

Quelques réductions à la décidabilité du fragment multiplicatif et exponentiel de la logique linéaire.

Philippe de Groote, Bruno Guillaume, Sylvain Salvati

INRIA Bordeaux sud-ouest, université de Bordeaux, LaBRI

Séminaire IML, 17 janvier 2008

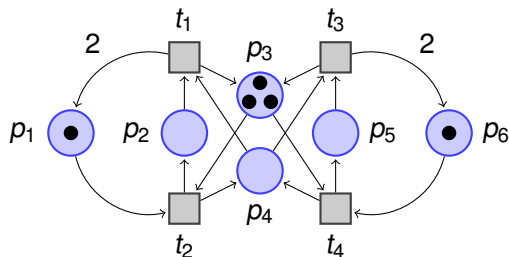
Des réseaux de Petri au VATA

Les VATA et MELL

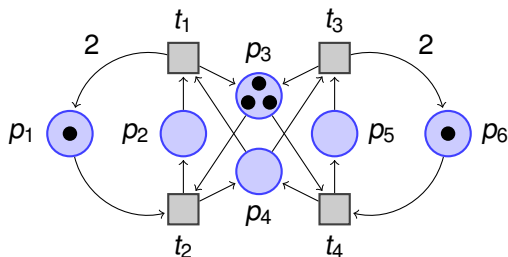
Les grammaires minimalistes et MELL

Réseaux de Petri, et logique linéaire

- ▶ Les réseaux de Pétri modélisent le parallélisme et l'accès concurrent à des ressources.



Réseaux de Petri, et logique linéaire



Le problème de l'accessibilité dans les réseaux de Pétri peut être vu comme un problème de recherche démonstration dans MELL. Démontrer le séquent:

$$!T_1, !T_2, !T_3, !T_4, p_1, p_3, p_3, p_3, p_6 \vdash p_4$$

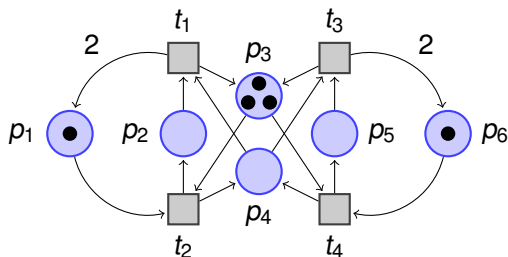
(avec, entre autre, $T_1 = (p_2 \otimes p_4) \multimap (p_1 \otimes p_1 \otimes p_3)$)

revient à montrer que la configuration où seul p_4 contient un jeton peut être atteinte à partir de la configuration initiale.

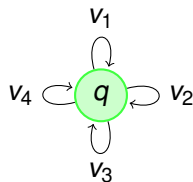
En revanche, la réciproque n'a pu être démontrée:

- ▶ on cherche un modèle de calcul plus général que les réseaux

Réseaux de Petri et automates



Les réseaux de Petri peuvent être simulés par des VAS:



$$c = (1, 0, 3, 0, 0, 1)$$

$$v_1 = (2, -1, 1, -1, 0, 0)$$

$$v_2 = (-1, 1, -1, 1, 0, 0)$$

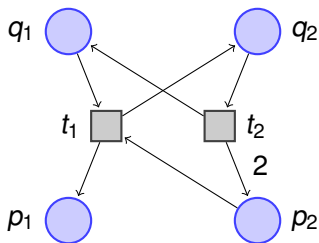
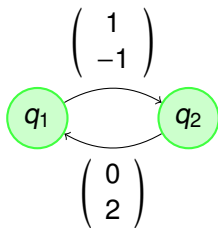
$$v_3 = (0, 0, 1, -1, -1, 2)$$

$$v_4 = (0, 0, -1, 1, 1, -1)$$

Une configuration est de la forme (q, v) où q est un état et $v \in \mathbb{N}^k$.
On ne peut utiliser une transition $(q, \mathbf{x}) \rightarrow (q', \mathbf{x} + \mathbf{z})$ que ssi $\mathbf{x} + \mathbf{z}$ est dans \mathbb{N}^k .

Réseaux de Petri et automates

Les VAS peuvent être simulés par des réseaux de Petri:



Généraliser les réseaux de Petri

	Chaînes	Arbres
États	$q(aw) \rightarrow q'(w)$	$a(q_1, \dots, q_n) \rightarrow q$

Généraliser les réseaux de Petri

	Chaînes	Arbres
États	$q(aw) \rightarrow q'(w)$	$a(q_1, \dots, q_n) \rightarrow q$
États et vecteurs	$[q, \mathbf{x}](aw)$ \rightarrow $[q', \mathbf{x} - \mathbf{y} + \mathbf{z}](w)$ avec $\mathbf{x} - \mathbf{y} \in \mathbb{N}^k$	

Généraliser les réseaux de Petri

	Chaînes	Arbres
États	$q(aw) \rightarrow q'(w)$	$a(q_1, \dots, q_n) \rightarrow q$
États et vecteurs	$[q, \mathbf{x}](aw)$ \rightarrow $[q', \mathbf{x} - \mathbf{y} + \mathbf{z}](w)$ avec $\mathbf{x} - \mathbf{y} \in \mathbb{N}^k$	$a([q_1, \mathbf{x}_1], \dots, [q_n, \mathbf{x}_n])$ \rightarrow $[q, (\sum_{i \in [1, n]} \mathbf{x}_i - \mathbf{z}_i) + \mathbf{z}]$ avec $\mathbf{x}_i - \mathbf{z}_i \in \mathbb{N}^k$

Des réseaux de Petri au VATA

Les VATA et MELL

Les grammaires minimalistes et MELL

Les automates d'arbres d'addition de vecteurs

Les k -VATA, sont des automates:

- ▶ qui généralisent les réseaux de Petri,

Les automates d'arbres d'addition de vecteurs

Les k -VATA, sont des automates:

- ▶ qui généralisent les réseaux de Petri,
- ▶ et la décidabilité de la vacuité de leur langage est équivalente à la décidabilité de **MELL** (dGGS 04).

Les automates d'arbres d'addition de vecteurs

Les k -VATA, sont des automates:

- ▶ qui généralisent les réseaux de Petri,
- ▶ et la décidabilité de la vacuité de leur langage est équivalente à la décidabilité de **MELL** (dGGS 04).

Les règles sont de la forme:

$$f((q_0, \mathbf{x}_0), \dots, (q_{n-1}, \mathbf{x}_{n-1})) \longrightarrow \left(q, \sum_{i \in n} (\mathbf{x}_i - \mathbf{z}_i) + \mathbf{z} \right)$$

Dans ces règles:

- ▶ \mathbf{x}_i sont des variables représentant des éléments de \mathbb{N}^k
- ▶ $\mathbf{z}_i \in \mathbb{N}^k$ et $\mathbf{z} \in \mathbb{N}^k$

Les automates d'arbres d'addition de vecteurs

Les k -VATA, sont des automates:

- ▶ qui généralisent les réseaux de Petri,
- ▶ et la décidabilité de la vacuité de leur langage est équivalente à la décidabilité de **MELL** (dGGS 04).

Les règles sont de la forme:

$$f((q_0, \mathbf{x}_0), \dots, (q_{n-1}, \mathbf{x}_{n-1})) \longrightarrow \left(q, \sum_{i \in n} (\mathbf{x}_i - \mathbf{z}_i) + \mathbf{z} \right)$$

Dans ces règles:

- ▶ \mathbf{x}_i sont des variables représentant des éléments de \mathbb{N}^k
- ▶ $\mathbf{z}_i \in \mathbb{N}^k$ et $\mathbf{z} \in \mathbb{N}^k$

Ces règles ne sont applicables que si $\mathbf{x}_i - \mathbf{z}_i \in \mathbb{N}^k$.

Les automates d'arbres d'addition de vecteurs

Les k -VATA, sont des automates:

- ▶ qui généralisent les réseaux de Petri,
- ▶ et la décidabilité de la vacuité de leur langage est équivalente à la décidabilité de **MELL** (dGGS 04).

Les règles sont de la forme:

$$f((q_0, \mathbf{x}_0), \dots, (q_{n-1}, \mathbf{x}_{n-1})) \longrightarrow \left(q, \sum_{i \in n} (\mathbf{x}_i - \mathbf{z}_i) + \mathbf{z} \right)$$

Dans ces règles:

- ▶ \mathbf{x}_i sont des variables représentant des éléments de \mathbb{N}^k
- ▶ $\mathbf{z}_i \in \mathbb{N}^k$ et $\mathbf{z} \in \mathbb{N}^k$

Ces règles ne sont applicables que si $\mathbf{x}_i - \mathbf{z}_i \in \mathbb{N}^k$.

Un arbre t appartient au langage défini par un VATA ssi les règles permettent de le réécrire dans une configuration acceptante.

VATA sous forme normale

Pour tout k -VATA \mathcal{A} , il existe un autre k -VATA \mathcal{B} dont le langage est vide ssi le langage de \mathcal{A} est vide et dont les règles sont de la forme:

- ▶ $f \longrightarrow (q, \mathbf{e}_i)$ pour $i \in [1, k]$
- ▶ $f((q_0, \mathbf{x}_0)) \longrightarrow (q, \mathbf{x}_0 - \mathbf{e}_i)$ pour $i \in [1, k]$,
- ▶ $f((q_0, \mathbf{x}_0), (q_1, \mathbf{x}_1)) \longrightarrow (q, \mathbf{x}_0 + \mathbf{x}_1)$.
- ▶ il y a une seule configuration acceptante: $(q_f, \mathbf{0})$

Vers un fragment simple de MELL

- ▶ Nous prouvons que la décidabilité de MELL est équivalente à celle d'un fragment très simple de IMELL.
- ▶ L'équivalence de la décidabilité de MELL et de IMELL s'obtient en utilisant une traduction de Kolmogorov.
- ▶ On procède en montrant que la décidabilité de IMELL est équivalente aux fragments suivants:

IMELL_0	Séquents de IMELL de la forme $!\Sigma, \Gamma \vdash A$. Les formules de Σ , Γ et A ne contiennent pas de $!$
IMELL_0^-	Séquent de IMELL_0 ne contenant que de formules implicatives
s- IMELL_0^-	Séquents de IMELL_0^- tels que les formules de Σ ne contiennent au plus que deux implications, celles de Γ sont atomiques et A est atomique

Presentation de IMELL

$$\frac{}{A \vdash A} \text{Ax.} \quad \frac{\Gamma \vdash A \quad \Delta, A \vdash B}{\Gamma, \Delta \vdash B} \text{Cut}$$

$$\frac{\Gamma \vdash A}{\Gamma, \mathbf{1} \vdash A} \mathbf{1}\text{-left} \quad \frac{}{\vdash \mathbf{1}} \mathbf{1}\text{-right}$$

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C} \otimes\text{-left} \quad \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \otimes\text{-right}$$

$$\frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \multimap B \vdash C} \multimap\text{-left} \quad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} \multimap\text{-right}$$

$$\frac{\Gamma \vdash B}{\Gamma, !A \vdash B} \text{Weakening} \quad \frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B} \text{Contraction}$$

De IMELL à IMELL₀

Soit $\Gamma \vdash A$ un séquent de IMELL:

- ▶ on note \mathcal{M} l'ensemble des formules F telles que $!F$ est une sous-formule de $\Gamma \vdash A$.
- ▶ Pour chaque $F \in \mathcal{M}$ on se donne une nouvelle formule atomique p_F .

On traduit $\Gamma \vdash A$ en $!\Sigma, \Gamma^* \vdash A^*$ avec:

- ▶ $\mathbf{1}^* = \mathbf{1}$, $a^* = a$, lorsque a est atomique,
- ▶ $(F \otimes G)^* = F^* \otimes G^*$, $(!F)^* = p_F$

Σ simule les règles associées à $!$ et pour chaque F , Σ est le plus grand ensemble qui contient les formules:

- ▶ $p_F \multimap F^*$ (déréliction),
- ▶ $p_F \multimap p_F \otimes p_F$ (contraction),
- ▶ $p_F \multimap \mathbf{1}$ (affaiblissement).

et tel que $G = p_{F_1} \multimap \dots \multimap p_{F_n} \multimap p_F \in \Sigma$ (avec $F_i \neq F_j$ si $i \neq j$) ssi:

$$!(\Sigma - \{G\}), p_{F_1}, \dots, p_{F_n} \vdash F^*$$

est prouvable.

Théorème: $\Gamma \vdash A$ est prouvable ssi $!\Sigma, \Gamma^* \vdash A^*$ est prouvable.

De IMELL_0 à IMELL_0^-

Soit $!\Sigma, \Gamma \vdash A$ un séquent de IMELL_0 , on le traduit en $!(\Sigma^+), \Gamma^+ \vdash A^+$ où (b est une nouvelle formule atomique):

- ▶ $\mathbf{1}^+ = b \multimap b$,
- ▶ $a^+ = (a \multimap b) \multimap b$,
- ▶ $(A \otimes B)^+ = (A^+ \multimap B^+ \multimap b) \multimap b$,
- ▶ $(A \multimap B)^+ = ((A^+ \multimap B^+) \multimap b) \multimap b$.

Théorème: $!\Sigma, \Gamma \vdash A$ est démontrable ssi $!(\Sigma^+), \Gamma^+ \vdash A^+$ est démontrable.

De $\text{IMELL}_0^{-\circ}$ à $\text{s-IMELL}_0^{-\circ}$

La transformation d'un séquent de $\text{IMELL}_0^{-\circ}$ à un séquent de $\text{s-IMELL}_0^{-\circ}$ est basée sur l'utilisation répétée de la propriété:

Lemme: Si $\Gamma \vdash A$ et $\Gamma' \vdash A'$ sont des séquent de IMELL tels que p est atomique et que $\Gamma = \Gamma'[p := F]$ et $A = A'[p := F]$ alors on a: $\Gamma \vdash A$ est démontrable ssi $!(p \multimap F), !(F \multimap p), \Gamma' \vdash A'$ est démontrable.

Démonstration dans s-IMELL₀[→]

Les démonstrations des séquents de s-IMELL₀[→] peuvent être obtenues en utilisant le système suivant:

$$\frac{}{!\Sigma, a \vdash a} (T_0) \quad \frac{!\Sigma, \Gamma \vdash a \quad a \multimap b \in \Sigma}{!\Sigma, \Gamma \vdash b} (T_1)$$

$$\frac{!\Sigma, \Gamma, a \vdash b \quad (a \multimap b) \multimap c \in \Sigma}{!\Sigma, \Gamma \vdash c} (T_2)$$

$$\frac{!\Sigma, \Gamma \vdash a \quad !\Sigma, \Delta \vdash b \quad a \multimap b \multimap c \in \Sigma}{!\Sigma, \Gamma, \Delta \vdash c} (T_3)$$

s-IMELL₀^o et les VATA

Soit \mathcal{A} un k -VATA sous forme normale.

On utilise comme formules atomiques:

- ▶ l'ensemble des états de \mathcal{A} ,
- ▶ l'ensemble $\{a_1; \dots; a_k\}$.

On définit $\Sigma_{\mathcal{A}}$ contient les formules:

- ▶ $a_i \multimap q$ si $f \rightarrow (q, \mathbf{e}_i)$ est une règle de \mathcal{A} ,
- ▶ $(a_i \multimap q_0) \multimap q$ si $f((q_0, \mathbf{x}_0)) \rightarrow (q, \mathbf{x}_0 - \mathbf{e}_i)$ est une règle de \mathcal{A} ,
- ▶ $q_0 \multimap (q_1 \multimap q)$ si $f((q_0, \mathbf{x}_0), (q_1, \mathbf{x}_1)) \rightarrow (q, \mathbf{x}_0 + \mathbf{x}_1)$ est une règle de \mathcal{A}

Soit $\mathbf{x} = (n_1, \dots, n_k)$ on note $\Gamma_{\mathbf{x}}$ le contexte $a_1^{n_1}, \dots, a_k^{n_k}$.

Théorème: $!\Sigma, \Gamma_{\mathbf{x}} \vdash q$ est démontrable ssi il existe un arbre que \mathcal{A} réécrit en (q, \mathbf{x}) .

s-IMELL₀[→] et les VATA

Si $!\Sigma, \Gamma \vdash a$ est un séquent de s-IMELL₀[→] alors on note:

- ▶ $\{a_1; \dots; a_k\}$ l'ensemble des formules atomiques du séquent,
- ▶ $\{f_1; \dots; f_n\}$ celui des formules de la forme $a \multimap b$,
- ▶ $\{g_1; \dots; g_m\}$ celui des formules de la forme $a \multimap (b \multimap c)$,
- ▶ $\{h_1; \dots; h_p\}$ celui des formules de la forme $(a \multimap b) \multimap c$.

On représente $\Gamma = a_1^{n_1}, \dots, a_k^{n_k}$ par $\mathbf{x}_\Gamma = (n_1, \dots, n_k)$. Et on définit \mathcal{A}_Σ , l'automate dont les règles sont:

- ▶ $f_i \longrightarrow [b, \mathbf{e}_j]$ si $f_i = a_j \multimap b$,
- ▶ $g_i((b, \mathbf{x})) \longrightarrow (c, \mathbf{x} - \mathbf{e}_j)$ si $g_i = (a_j \multimap b) \multimap c$
- ▶ $h_i((a, \mathbf{x}_1), (b, \mathbf{x}_2)) \longrightarrow (c, \mathbf{x}_1, \mathbf{x}_2)$ si $h_i = a \multimap b \multimap c$.

La configuration acceptante de \mathcal{A} est (a, \mathbf{x}_Γ)

Théorème: le langage de \mathcal{A} n'est pas vide ssi $!\Sigma, \Gamma \vdash a$ est démontrable.

Des réseaux de Petri au VATA

Les VATA et MELL

Les grammaires minimalistes et MELL

Les grammaires minimalistes

Les grammaires minimalistes sont proposées par (Stabler 97) pour rendre compte du programme minimaliste de Chomsky. Nous montrons que:

- ▶ la décidabilité du problème de l'analyse pour ces grammaires est équivalente à celle de MELL,
- ▶ les analyses sont bien représentées par des démonstrations en logique linéaire.

Les grammaires minimalistes

Une grammaire minimaliste G_{MG} est un quintuplet $\langle B, F, W, \mathcal{L}, c \rangle$
où:

- ▶ B est un ensemble fini de traits de base,

Les grammaires minimalistes

Une grammaire minimaliste G_{MG} est un quintuplet $\langle B, F, W, \mathcal{L}, c \rangle$
où:

- ▶ B est un ensemble fini de traits de base,
- ▶ F est un ensemble fini de traits de mouvement,

Les grammaires minimalistes

Une grammaire minimaliste G_{MG} est un quintuplet $\langle B, F, W, \mathcal{L}, c \rangle$
où:

- ▶ B est un ensemble fini de traits de base,
- ▶ F est un ensemble fini de traits de mouvement,
- ▶ W est un ensemble fini de mot,

Les grammaires minimalistes

Une grammaire minimaliste G_{MG} est un quintuplet $\langle B, F, W, \mathcal{L}, c \rangle$
où:

- ▶ B est un ensemble fini de traits de base,
- ▶ F est un ensemble fini de traits de mouvement,
- ▶ W est un ensemble fini de mot,
- ▶ \mathcal{L} est un sous-ensemble fini de:

$$W \cup \{\epsilon\} \times [= B[= B| + F|\bar{+}F]^*]^? B[-F]^*$$

Les grammaires minimalistes

Une grammaire minimaliste G_{MG} est un quintuplet $\langle B, F, W, \mathcal{L}, c \rangle$
où:

- ▶ B est un ensemble fini de traits de base,
- ▶ F est un ensemble fini de traits de mouvement,
- ▶ W est un ensemble fini de mot,
- ▶ \mathcal{L} est un sous-ensemble fini de:

$$W \cup \{\epsilon\} \times [= B[= B| + F|\bar{+}F]^*]^? B[-F]^*$$

- ▶ $c \in B$

Les grammaires minimalistes

Une grammaire minimaliste G_{MG} est un quintuplet $\langle B, F, W, \mathcal{L}, c \rangle$
où:

- ▶ B est un ensemble fini de traits de base,
- ▶ F est un ensemble fini de traits de mouvement,
- ▶ W est un ensemble fini de mot,
- ▶ \mathcal{L} est un sous-ensemble fini de:

$$W \cup \{\epsilon\} \times [= B[= B| + F|\bar{+}F]^*] ? B[-F]^*$$

- ▶ $c \in B$

Nous nous plaçons de point de vue algébrique de (Stabler 97).

Les dérivations de la grammaire G_{MG} sont des arbres définis par:

$$\mathcal{D}_{G_{MG}} := \mathcal{L} | \mathbf{merge}(\mathcal{D}_{G_{MG}}, \mathcal{D}_{G_{MG}}) | \mathbf{move}(\mathcal{D}_{G_{MG}})$$

Les grammaires minimalistes

Une grammaire minimaliste G_{MG} est un quintuplet $\langle B, F, W, \mathcal{L}, c \rangle$
où:

- ▶ B est un ensemble fini de traits de base,
- ▶ F est un ensemble fini de traits de mouvement,
- ▶ W est un ensemble fini de mot,
- ▶ \mathcal{L} est un sous-ensemble fini de:

$$W \cup \{\epsilon\} \times [= B[= B| + F|\bar{+}F]^*]^? B[-F]^*$$

- ▶ $c \in B$

Nous nous plaçons de point de vue algébrique de (Stabler 97).

Les dérivations de la grammaire G_{MG} sont des arbres définis par:

$$\mathcal{D}_{G_{MG}} := \mathcal{L} | \mathbf{merge}(\mathcal{D}_{G_{MG}}, \mathcal{D}_{G_{MG}}) | \mathbf{move}(\mathcal{D}_{G_{MG}})$$

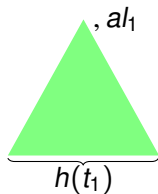
Ces arbres sont interprétés par une fonction partielle h comme des arbres dérivés:

$$\mathcal{T}_{G_{MG}} := W \cup \{\epsilon\} | \langle (\mathcal{T}_{G_{MG}}, \mathcal{T}_{G_{MG}}) | \rangle (\mathcal{T}_{G_{MG}}, \mathcal{T}_{G_{MG}}) | (\mathcal{T}, \mathbb{L}_{G_{MG}})$$

où $\mathbb{L}_{G_{MG}} = \{l(v, l') \in \mathcal{L}\}$.

Définition de h : merge

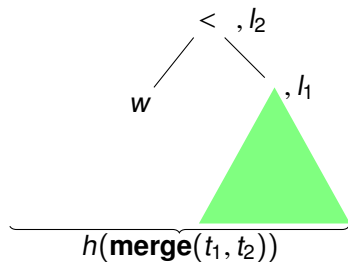
Calcul de $h(\text{merge}(t_1, t_2))$ cas 1:



$$\underbrace{(w, = a_2)}_{h(t_2)}$$

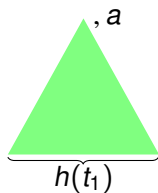
Définition de h : merge

Calcul de $h(\text{merge}(t_1, t_2))$ cas 1:



Définition de h : merge

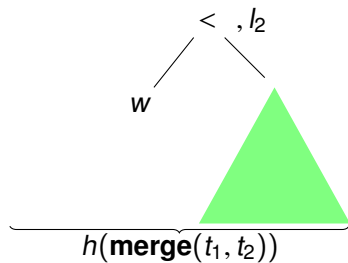
Calcul de $h(\text{merge}(t_1, t_2))$ cas 2:



$$\underbrace{(w, = a/2)}_{h(t_2)}$$

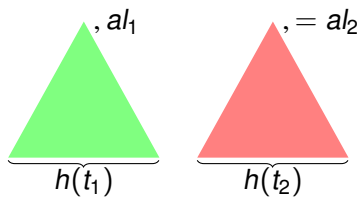
Définition de h : merge

Calcul de $h(\text{merge}(t_1, t_2))$ cas 2:



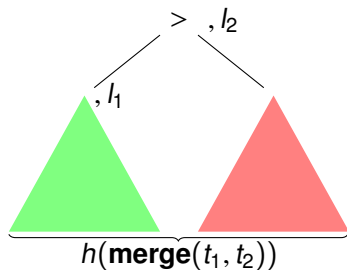
Définition de h : merge

Calcul de $h(\text{merge}(t_1, t_2))$ cas 3:



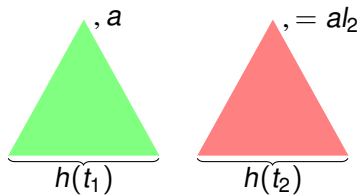
Définition de h : merge

Calcul de $h(\text{merge}(t_1, t_2))$ cas 3:



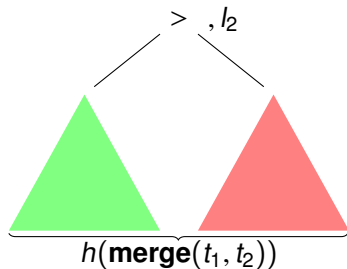
Définition de h : merge

Calcul de $h(\text{merge}(t_1, t_2))$ cas 4:



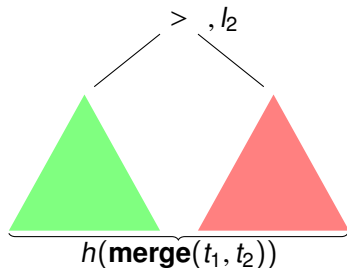
Définition de h : merge

Calcul de $h(\text{merge}(t_1, t_2))$ cas 4:



Définition de h : merge

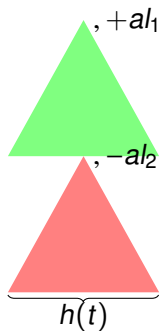
Calcul de $h(\text{merge}(t_1, t_2))$ cas 4:



Dans les autres cas $h(\text{merge}(t_1, t_2))$ n'est pas défini.

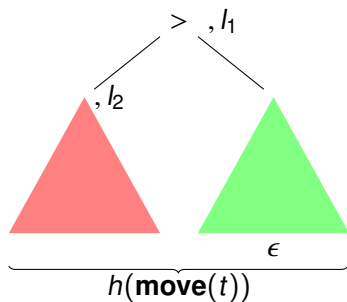
Définition de h : **move**

Calcul de $h(\mathbf{move}(t))$ cas 1:



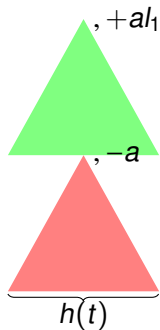
Définition de h : **move**

Calcul de $h(\mathbf{move}(t))$ cas 1:



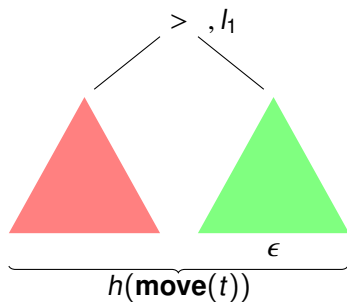
Définition de h : **move**

Calcul de $h(\mathbf{move}(t))$ cas 2:



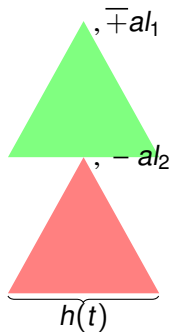
Définition de h : **move**

Calcul de $h(\mathbf{move}(t))$ cas 2:



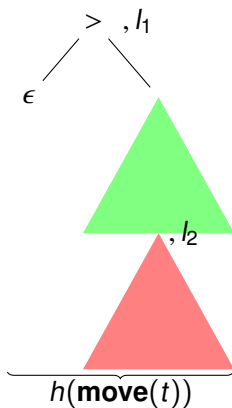
Définition de h : **move**

Calcul de $h(\mathbf{move}(t))$ cas 3:



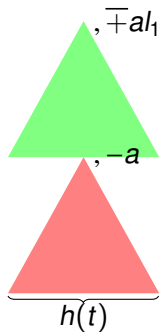
Définition de h : **move**

Calcul de $h(\mathbf{move}(t))$ cas 3:



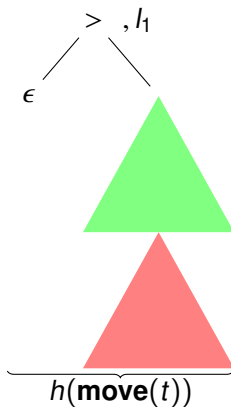
Définition de h : **move**

Calcul de $h(\mathbf{move}(t))$ cas 4:



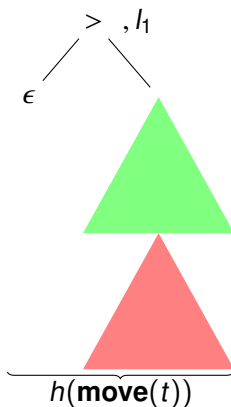
Définition de h : **move**

Calcul de $h(\mathbf{move}(t))$ cas 4:



Définition de h : **move**

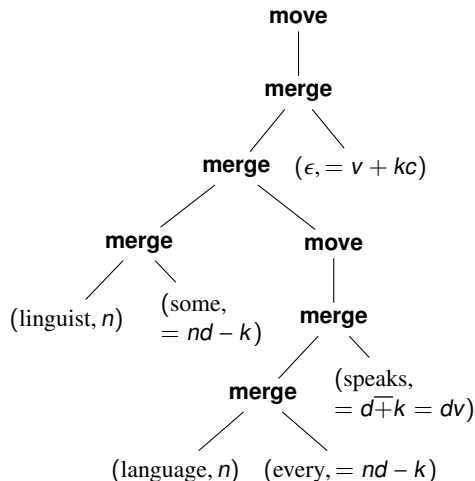
Calcul de $h(\mathbf{move}(t))$ cas 4:



Dans les autres cas $h(\mathbf{move}(t))$ n'est pas défini.

Un exemple

On prend le lexique suivant: $(\text{every}, = nd - k)$, $(\text{language}, n)$, $(\text{some}, = nd - k)$, $(\text{linguist}, n)$, $(\text{speaks}, = d + k = dv)$, $(\epsilon, = v + kc)$.



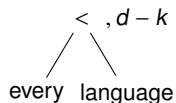
Un exemple

On prend le lexique suivant: (every, = $nd - k$), (language, n), (some, = $nd - k$), (linguist, n), (speaks, = $d\bar{+}k = dv$), (ϵ , = $v + kc$).

(language, n) (every, = $nd - k$)

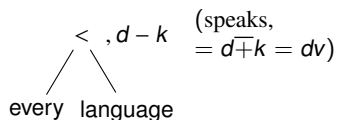
Un exemple

On prend le lexique suivant: (every, = $nd - k$), (language, n), (some, = $nd - k$), (linguist, n), (speaks, = $d+k = dv$), (ϵ , = $v + kc$).



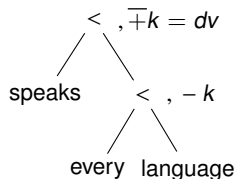
Un exemple

On prend le lexique suivant: (every, = $nd - k$), (language, n), (some, = $nd - k$), (linguist, n), (speaks, = $d+k = dv$), (ϵ , = $v + kc$).



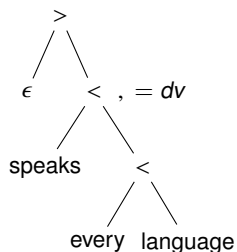
Un exemple

On prend le lexique suivant: (every, = $nd - k$), (language, n), (some, = $nd - k$), (linguist, n), (speaks, = $d\bar{+}k = dv$), (ϵ , = $v + kc$).



Un exemple

On prend le lexique suivant: (every, = $nd - k$), (language, n), (some, = $nd - k$), (linguist, n), (speaks, = $d^+k = dv$), (ϵ , = $v + kc$).

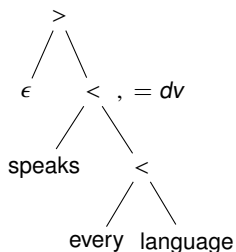


Un exemple

On prend le lexique suivant: (every, = $nd - k$), (language, n), (some, = $nd - k$), (linguist, n), (speaks, = $d+k = dv$), (ϵ , = $v + kc$).

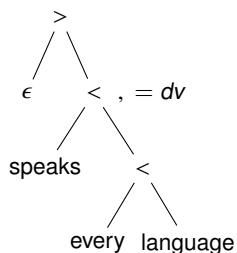
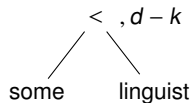
(linguist, n)

(some,
= $nd - k$)



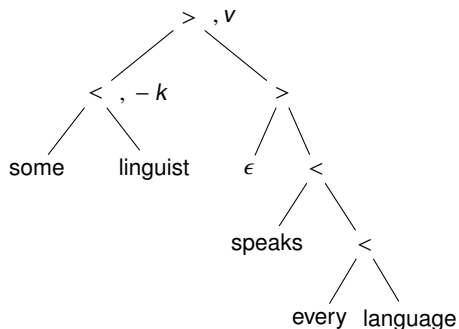
Un exemple

On prend le lexique suivant: (every, = $nd - k$), (language, n), (some, = $nd - k$), (linguist, n), (speaks, = $d+k = dv$), (ϵ , = $v + kc$).



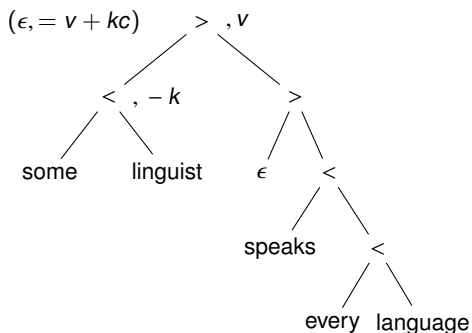
Un exemple

On prend le lexique suivant: (every, = $nd - k$), (language, n), (some, = $nd - k$), (linguist, n), (speaks, = $d^+k = dv$), (ϵ , = $v + kc$).



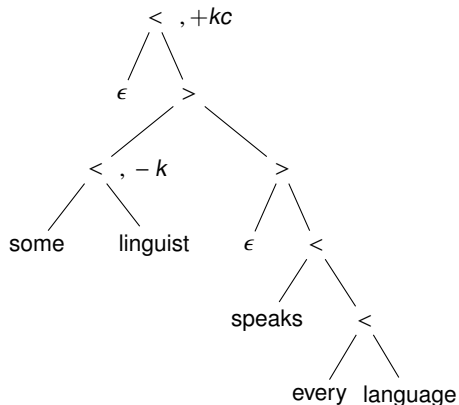
Un exemple

On prend le lexique suivant: (every, = $nd - k$), (language, n), (some, = $nd - k$), (linguist, n), (speaks, = $d + k = dv$), (ϵ , = $v + kc$).



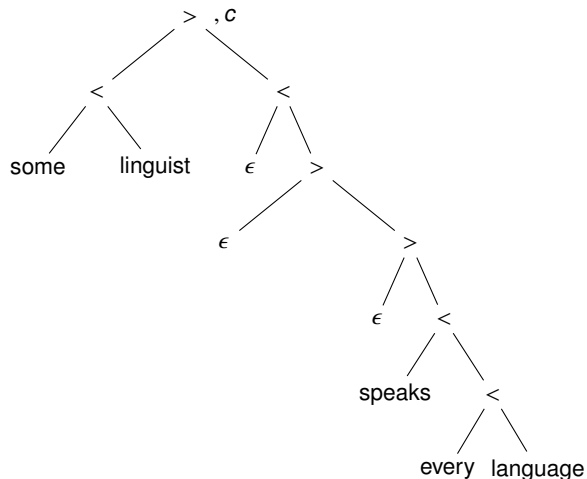
Un exemple

On prend le lexique suivant: (every, = $nd - k$), (language, n), (some, = $nd - k$), (linguist, n), (speaks, = $d + k = dv$), (ϵ , = $v + kc$).



Un exemple

On prend le lexique suivant: (every, = $nd - k$), (language, n), (some, = $nd - k$), (linguist, n), (speaks, = $d^+k = dv$), (ϵ , = $v + kc$).



Les dérivations minimalistes

Les dérivations minimalistes présentent plusieurs difficultés formelles:

- ▶ elles ne représentent pas l'analyse d'une unique phrase (l'interprétation de **move** n'est pas déterministe),
- ▶ l'interprétation sémantique ne peut se faire indépendamment de l'interprétation syntaxique,
- ▶ la correction d'une dérivation est définie de manière algorithmique.

Les dérivations minimalistes

Les dérivations minimalistes présentent plusieurs difficultés formelles:

- ▶ elles ne représentent pas l'analyse d'une unique phrase (l'interprétation de **move** n'est pas déterministe),
- ▶ l'interprétation sémantique ne peut se faire indépendamment de l'interprétation syntaxique,
- ▶ la correction d'une dérivation est définie de manière algorithmique.

Quelle est la nature des dérivations minimalistes ?

Les dérivations minimalistes

Les dérivations minimalistes présentent plusieurs difficultés formelles:

- ▶ elles ne représentent pas l'analyse d'une unique phrase (l'interprétation de **move** n'est pas déterministe),
- ▶ l'interprétation sémantique ne peut se faire indépendamment de l'interprétation syntaxique,
- ▶ la correction d'une dérivation est définie de manière algorithmique.

Quelle est la nature des dérivations minimalistes ?

Il y a eu plusieurs tentatives pour la définir à partir de représentations logiques: (Stabler 97), (Vermaat 99), (Lecomte, Retoré 99,01) (Lecomte 01,03,04,05), (Anoun, Lecomte 06), (Amblard Lecomte 06), (Amblard 07).

Les dérivations minimalistes

Les dérivations minimalistes présentent plusieurs difficultés formelles:

- ▶ elles ne représentent pas l'analyse d'une unique phrase (l'interprétation de **move** n'est pas déterministe),
- ▶ l'interprétation sémantique ne peut se faire indépendamment de l'interprétation syntaxique,
- ▶ la correction d'une dérivation est définie de manière algorithmique.

Quelle est la nature des dérivations minimalistes ?

Il y a eu plusieurs tentatives pour la définir à partir de représentations logiques: (Stabler 97), (Vermaat 99), (Lecomte, Retoré 99,01) (Lecomte 01,03,04,05), (Anoun, Lecomte 06), (Amblard Lecomte 06), (Amblard 07).

Mais aucune ne donne de justification vraiment satisfaisante.

Les dérivations minimalistes

Les dérivations minimalistes présentent plusieurs difficultés formelles:

- ▶ elles ne représentent pas l'analyse d'une unique phrase (l'interprétation de **move** n'est pas déterministe),
- ▶ l'interprétation sémantique ne peut se faire indépendamment de l'interprétation syntaxique,
- ▶ la correction d'une dérivation est définie de manière algorithmique.

Quelle est la nature des dérivations minimalistes ?

Il y a eu plusieurs tentatives pour la définir à partir de représentations logiques: (Stabler 97), (Vermaat 99), (Lecomte, Retoré 99,01) (Lecomte 01,03,04,05), (Anoun, Lecomte 06), (Amblard Lecomte 06), (Amblard 07).

Mais aucune ne donne de justification vraiment satisfaisante.

La théorie formelle des langages (KRS 07) apporte une réponse pour le cas particulier de la **SMC**.

Vérifier que le langage est vide et construire une analyse.

Nous montrons tout d'abord que vérifier la vacuité du langage d'une grammaire minimaliste est équivalent (pour la décidabilité) au problème de vérifier si une phrase appartient au langage défini par une grammaire minimaliste.

Vérifier que le langage est vide et construire une analyse.

Nous montrons tout d'abord que vérifier la vacuité du langage d'une grammaire minimaliste est équivalent (pour la décidabilité) au problème de vérifier si une phrase appartient au langage défini par une grammaire minimaliste.

Théorème: Savoir décider si un mot appartient au langage défini par une grammaire minimaliste implique savoir décider si une grammaire minimaliste a un langage vide.

Vérifier que le langage est vide et construire une analyse.

Nous montrons tout d'abord que vérifier la vacuité du langage d'une grammaire minimaliste est équivalent (pour la décidabilité) au problème de vérifier si une phrase appartient au langage défini par une grammaire minimaliste.

Théorème: Savoir décider si un mot appartient au langage défini par une grammaire minimaliste implique savoir décider si une grammaire minimaliste a un langage vide.

Théorème: Soit G_{MG} une grammaire minimaliste et REG un ensemble régulier, il existe une grammaire G'_{MG} dont le langage est l'intersection de celui de G_{MG} et de REG .

Vérifier que le langage est vide et construire une analyse.

Nous montrons tout d'abord que vérifier la vacuité du langage d'une grammaire minimaliste est équivalent (pour la décidabilité) au problème de vérifier si une phrase appartient au langage défini par une grammaire minimaliste.

Théorème: Savoir décider si un mot appartient au langage défini par une grammaire minimaliste implique savoir décider si une grammaire minimaliste a un langage vide.

Théorème: Soit G_{MG} une grammaire minimaliste et REG un ensemble régulier, il existe une grammaire G'_{MG} dont le langage est l'intersection de celui de G_{MG} et de REG .

Corollaire: Savoir décider si une grammaire minimaliste a un langage vide implique savoir décider si un mot appartient au langage défini par une grammaire minimaliste.

Grammaires minimalistes et VATA

Soit \mathcal{A} un k -VATA sous forme normale de configuration acceptante $(q_f, \mathbf{0})$.

On représente ces règles par les entrées d'une grammaire minimaliste de catégorie acceptante q_f :

- ▶ $(\epsilon, q - \mathbf{i})$ si $f \rightarrow (q, \mathbf{e}_i)$ est une règle de \mathcal{A} ,
- ▶ $(\epsilon, = q_0 + \mathbf{i}q)$ si $f((q_0, \mathbf{x}_0)) \rightarrow (q, \mathbf{x}_0 - \mathbf{e}_i)$ est une règle de \mathcal{A}
- ▶ $(\epsilon, = q_0 = q_1q)$ si $f((q_0, \mathbf{x}_0), (q_1, \mathbf{x}_1)) \rightarrow (q, \mathbf{x}_0 + \mathbf{x}_1)$ est une règle de \mathcal{A}

Grammaires minimalistes et VATA

Soit \mathcal{A} un k -VATA sous forme normale de configuration acceptante $(q_f, \mathbf{0})$.

On représente ces règles par les entrées d'une grammaire minimaliste de catégorie acceptante q_f :

- ▶ $(\epsilon, q - \mathbf{i})$ si $f \rightarrow (q, \mathbf{e}_i)$ est une règle de \mathcal{A} ,
- ▶ $(\epsilon, = q_0 + \mathbf{i}q)$ si $f((q_0, \mathbf{x}_0)) \rightarrow (q, \mathbf{x}_0 - \mathbf{e}_i)$ est une règle de \mathcal{A}
- ▶ $(\epsilon, = q_0 = q_1q)$ si $f((q_0, \mathbf{x}_0), (q_1, \mathbf{x}_1)) \rightarrow (q, \mathbf{x}_0 + \mathbf{x}_1)$ est une règle de \mathcal{A}

Théorème: le langage de la grammaire minimaliste ainsi définie est vide ssi le langage de \mathcal{B} est vide.

Grammaires minimalistes et VATA

Soit \mathcal{A} un k -VATA sous forme normale de configuration acceptante $(q_f, \mathbf{0})$.

On représente ces règles par les entrées d'une grammaire minimaliste de catégorie acceptante q_f :

- ▶ $(\epsilon, q - \mathbf{i})$ si $f \rightarrow (q, \mathbf{e}_i)$ est une règle de \mathcal{A} ,
- ▶ $(\epsilon, = q_0 + \mathbf{i}q)$ si $f((q_0, \mathbf{x}_0)) \rightarrow (q, \mathbf{x}_0 - \mathbf{e}_i)$ est une règle de \mathcal{A}
- ▶ $(\epsilon, = q_0 = q_1 q)$ si $f((q_0, \mathbf{x}_0), (q_1, \mathbf{x}_1)) \rightarrow (q, \mathbf{x}_0 + \mathbf{x}_1)$ est une règle de \mathcal{A}

Théorème: le langage de la grammaire minimaliste ainsi définie est vide ssi le langage de \mathcal{B} est vide.

Corollaire: On peut définir une grammaire minimaliste dont le langage n'est pas semi-linéaire (ce qui infirme la conjecture de Michaelis 07.)

Les listes de traits et les types

Pour une grammaire minimaliste $G_{MG} = \langle B, F, W, \mathcal{L}, c \rangle$, on définit un ensemble de types:

- ▶ $d(I)$ est un type qui représente les dérivations pour lesquelles il reste la liste de traits I à utiliser,

Les listes de traits et les types

Pour une grammaire minimaliste $G_{MG} = \langle B, F, W, \mathcal{L}, c \rangle$, on définit un ensemble de types:

- ▶ $d(I)$ est un type qui représente les dérivations pour lesquelles il reste la liste de traits I à utiliser,
- ▶ $e(I)$ est le type d'une entrée lexicale associée à la liste de traits I ,

Les listes de traits et les types

Pour une grammaire minimaliste $G_{MG} = \langle B, F, W, \mathcal{L}, c \rangle$, on définit un ensemble de types:

- ▶ $d(I)$ est un type qui représente les dérivations pour lesquelles il reste la liste de traits I à utiliser,
- ▶ $e(I)$ est le type d'une entrée lexicale associée à la liste de traits I ,
- ▶ $h(I)$ est le type d'un élément qui doit être déplacé en utilisant les traits de mouvement de la liste I .

Les listes de traits et les types

Pour une grammaire minimaliste $G_{MG} = \langle B, F, W, \mathcal{L}, c \rangle$, on définit un ensemble de types:

- ▶ $d(I)$ est un type qui représente les dérivations pour lesquelles il reste la liste de traits I à utiliser,
- ▶ $e(I)$ est le type d'une entrée lexicale associée à la liste de traits I ,
- ▶ $h(I)$ est le type d'un élément qui doit être déplacé en utilisant les traits de mouvement de la liste I .

On peut énumérer tous les types nécessaires en prenant I dans $\mathbb{L}_{G_{MG}}$.

merge et move comme opérateurs linéaires

Nous définissons pour $G_{MG} = \langle B, F, W, \mathcal{L}, c \rangle$ l'ensemble des constantes logiques suivantes:

- ▶ $(w, l) : e(l)$ si (w, l) est dans \mathcal{L}

merge et move comme opérateurs linéaires

Nous définissons pour $G_{MG} = \langle B, F, W, \mathcal{L}, c \rangle$ l'ensemble des constantes logiques suivantes:

- ▶ $(w, l) : e(l)$ si (w, l) est dans \mathcal{L}
- ▶ $merge_1 : [d(a_1)|e(a_1)] \multimap e(= a_2) \multimap h(l_1) \multimap d(l_2)$

merge et move comme opérateurs linéaires

Nous définissons pour $G_{MG} = \langle B, F, W, \mathcal{L}, c \rangle$ l'ensemble des constantes logiques suivantes:

- ▶ $(w, l) : e(l)$ si (w, l) est dans \mathcal{L}
- ▶ $merge_1 : [d(al_1)|e(al_1)] \multimap e(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_2 : [d(al_1)|e(al_1)] \multimap d(= al_2) \multimap h(l_1) \multimap d(l_2)$

merge et move comme opérateurs linéaires

Nous définissons pour $G_{MG} = \langle B, F, W, \mathcal{L}, c \rangle$ l'ensemble des constantes logiques suivantes:

- ▶ $(w, l) : e(l)$ si (w, l) est dans \mathcal{L}
- ▶ $merge_1 : [d(al_1)|e(al_1)] \multimap e(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_2 : [d(al_1)|e(al_1)] \multimap d(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_3 : [d(a)|e(a)] \multimap e(= al_2) \multimap d(l_2)$

merge et move comme opérateurs linéaires

Nous définissons pour $G_{MG} = \langle B, F, W, \mathcal{L}, c \rangle$ l'ensemble des constantes logiques suivantes:

- ▶ $(w, l) : e(l)$ si (w, l) est dans \mathcal{L}
- ▶ $merge_1 : [d(al_1)|e(al_1)] \multimap e(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_2 : [d(al_1)|e(al_1)] \multimap d(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_3 : [d(a)|e(a)] \multimap e(= al_2) \multimap d(l_2)$
- ▶ $merge_4 : [d(a)|e(a)] \multimap d(= al_2) \multimap d(l_2)$

merge et move comme opérateurs linéaires

Nous définissons pour $G_{MG} = \langle B, F, W, \mathcal{L}, c \rangle$ l'ensemble des constantes logiques suivantes:

- ▶ $(w, l) : e(l)$ si (w, l) est dans \mathcal{L}
- ▶ $merge_1 : [d(al_1)|e(al_1)] \multimap e(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_2 : [d(al_1)|e(al_1)] \multimap d(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_3 : [d(a)|e(a)] \multimap e(= al_2) \multimap d(l_2)$
- ▶ $merge_4 : [d(a)|e(a)] \multimap d(= al_2) \multimap d(l_2)$
- ▶ $move_1 : (h(-al_1) \multimap d(+al_2)) \multimap h(l_1) \multimap d(l_2)$

merge et move comme opérateurs linéaires

Nous définissons pour $G_{MG} = \langle B, F, W, \mathcal{L}, c \rangle$ l'ensemble des constantes logiques suivantes:

- ▶ $(w, l) : e(l)$ si (w, l) est dans \mathcal{L}
- ▶ $merge_1 : [d(al_1)|e(al_1)] \multimap e(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_2 : [d(al_1)|e(al_1)] \multimap d(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_3 : [d(a)|e(a)] \multimap e(= al_2) \multimap d(l_2)$
- ▶ $merge_4 : [d(a)|e(a)] \multimap d(= al_2) \multimap d(l_2)$
- ▶ $move_1 : (h(-al_1) \multimap d(+al_2)) \multimap h(l_1) \multimap d(l_2)$
- ▶ $move'_1 : (h(-a) \multimap d(+al)) \multimap d(l)$

merge et move comme opérateurs linéaires

Nous définissons pour $G_{MG} = \langle B, F, W, \mathcal{L}, c \rangle$ l'ensemble des constantes logiques suivantes:

- ▶ $(w, l) : e(l)$ si (w, l) est dans \mathcal{L}
- ▶ $merge_1 : [d(al_1)|e(al_1)] \multimap e(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_2 : [d(al_1)|e(al_1)] \multimap d(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_3 : [d(a)|e(a)] \multimap e(= al_2) \multimap d(l_2)$
- ▶ $merge_4 : [d(a)|e(a)] \multimap d(= al_2) \multimap d(l_2)$
- ▶ $move_1 : (h(-al_1) \multimap d(+al_2)) \multimap h(l_1) \multimap d(l_2)$
- ▶ $move'_1 : (h(-a) \multimap d(+al)) \multimap d(l)$
- ▶ $move_2 : (h(-al_1) \multimap d(\overline{+}al_2)) \multimap h(l_1) \multimap d(l_2)$

merge et move comme opérateurs linéaires

Nous définissons pour $G_{MG} = \langle B, F, W, \mathcal{L}, c \rangle$ l'ensemble des constantes logiques suivantes:

- ▶ $(w, l) : e(l)$ si (w, l) est dans \mathcal{L}
- ▶ $merge_1 : [d(al_1)|e(al_1)] \multimap e(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_2 : [d(al_1)|e(al_1)] \multimap d(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_3 : [d(a)|e(a)] \multimap e(= al_2) \multimap d(l_2)$
- ▶ $merge_4 : [d(a)|e(a)] \multimap d(= al_2) \multimap d(l_2)$
- ▶ $move_1 : (h(-al_1) \multimap d(+al_2)) \multimap h(l_1) \multimap d(l_2)$
- ▶ $move'_1 : (h(-a) \multimap d(+al)) \multimap d(l)$
- ▶ $move_2 : (h(-al_1) \multimap d(\overline{+}al_2)) \multimap h(l_1) \multimap d(l_2)$
- ▶ $move'_2 : (h(-a) \multimap d(\overline{+}al)) \multimap d(l)$

merge et move comme opérateurs linéaires

Nous définissons pour $G_{MG} = \langle B, F, W, \mathcal{L}, c \rangle$ l'ensemble des constantes logiques suivantes:

- ▶ $(w, l) : e(l)$ si (w, l) est dans \mathcal{L}
- ▶ $merge_1 : [d(al_1)|e(al_1)] \multimap e(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_2 : [d(al_1)|e(al_1)] \multimap d(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_3 : [d(a)|e(a)] \multimap e(= al_2) \multimap d(l_2)$
- ▶ $merge_4 : [d(a)|e(a)] \multimap d(= al_2) \multimap d(l_2)$
- ▶ $move_1 : (h(-al_1) \multimap d(+al_2)) \multimap h(l_1) \multimap d(l_2)$
- ▶ $move'_1 : (h(-a) \multimap d(+al)) \multimap d(l)$
- ▶ $move_2 : (h(-al_1) \multimap d(\overline{+}al_2)) \multimap h(l_1) \multimap d(l_2)$
- ▶ $move'_2 : (h(-a) \multimap d(\overline{+}al)) \multimap d(l)$

Théorème: il existe une dérivation correcte pour G_{MG} si et seulement si il existe un λ -terme clos de type $d(c)$ construit avec ces constantes.

merge et move comme opérateurs linéaires

Nous définissons pour $G_{MG} = \langle B, F, W, \mathcal{L}, c \rangle$ l'ensemble des constantes logiques suivantes:

- ▶ $(w, l) : e(l)$ si (w, l) est dans \mathcal{L}
- ▶ $merge_1 : [d(al_1)|e(al_1)] \multimap e(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_2 : [d(al_1)|e(al_1)] \multimap d(= al_2) \multimap h(l_1) \multimap d(l_2)$
- ▶ $merge_3 : [d(a)|e(a)] \multimap e(= al_2) \multimap d(l_2)$
- ▶ $merge_4 : [d(a)|e(a)] \multimap d(= al_2) \multimap d(l_2)$
- ▶ $move_1 : (h(-al_1) \multimap d(+al_2)) \multimap h(l_1) \multimap d(l_2)$
- ▶ $move'_1 : (h(-a) \multimap d(+al)) \multimap d(l)$
- ▶ $move_2 : (h(-al_1) \multimap d(\overline{+}al_2)) \multimap h(l_1) \multimap d(l_2)$
- ▶ $move'_2 : (h(-a) \multimap d(\overline{+}al)) \multimap d(l)$

Théorème: il existe une dérivation correcte pour G_{MG} si et seulement si il existe un λ -terme clos de type $d(c)$ construit avec ces constantes.

Corollaire: Savoir décider si un séquent est prouvable dans **MELL** implique savoir décider de la vacuité du langage d'une grammaire minimaliste. (c.f. (dGGS 04)).

Quelques remarques.

- ▶ Pour Stabler une des bonnes propriétés des grammaires minimalistes est qu'elles donnent des descriptions concises. Ici, on constate que cela est dû à une quantification universelle sur les types des opérateurs.

Quelques remarques.

- ▶ Pour Stabler une des bonnes propriétés des grammaires minimalistes est qu'elles donnent des descriptions concises. Ici, on constate que cela est dû à une quantification universelle sur les types des opérateurs.
- ▶ On peut énumérer tous les opérateurs avec des types sans quantificateurs pour une grammaire G_{MG} et on obtient un nombre quadratique d'opérateurs en fonction de la taille de $\mathbb{L}_{G_{MG}}$.

Quelques remarques.

- ▶ Pour Stabler une des bonnes propriétés des grammaires minimalistes est qu'elles donnent des descriptions concises. Ici, on constate que cela est dû à une quantification universelle sur les types des opérateurs.
- ▶ On peut énumérer tous les opérateurs avec des types sans quantificateurs pour une grammaire G_{MG} et on obtient un nombre quadratique d'opérateurs en fonction de la taille de $\mathbb{L}_{G_{MG}}$.
- ▶ Les traits ne sont pas utilisés comme des ressources, c'est la gestion des types qui produit l'effet qu'ils sont utilisés une et une seule fois.

Quelques remarques.

- ▶ Pour Stabler une des bonnes propriétés des grammaires minimalistes est qu'elles donnent des descriptions concises. Ici, on constate que cela est dû à une quantification universelle sur les types des opérateurs.
- ▶ On peut énumérer tous les opérateurs avec des types sans quantificateurs pour une grammaire G_{MG} et on obtient un nombre quadratique d'opérateurs en fonction de la taille de $\mathbb{L}_{G_{MG}}$.
- ▶ Les traits ne sont pas utilisés comme des ressources, c'est la gestion des types qui produit l'effet qu'ils sont utilisés une et une seule fois.
- ▶ Cette approche permet de rajouter de nouveaux opérateurs permettant l'extension des grammaires minimalistes.

Conclusion

- ▶ Nous avons introduit les VATA, une extension naturelle des VAS, dont la décidabilité du problème d'accessibilité est équivalente à celle de MELL.
- ▶ Les VATA nous ont permis de montrer que l'analyse dans les grammaires minimalistes de Stabler est aussi difficile que la recherche de démonstration dans MELL.
- ▶ Cela nous donne de bons arguments pour représenter les structures profondes des grammaires minimalistes par des démonstrations dans MELL.