_n))$$

**Exemple 4.2**




**Théorème 4.3**

Un langage $L \subseteq \mathcal{T}(\Sigma)$ est couverture-reconnaissable ssi $C_@(L)$ est reconnaissable.

**Corollaire 4.4**

La reconnaissabilité couverte est close par opérations binaires.

## 4.3 Liens avec la logique

**Définition 4.4** Logique sur les arbres non classés (UTL)

On définit UTL par MSO(enfant, suivant) interprété sur $\mathfrak{M} = \mathbb{N}^*$, où
— enfant$(x, y)$ ssi $y = xi$ pour un certain $i \in \mathbb{N}$
— suivant$(x, y)$ ssi $\exists z, i, x = zi \wedge y = z(i+1)$.

On peut alors définir des prédicats droit pour $y$ est un descendant droit de $x$, et descendant pour $y$ est un descendant de $x$.

**Théorème 4.5** UTL = NHA

Un langage $L \subseteq \mathcal{T}(\Sigma)$ est couverture-reconnaissable ssi $L = \mathcal{L}(\varphi)$ pour une certaine formule $\varphi(S, S_\Sigma)$ de UTL.

# 5 Tuples d'arbres

## 5.1 Définitions

**Définition 5.1** Couple d'arbre

Soient $t_1, t_2 \in \mathcal{T}(\mathcal{F})$ des arbres classés.
On ajoute un nouveau symbole $-$ à $\mathcal{F}_0$ et on pose

$$\mathcal{F}' := \{\langle f, g\rangle(k) \mid f \in \mathcal{F}_m, g \in \mathcal{F}_n, k = \max(m, n).$$

$\langle t_1, t_2\rangle$ désigne l'arbre $t$ de $\mathcal{T}(\mathcal{F})$ défini par
— $\mathrm{Pos}_t = \mathrm{Pos}_{t_1} \cup \mathrm{Pos}_{t_2}$
— Pour $p \in \mathrm{Pos}_t$,

$$t(p) = \begin{cases} \langle f, g\rangle & \text{si } t \in \mathrm{Pos}_{t_1} \cap \mathrm{Pos}_{t_2}, t_1(p) = f, t_2(p) = g \\ \langle f, -\rangle & \text{si } t \in \mathrm{Pos}_{t_1} \setminus \mathrm{Pos}_{t_2}, t_1(p) = f \\ \langle -, g\rangle & \text{si } t \in \mathrm{Pos}_{t_2} \setminus \mathrm{Pos}_{t_1}, t_2(p) = g. \end{cases}$$

On considère une relation binaire $R \subseteq \mathcal{T}(\mathcal{F})^2$.

**Notation 5.1**

On note
— $\mathfrak{R}_2$ la classe des relations reconnaissables (langages reconnaissables sur $\mathcal{F}'$)
— $\mathfrak{X}_2$ la classe des unions finies de cross-produits, pour $L_1^{(i)}$ et $L_2^{(i)}$ reconnaissables,

$$\mathfrak{X}_2 = \bigcup_{i=1}^{n} \left( L_1^{(i)} \times L_2^{(i)} \right)$$

— $\mathfrak{T}_2$ la classe des relations reconnaissables par un GTT.

**Définition 5.2** **Transducteur d'arbres au sol (GTT)**

Un transducteur d'arbres au sol est une paire $\mathcal{G} = \langle \mathcal{A}_1, \mathcal{A}_2 \rangle$ d'automates d'arbres ascendants NFTA sur $\mathcal{F}$ (dont les états peuvent ne pas être disjoints).
La relation acceptée par $\mathcal{G}$ est

$$degueulasse$$

## 5.2 Propriétés

**Proposition 5.1** **Relations entre $\mathfrak{R}_2$, $\mathfrak{X}_2$ et $\mathfrak{T}_2$**

1. $\mathfrak{R}_2 \not\subseteq \mathfrak{X}_2$ et $\mathfrak{T}_2 \not\subseteq \mathfrak{X}_2$
2. $\mathfrak{R}_2 \not\subseteq \mathfrak{T}_2$ et $\mathfrak{X}_2 \not\subseteq \mathfrak{T}_2$
3. $\mathfrak{X}_2 \subseteq \mathfrak{R}_2$
4. $\mathfrak{T}_2 \subseteq \mathfrak{R}_2$
5. $\mathfrak{X}_2 \cup \mathfrak{T}_2 \subsetneq \mathfrak{R}_2$

**Preuve du proposition 5.1.**

1. $\{\langle t, t \rangle \mid t \in \mathcal{T}(\mathcal{F})\}$ est dans $\mathfrak{T}_2 \cap \mathfrak{R}_2$ mais pas dans $\mathfrak{X}_2$.

2. $\emptyset$ est dans $\mathfrak{X}_2 \cap \mathfrak{R}_2$ mais pas dans $\mathfrak{T}_2$.

3. On prend $A_i = (Q_i, \mathcal{F}, G_i, \Delta_i)$ des NFTA et $R = \mathcal{L}(\mathcal{A}_1) \times \mathcal{L}(\mathcal{A}_2) \in \mathfrak{X}_2$.

4. On prend $\mathcal{G} = (\mathcal{A}_1, \mathcal{A}_2)$, $A_i = (Q_i, \mathcal{F}, G_i, \Delta_i)$.
On construit un NFTA $\mathcal{A}' = (Q', \mathcal{F}', \{q_f\}, \Delta')$ avec $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{G})$.

5. On prend $\mathcal{F} = \{f(1), g(1), a\}$ et

$$R = \{\langle t_1, t_2 \rangle \mid \exists C \in \mathcal{C}(\mathcal{F}), t \in \mathcal{T}(\mathcal{F}) : t_1 = C[t] \wedge t_2 = C[f(t)]\}.$$

$R \notin \mathfrak{X}_2$ car $\langle a, f(a) \rangle \in R$ et $\langle f(a), f(f(a)) \rangle \in R$, mais $\langle a, f(f(a)) \rangle \notin R$
$R \notin \mathfrak{T}_2$ car si $R$ est accepté par un GTT $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$ ayant $n$ états en commun, alors pour tout $i \in \mathbb{N}$
$R \in \mathfrak{R}_2$ avec $\mathcal{A} = (\{q_a, q_f, q_g, q\}, \mathcal{F}, \{q\}, \Delta)$ où, pour $x, y \in \{f, g, a\}$,

$$\langle -, a \rangle \to q_a \qquad \langle x, y \rangle(q_x) \to q_y \qquad q_f \to q \qquad \langle x, x \rangle \to q$$

*Youpi !*

**Propriété 5.2** **Clôture par opérations binaires**

$\mathfrak{X}_2$ et $\mathfrak{R}_2$ sont clos par opérations binaires.

**Propriété 5.3** **Clôture transitive**

Si $R \in \mathfrak{T}_2$ alors $R^* \in \mathfrak{T}_2$.

**Preuve du propriété 5.3.**

*Voilà !*

# Index des définitions

# Index des résultats