λ-calcul et théorie des types

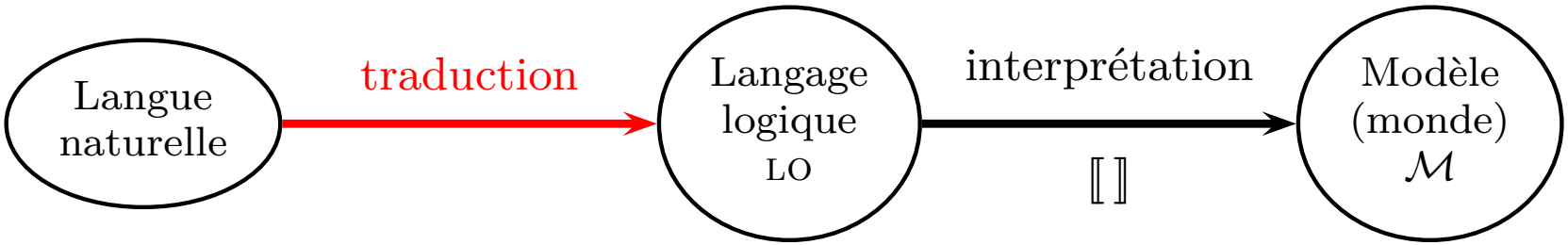
Vers une interface syntaxe-sémantique

Laurent Roussarie

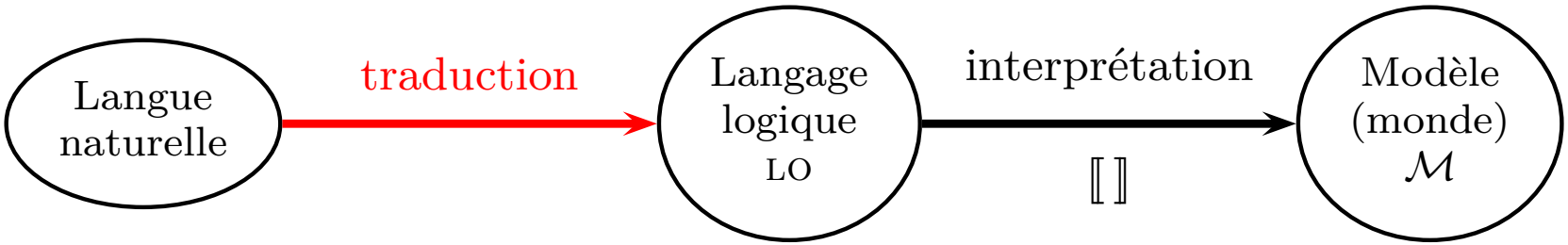
Sémantique, M1 LTD

2014

Introduction



Introduction



- **Objectif :** Se donner une système des règles qui permette d'obtenir "automatiquement" la représentation sémantique d'une phrase tout en restant compositionnel

Construction du sens

Décorer les arbres

Principe de compositionnalité

Le sens d'une expression est une fonction du sens de ses parties et de sa structure syntaxique.

☞ Notre système de règles va opérer sur des analyses syntaxique, c'est-à-dire des arbres.

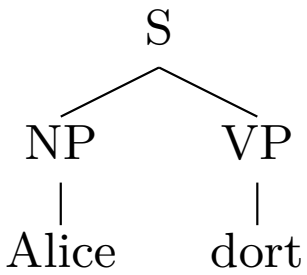
Construction du sens

Décorer les arbres

Principe de compositionnalité

Le sens d'une expression est une fonction du sens de ses parties et de sa structure syntaxique.

☞ Notre système de règles va opérer sur des analyses syntaxique, c'est-à-dire des arbres.



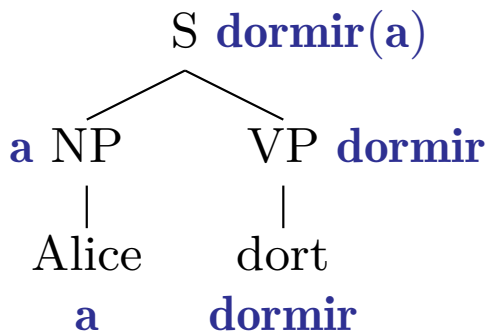
Construction du sens

Décorer les arbres

Principe de compositionnalité

Le sens d'une expression est une fonction du sens de ses parties et de sa structure syntaxique.

☞ Notre système de règles va opérer sur des analyses syntaxique, c'est-à-dire des arbres.

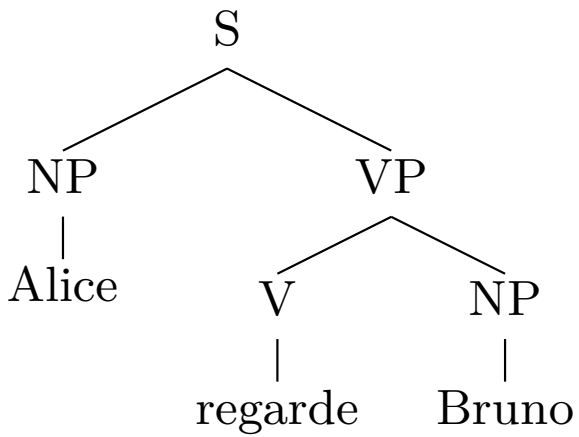


Enjeu : Comment passer "mécaniquement" des ingrédients (nœuds) au plat final (racine) ?

Construction du sens

Expressions non saturées

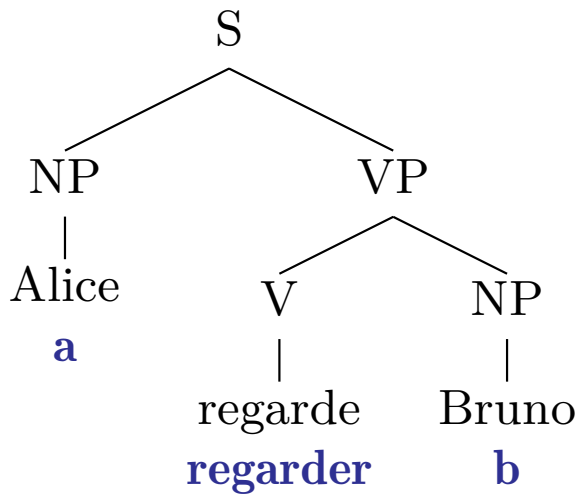
Par compositionnalité, tout constituant interprétable doit avoir une représentation sémantique valide.



Construction du sens

Expressions non saturées

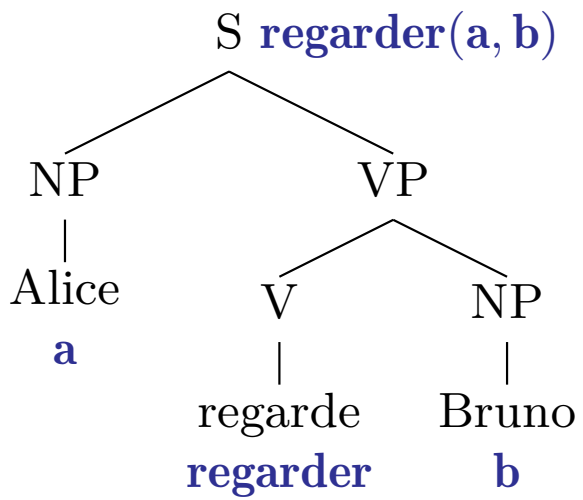
Par compositionnalité, tout constituant interprétable doit avoir une représentation sémantique valide.



Construction du sens

Expressions non saturées

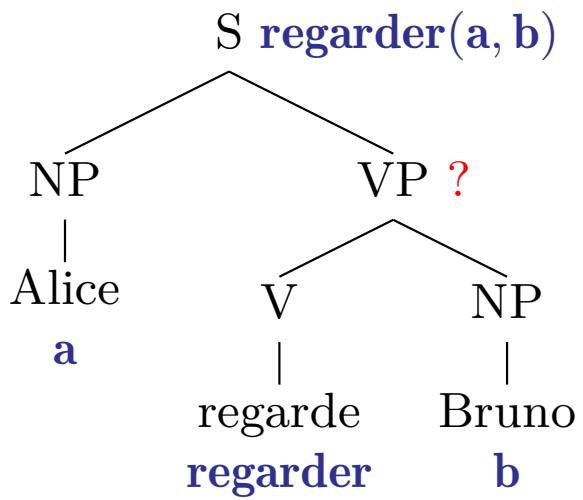
Par compositionnalité, tout constituant interprétable doit avoir une représentation sémantique valide.



Construction du sens

Expressions non saturées

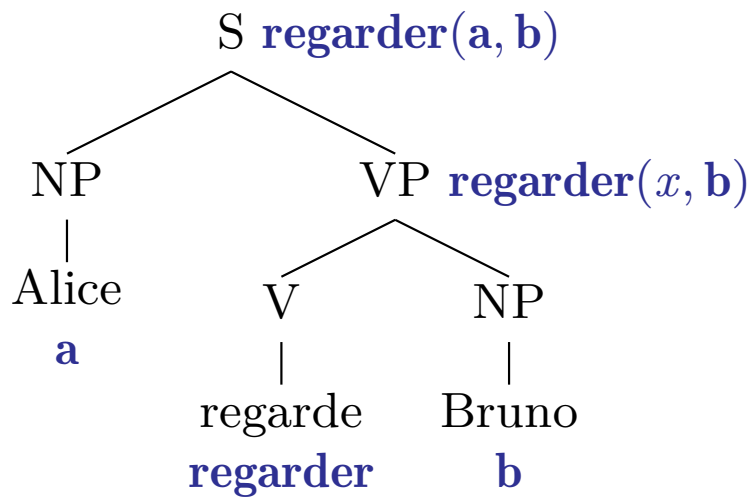
Par compositionnalité, tout constituant interprétable doit avoir une représentation sémantique valide.



Construction du sens

Expressions non saturées

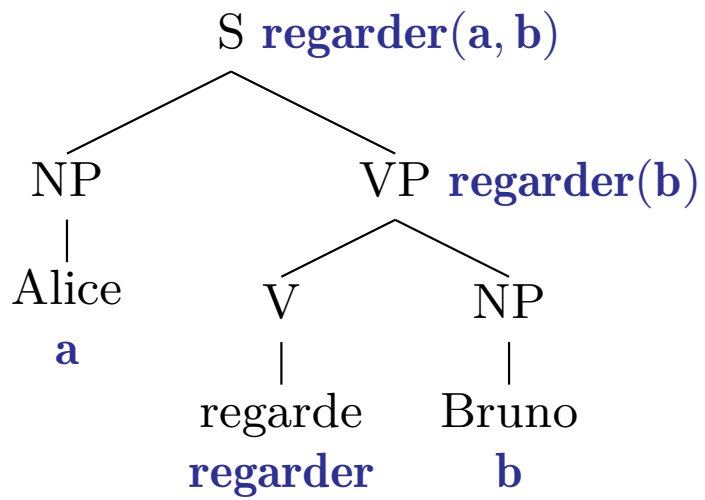
Par compositionnalité, tout constituant interprétable doit avoir une représentation sémantique valide.



Construction du sens

Expressions non saturées

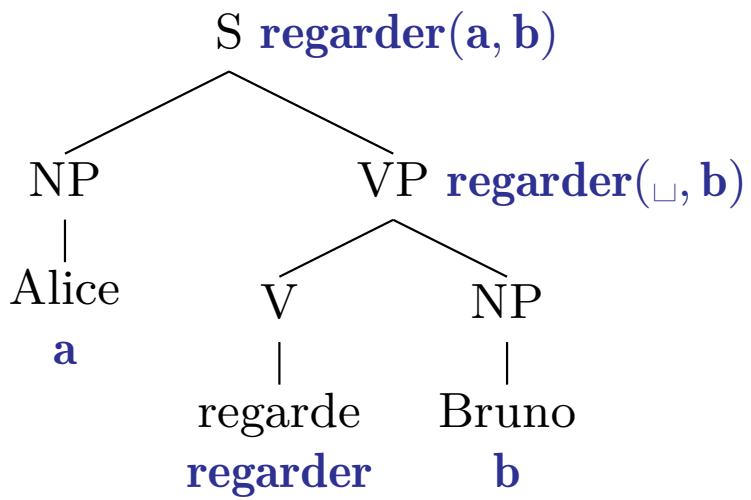
Par compositionnalité, tout constituant interprétable doit avoir une représentation sémantique valide.



Construction du sens

Expressions non saturées

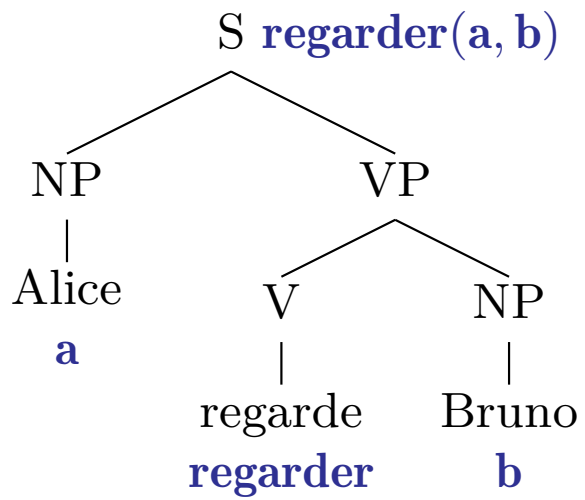
Par compositionnalité, tout constituant interprétable doit avoir une représentation sémantique valide.



Construction du sens

Expressions non saturées

Par compositionnalité, tout constituant interprétable doit avoir une représentation sémantique valide.



Allons voir ce que peut nous dire la sémantique de LO.

Les prédicats dénotent des fonctions

Prédicats à une place

$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) =$ l'**ensemble** de tous les dormeurs de w.

Les prédicats dénotent des fonctions

Prédicats à une place

$$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) = \text{l'}\textbf{ensemble} \text{ de tous les dormeurs de } w.$$

$$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) = \begin{array}{ccc} \mathcal{A} & \longrightarrow & \{0\,;1\} \\ \text{x} & \longmapsto & \begin{cases} 1 \text{ si x dort dans } w \\ 0 \text{ sinon} \end{cases} \end{array}$$

Les prédicats dénotent des fonctions

Prédicats à une place

$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) =$ l'**ensemble** de tous les dormeurs de w.

$$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) = \begin{array}{ccc} \mathcal{A} & \longrightarrow & \{0\ ;\ 1\} \\ \text{x} & \longmapsto & \begin{cases} 1 \text{ si x dort dans } w \\ 0 \text{ sinon} \end{cases} \end{array}$$

$$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = \left[\begin{array}{ll} \textsc{Alice} & \\ \textsc{Bruno} & \longrightarrow 1 \\ \textsc{Charles} & \longrightarrow 0 \\ \textsc{Dina} & \end{array} \right]$$

Les prédicats dénotent des fonctions

Prédicats à une place

$$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) = \text{l'}\textbf{ensemble} \text{ de tous les dormeurs de } w.$$

$$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) = \mathcal{A} \longrightarrow \{0\ ;\ 1\}$$

$$\text{x} \longmapsto \begin{cases} 1 \text{ si x dort dans } w \\ 0 \text{ sinon} \end{cases}$$

$$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = \left[\begin{array}{ll} \textsc{Alice} & \\ \textsc{Bruno} & 1 \\ \textsc{Charles} & \\ \textsc{Dina} & 0 \end{array}\right] \equiv \left\{\begin{array}{l} \textsc{Alice}\ ; \\ \textsc{Bruno}\ ; \\ \textsc{Dina} \end{array}\right\}$$

Les prédicats dénotent des fonctions

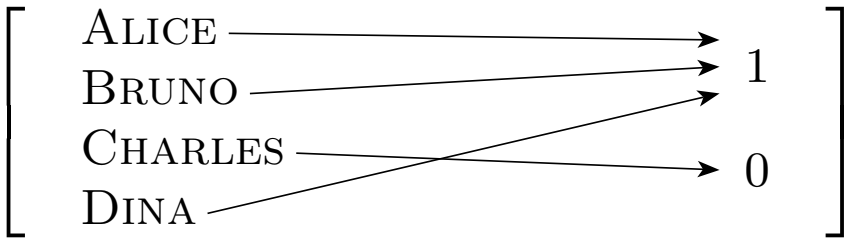
Prédicats à une place

$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) =$ l'**ensemble** de tous les dormeurs de w.

$$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) = \mathcal{A} \longrightarrow \{0\ ;\ 1\}$$

$$\text{x} \longmapsto \begin{cases} 1 \text{ si x dort dans } w \\ 0 \text{ sinon} \end{cases}$$

$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} =$



$[\![\textit{Alice dort}]\!]^{\mathcal{M},w,g} = ?$

Les prédicats dénotent des fonctions

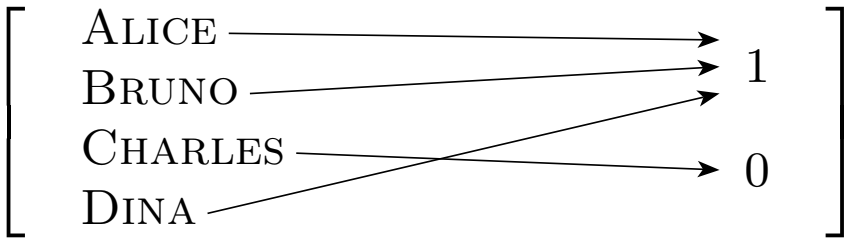
Prédicats à une place

$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) =$ l'**ensemble** de tous les dormeurs de w.

$$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) = \mathcal{A} \longrightarrow \{0\ ;\ 1\}$$

$$\text{x} \longmapsto \begin{cases} 1 \text{ si x dort dans } w \\ 0 \text{ sinon} \end{cases}$$

$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} =$



$[\![\textit{Alice dort}]\!]^{\mathcal{M},w,g} = ?$

$[\![\textbf{dormir(a)}]\!]^{\mathcal{M},w,g} =$

Les prédicats dénotent des fonctions

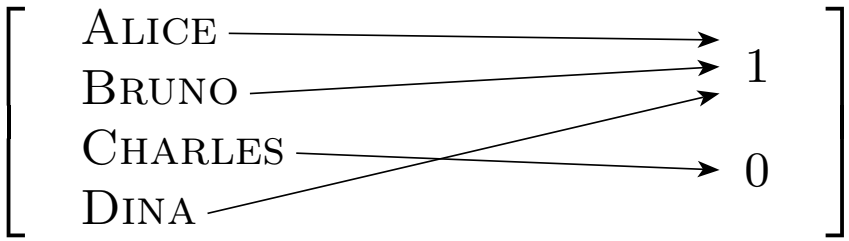
Prédicats à une place

$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) =$ l'**ensemble** de tous les dormeurs de w.

$$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) = \mathcal{A} \longrightarrow \{0\ ;\ 1\}$$

$$\text{x} \longmapsto \begin{cases} 1 \text{ si x dort dans } w \\ 0 \text{ sinon} \end{cases}$$

$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} =$



$[\![\textit{Alice dort}]\!]^{\mathcal{M},w,g} = ?$

$[\![\textbf{dormir(a)}]\!]^{\mathcal{M},w,g} =$

$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g}$

Les prédicats dénotent des fonctions

Prédicats à une place

$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) =$ l'**ensemble** de tous les dormeurs de w.

$$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) = \begin{array}{ccc} \mathcal{A} & \longrightarrow & \{0\ ;\ 1\} \\ \text{x} & \longmapsto & \begin{cases} 1 \text{ si x dort dans } w \\ 0 \text{ sinon} \end{cases} \end{array}$$

$$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = \left[\begin{array}{ll} \textsc{Alice} & \\ \textsc{Bruno} & 1 \\ \textsc{Charles} & \\ \textsc{Dina} & 0 \end{array} \right]$$

$[\![\textit{Alice dort}]\!]^{\mathcal{M},w,g} = ?$

$[\![\textbf{dormir(a)}]\!]^{\mathcal{M},w,g} =$

$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g}([\![\textbf{a}]\!]^{\mathcal{M},w,g})$

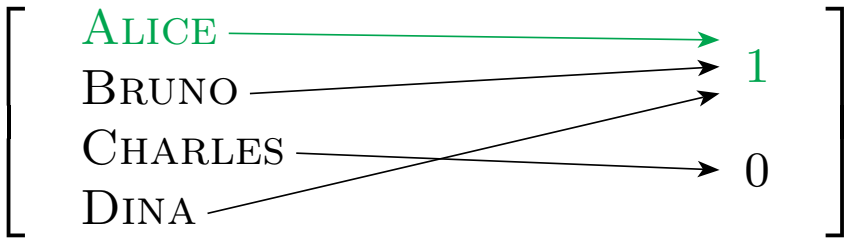
Les prédicats dénotent des fonctions

Prédicats à une place

$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) =$ l'**ensemble** de tous les dormeurs de w.

$$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} = F(w, \textbf{dormir}) = \begin{array}{ccc} \mathcal{A} & \longrightarrow & \{0\ ;\ 1\} \\ \text{x} & \longmapsto & \begin{cases} 1 \text{ si x dort dans } w \\ 0 \text{ sinon} \end{cases} \end{array}$$

$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g} =$



$[\![\textit{Alice dort}]\!]^{\mathcal{M},w,g} = 1$

$[\![\textbf{dormir}(\textbf{a})]\!]^{\mathcal{M},w,g} = 1$

$[\![\textbf{dormir}]\!]^{\mathcal{M},w,g}([\![\textbf{a}]\!]^{\mathcal{M},w,g}) = 1$

Les prédicats dénotent des fonctions

Prédicats à deux arguments

Les prédicats dénotent des fonctions

Prédicats à deux arguments

... regarde Bruno

$$[\![\textbf{regarder}(\sqcup, \textbf{b})]\!]^{\mathcal{M},w,g} = \left[\begin{array}{ll} \textsc{Alice} & \\ \textsc{Bruno} & 1 \\ \textsc{Charles} & \\ \textsc{Dina} & 0 \end{array}\right]$$

Les prédicats dénotent des fonctions

Prédicats à deux arguments

... regarde Alice

$$[\![\textbf{regarder}(\sqcup, \textbf{a})]\!]^{\mathcal{M},w,g} = \left[\begin{array}{ll} \textsc{Alice} & \\ \textsc{Bruno} & 1 \\ \textsc{Charles} & \\ \textsc{Dina} & 0 \end{array}\right]$$

Les prédicats dénotent des fonctions

Prédicats à deux arguments

... regarde Charles

$$[\![\textbf{regarder}(\sqcup, \textbf{c})]\!]^{\mathcal{M},w,g} = \left[\begin{array}{ll} \textsc{Alice} & \\ \textsc{Bruno} & 1 \\ \textsc{Charles} & \\ \textsc{Dina} & 0 \end{array}\right]$$

Les prédicats dénotent des fonctions

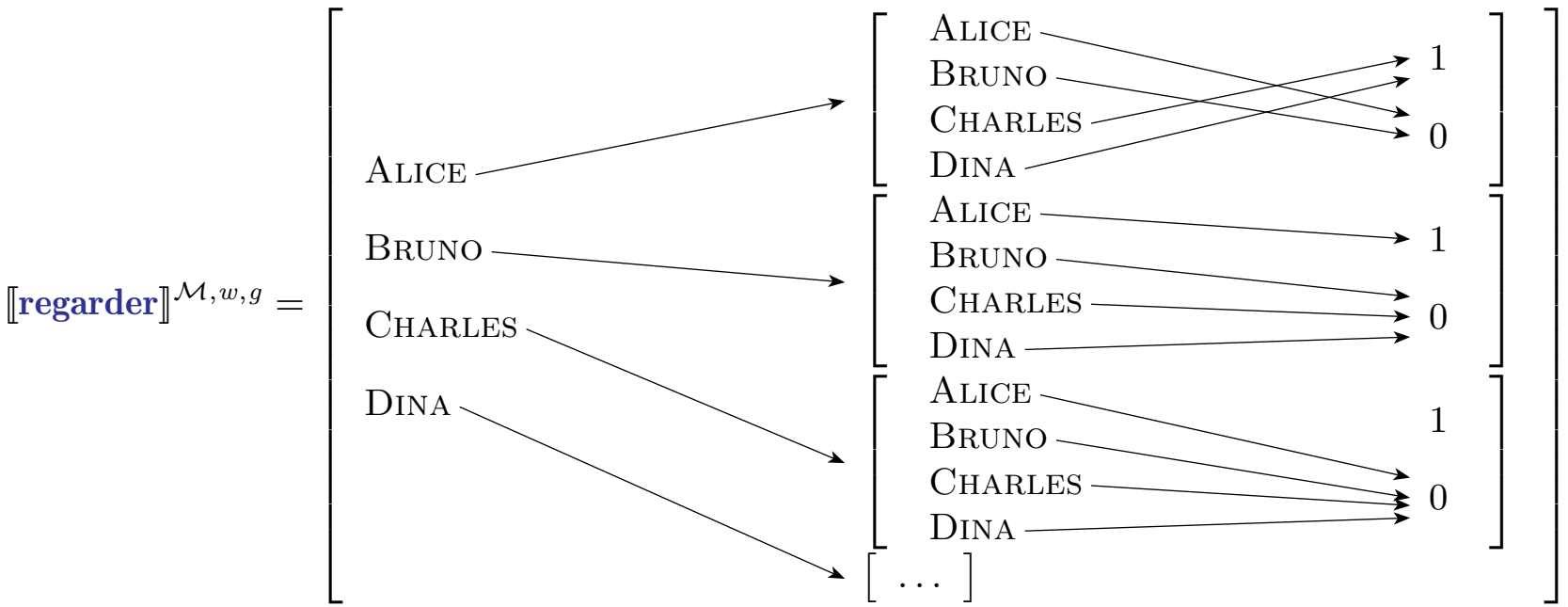
Prédicats à deux arguments

... regarde ...

$$[\![\textbf{regarder}]\!]^{\mathcal{M},w,g} = \left[\begin{array}{ll} \textsc{Alice} & \longrightarrow \text{« qui regarde } \textsc{Alice} \text{ ? »} \\ \textsc{Bruno} & \longrightarrow \text{« qui regarde } \textsc{Bruno} \text{ ? »} \\ \textsc{Charles} & \longrightarrow \text{« qui regarde } \textsc{Charles} \text{ ? »} \\ \textsc{Dina} & \longrightarrow \text{« qui regarde } \textsc{Dina} \text{ ? »} \end{array}\right]$$

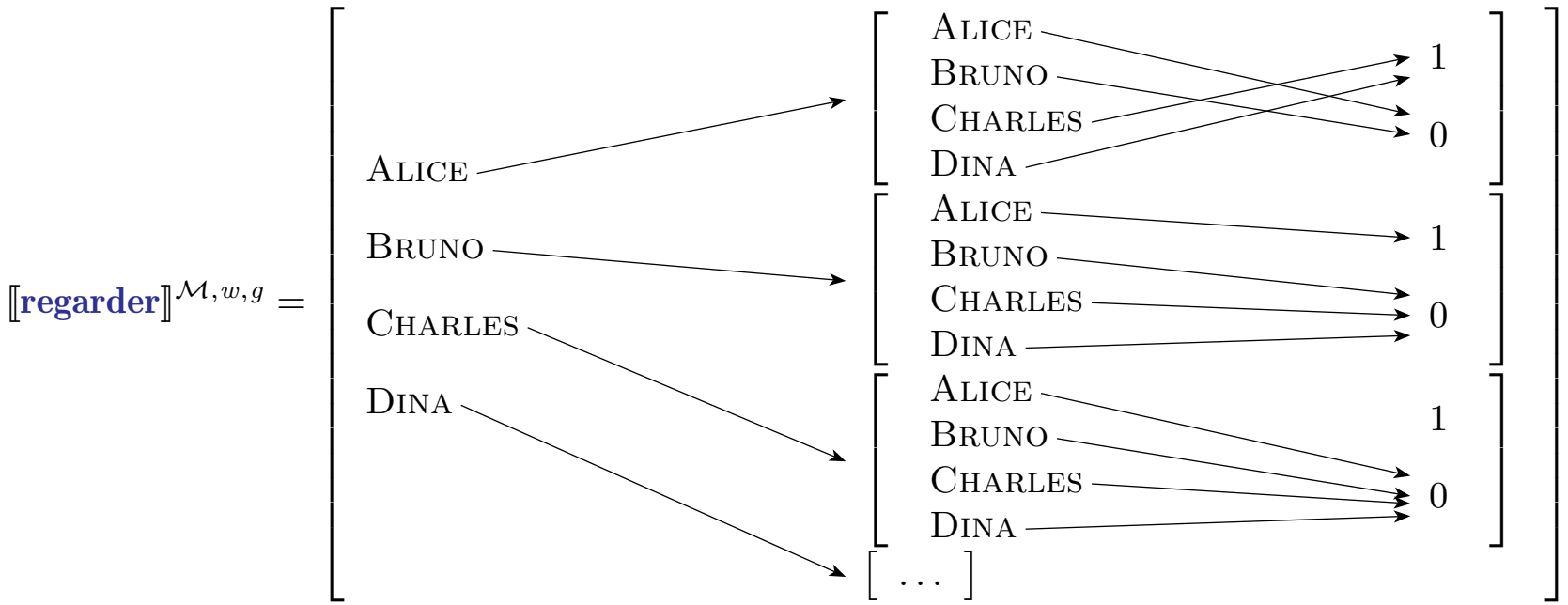
Les prédicats dénotent des fonctions

Prédicats à deux arguments



Les prédicats dénotent des fonctions

Prédicats à deux arguments

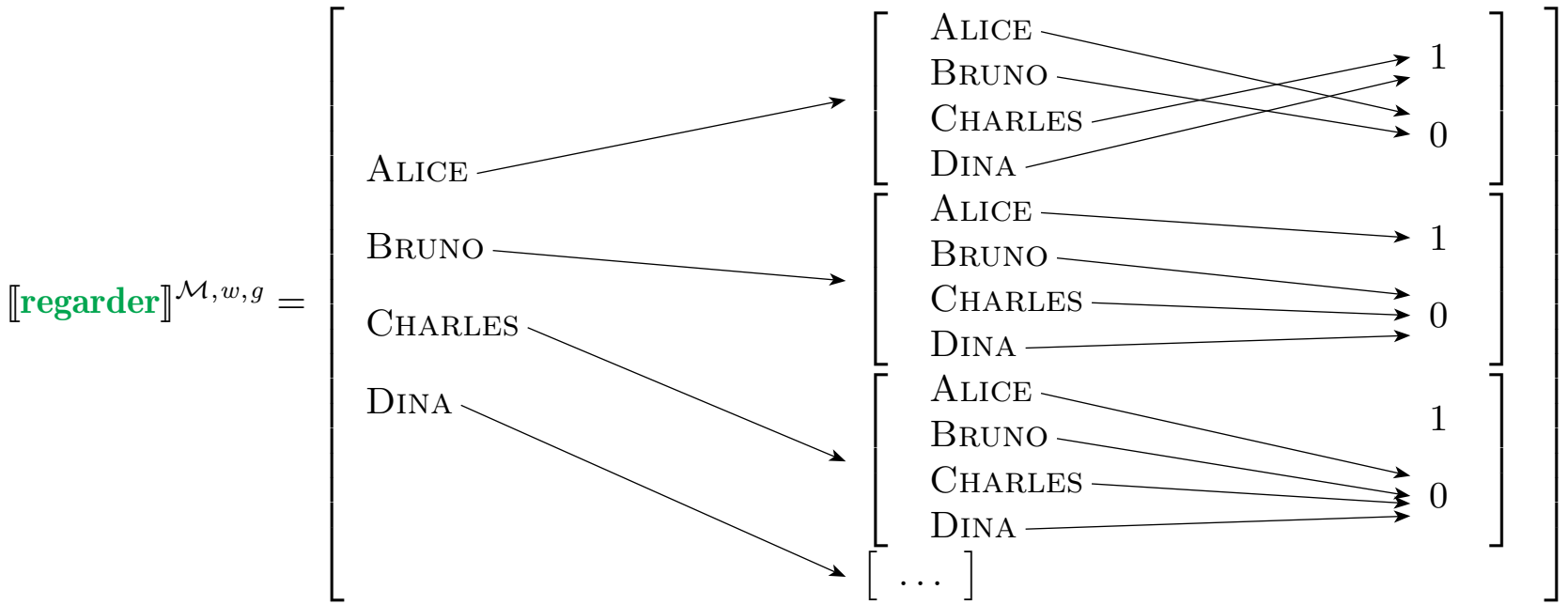


$[\![\textit{Dina regarde Alice}]\!]^{\mathcal{M},w,g} = ?$

$[\![\textbf{regarder}(\textbf{d},\textbf{a})]\!]^{\mathcal{M},w,g} = ?$

Les prédicats dénotent des fonctions

Prédicats à deux arguments



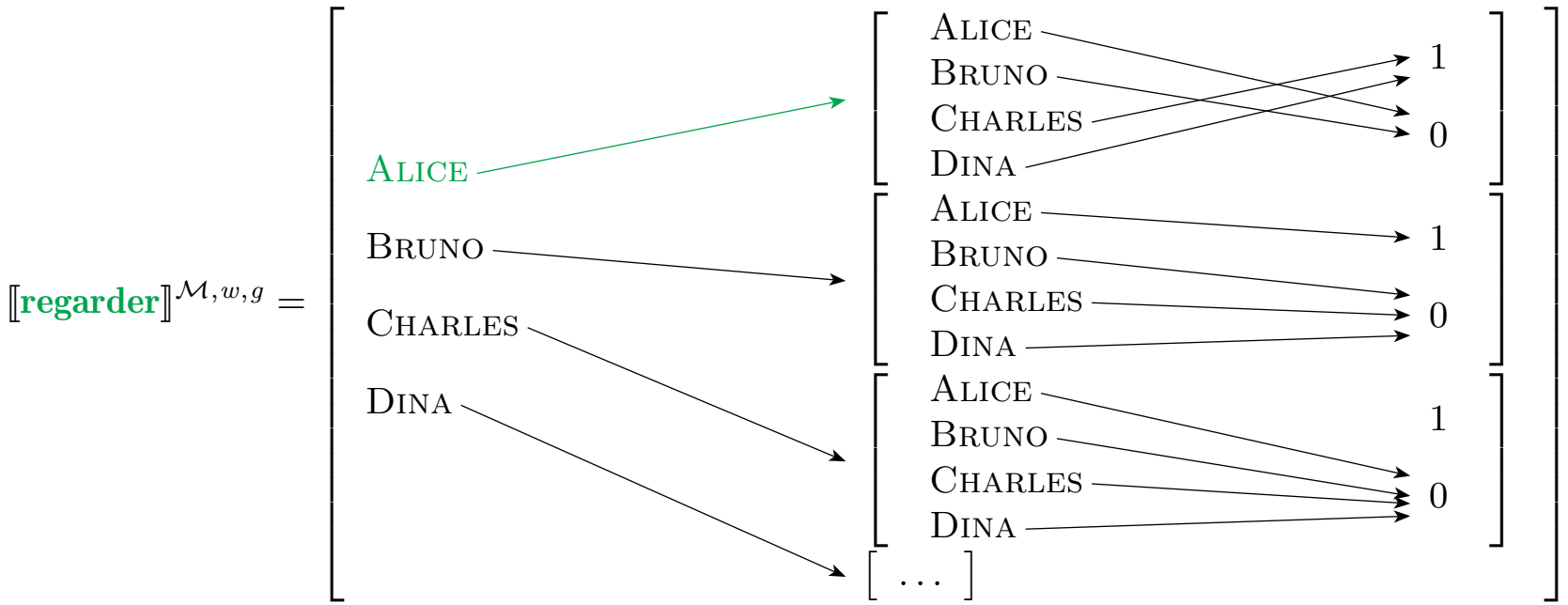
$[\![\textit{Dina regarde Alice}]\!]^{\mathcal{M},w,g} = ?$

$[\![\textbf{regarder}(\textbf{d},\textbf{a})]\!]^{\mathcal{M},w,g} = ?$

$[\![\textbf{regarder}]\!]^{\mathcal{M},w,g}$

Les prédicats dénotent des fonctions

Prédicats à deux arguments



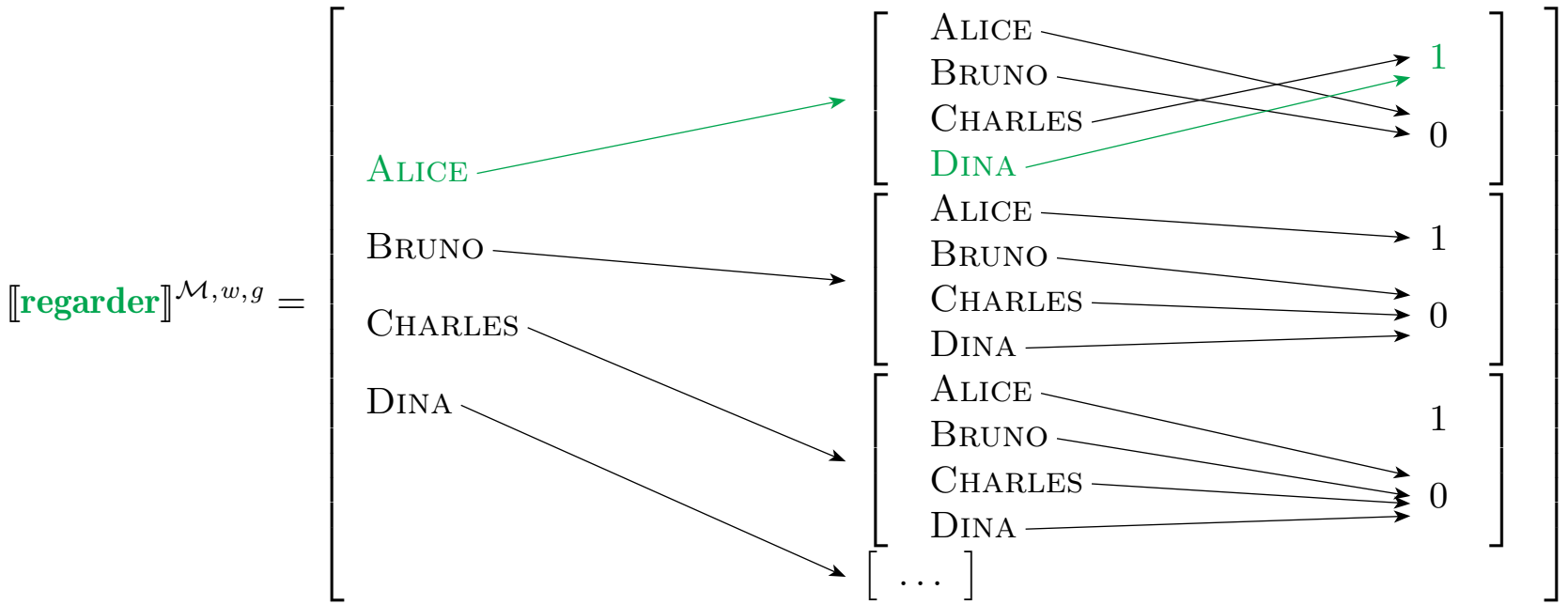
$[\![\textit{Dina regarde Alice}]\!]^{\mathcal{M},w,g} = ?$

$[\![\textbf{regarder}(\textbf{d},\textbf{a})]\!]^{\mathcal{M},w,g} = ?$

$[\![\textbf{regarder}]\!]^{\mathcal{M},w,g}([\![\textbf{a}]\!]^{\mathcal{M},w,g})$

Les prédicats dénotent des fonctions

Prédicats à deux arguments



$[\![\textit{Dina regarde Alice}]\!]^{\mathcal{M},w,g} = 1$

$[\![\textbf{regarder}(\textbf{d},\textbf{a})]\!]^{\mathcal{M},w,g} = 1$

$[\![\textbf{regarder}]\!]^{\mathcal{M},w,g}([\![\textbf{a}]\!]^{\mathcal{M},w,g})([\![\textbf{d}]\!]^{\mathcal{M},w,g}) = 1$

Retour à la syntaxe de LO

Des trous dans les formules

Les prédicats = des formules incomplètes, non saturées

☞ On va écrire des formules “inachevées” en indiquant quels éléments manquent.

$\mathbf{dormir}(x)$	saturée

Retour à la syntaxe de LO

Des trous dans les formules

Les prédicats = des formules incomplètes, non saturées

☞ On va écrire des formules “inachevées” en indiquant quels éléments manquent.

$\lambda x\, \mathbf{dormir}(x)$	désaturée

$\lambda x \approx$ “dans ce qui suit, il manque x”

Retour à la syntaxe de LO

Des trous dans les formules

Les prédicats = des formules incomplètes, non saturées

☞ On va écrire des formules “inachevées” en indiquant quels éléments manquent.

$\lambda x\, \mathbf{dormir}(x)$	désaturée

$\lambda x \approx$ “dans ce qui suit, il manque une valeur pour x”

Retour à la syntaxe de LO

Des trous dans les formules

Les prédicats = des formules incomplètes, non saturées

☞ On va écrire des formules “inachevées” en indiquant quels éléments manquent.

$\lambda x\, \mathbf{dormir}(x)$ désaturée

$\lambda x \approx$ “dans ce qui suit, il manque une valeur pour x”

- λ : opérateur d’abstraction
- λ s’applique à une variable (c’est un lieur)
- On appellera une expression qui commence par λ un λ-terme.

Retour à la syntaxe de LO

Des trous dans les formules

Les prédicats = des formules incomplètes, non saturées

☞ On va écrire des formules “inachevées” en indiquant quels éléments manquent.

$\lambda x\, \mathbf{dormir}(x)$ désaturée

$\lambda x \approx$ “dans ce qui suit, il manque une valeur pour x”

- λ : opérateur d’abstraction
- λ s’applique à une variable (c’est un lieur)
- On appellera une expression qui commence par λ un λ-terme.

$\lambda y \lambda x\, \mathbf{regarder}(x, y)$

Retour à la syntaxe de LO

Des trous dans les formules

Les prédicats = des formules incomplètes, non saturées

☞ On va écrire des formules “inachevées” en indiquant quels éléments manquent.

$\lambda x\, \mathbf{dormir}(x)$ désaturée

$\lambda x \approx$ “dans ce qui suit, il manque une valeur pour x”

- λ : opérateur d’abstraction
- λ s’applique à une variable (c’est un lieur)
- On appellera une expression qui commence par λ un λ-terme.

$\lambda y \lambda x\, \mathbf{regarder}(x, y)$

$[_{\text{VP}}\ \textit{regarde}\ \textit{Bruno}] \rightsquigarrow \lambda x\, \mathbf{regarder}(x, \mathbf{b})$

Retour à la syntaxe de LO

Des trous dans les formules

Les prédicats = des formules incomplètes, non saturées

☞ On va écrire des formules “inachevées” en indiquant quels éléments manquent.

$\lambda x\, \mathbf{dormir}(x)$ désaturée

$\lambda x \approx$ “dans ce qui suit, il manque une valeur pour x”

- λ : opérateur d’abstraction
- λ s’applique à une variable (c’est un lieur)
- On appellera une expression qui commence par λ un λ-terme.

$\lambda y \lambda x\, \mathbf{regarder}(x, y)$

$[_{\text{VP}}\ \textit{regarde}\ \textit{Bruno}] \rightsquigarrow \lambda x\, \mathbf{regarder}(x, \mathbf{b})$
Remarque : $\lambda x\, \mathbf{dormir}(x) \equiv \mathbf{dormir}$ et $\lambda y \lambda x\, \mathbf{regarder}(x, y) \equiv \mathbf{regarder}$

λ-abstraction

Une nouvelle règle syntaxique de LO

$\lambda x\, \mathbf{dormir}(x)$, $\lambda y \lambda x\, \mathbf{regarder}(x, y)$, $\lambda x\, \mathbf{regarder}(x, \mathbf{b})$ sont des expressions bien formées de LO (mais pas des formules !)

λ-abstraction

Une nouvelle règle syntaxique de LO

$\lambda x\, \mathbf{dormir}(x)$, $\lambda y \lambda x\, \mathbf{regarder}(x, y)$, $\lambda x\, \mathbf{regarder}(x, \mathbf{b})$ sont des expressions bien formées de LO (mais pas des formules !)

λ-abstraction

Si α est une expression bien formée de LO et si v est une variable de LO, alors $\lambda v \alpha$ est une expression bien formée de LO.

λ-abstraction

Une nouvelle règle syntaxique de LO

$\lambda x\, \mathbf{dormir}(x)$, $\lambda y \lambda x\, \mathbf{regarder}(x, y)$, $\lambda x\, \mathbf{regarder}(x, \mathbf{b})$ sont des expressions bien formées de LO (mais pas des formules !)

λ-abstraction

Si α est une expression bien formée de LO et si v est une variable de LO, alors $\lambda v \alpha$ est une expression bien formée de LO.

Un $\lambda \longrightarrow$ prédicat à un argument,
Deux $\lambda \longrightarrow$ prédicat à deux arguments, etc.
donc $\lambda x\, \mathbf{regarder}(x, \mathbf{b})$ est un prédicat à un argument (normal, c'est un VP)

Types : catégories d'expressions de LO

Types de base

Nous allons catégoriser les expressions de LO selon ce qu'elles dénotent dans le modèle.
Ces catégories s'appellent des types.

Une expression est bien formée si et seulement si elle a un type.

Types : catégories d'expressions de LO

Types de base

Nous allons catégoriser les expressions de LO selon ce qu'elles dénotent dans le modèle.
Ces catégories s'appellent des types.

Une expression est bien formée si et seulement si elle a un type.

- Les expressions qui dénotent des individus de $\mathcal{A}$ sont de type e.
  Ce sont les termes.

Types : catégories d'expressions de LO

Types de base

Nous allons catégoriser les expressions de LO selon ce qu'elles dénotent dans le modèle.
Ces catégories s'appellent des types.

Une expression est bien formée si et seulement si elle a un type.

- Les expressions qui dénotent des individus de $\mathcal{A}$ sont de type e.
  Ce sont les termes.
- Les expressions qui dénotent une valeur de vérité (1/0) sont de type t.
  Ce sont les formules.

Types : catégories d'expressions de LO

Types de base

Nous allons catégoriser les expressions de LO selon ce qu'elles dénotent dans le modèle.
Ces catégories s'appellent des types.

Une expression est bien formée si et seulement si elle a un type.

- Les expressions qui dénotent des individus de $\mathcal{A}$ sont de type e.
  Ce sont les termes.
- Les expressions qui dénotent une valeur de vérité (1/0) sont de type t.
  Ce sont les formules.
- Et les autres expressions ? Les prédicats ?

Types : catégories d'expressions de LO

Types de base

Nous allons catégoriser les expressions de LO selon ce qu'elles dénotent dans le modèle.
Ces catégories s'appellent des types.

Une expression est bien formée si et seulement si elle a un type.

- Les expressions qui dénotent des individus de $\mathcal{A}$ sont de type e.
  Ce sont les termes.
- Les expressions qui dénotent une valeur de vérité (1/0) sont de type t.
  Ce sont les formules.
- Et les autres expressions ? Les prédicats ?
  Ils dénotent des fonctions.

Interprétation des λ-termes

Les prédicats à un argument, comme **dormir** ou $\lambda x\,\textbf{dormir}(x)$ dénotent des fonctions de $\mathcal{A}$ vers $\{0\,;\,1\}$. $\mathcal{A} \longrightarrow \{0\,;\,1\}$

$$[\![\lambda x\,\textbf{dormir}(x)]\!]^{\mathcal{M},w,g} = \left[\begin{array}{ll} \textsc{Alice} & \\ \textsc{Bruno} & 1 \\ \textsc{Charles} & \\ \textsc{Dina} & 0 \end{array}\right]$$

- $[\![\lambda x\,\textbf{dormir}(x)]\!]^{\mathcal{M},w,g}$ attend un argument qui est un individu et retourne comme résultat une valeur de vérité.
- $\lambda x\,\textbf{dormir}(x)$ attend un argument de type e et donne comme résultat une expression qui est de type t, i.e. une formule.

Interprétation des λ-termes

Les prédicats à un argument, comme **dormir** ou $\lambda x\,\textbf{dormir}(x)$ dénotent des fonctions de $\mathcal{A}$ vers $\{0\,;\,1\}$. $\mathcal{A} \longrightarrow \{0\,;\,1\}$

$$[\![\lambda x\,\textbf{dormir}(x)]\!]^{\mathcal{M},w,g} = \left[\begin{array}{ll} \textsc{Alice} & \\ \textsc{Bruno} & 1 \\ \textsc{Charles} & \\ \textsc{Dina} & 0 \end{array}\right]$$

- $[\![\lambda x\,\textbf{dormir}(x)]\!]^{\mathcal{M},w,g}$ attend un argument qui est un individu et retourne comme résultat une valeur de vérité.
- $\lambda x\,\textbf{dormir}(x)$ attend un argument de type e et donne comme résultat une expression qui est de type t, i.e. une formule.

C'est une fonction de type e → t

Interprétation des λ-termes

Les prédicats à un argument, comme **dormir** ou $\lambda x\,\textbf{dormir}(x)$ dénotent des fonctions de $\mathcal{A}$ vers $\{0\,;\,1\}$. $\mathcal{A} \longrightarrow \{0\,;\,1\}$

$$[\![\lambda x\,\textbf{dormir}(x)]\!]^{\mathcal{M},w,g} = \left[\begin{array}{ll} \textsc{Alice} & \\ \textsc{Bruno} & 1 \\ \textsc{Charles} & \\ \textsc{Dina} & 0 \end{array}\right]$$

- $[\![\lambda x\,\textbf{dormir}(x)]\!]^{\mathcal{M},w,g}$ attend un argument qui est un individu et retourne comme résultat une valeur de vérité.
- $\lambda x\,\textbf{dormir}(x)$ attend un argument de type e et donne comme résultat une expression qui est de type t, i.e. une formule.

C'est une fonction de type $\langle \mathrm{e}, \mathrm{t} \rangle$

Interprétation des λ-termes

Les prédicats à un argument, comme **dormir** ou $\lambda x\,\textbf{dormir}(x)$ dénotent des fonctions de $\mathcal{A}$ vers $\{0\,;\,1\}$. $\mathcal{A} \longrightarrow \{0\,;\,1\}$

$$[\![\lambda x\,\textbf{dormir}(x)]\!]^{\mathcal{M},w,g} = \left[\begin{array}{ll} \textsc{Alice} & \\ \textsc{Bruno} & 1 \\ \textsc{Charles} & \\ \textsc{Dina} & 0 \end{array}\right]$$

- $[\![\lambda x\,\textbf{dormir}(x)]\!]^{\mathcal{M},w,g}$ attend un argument qui est un individu et retourne comme résultat une valeur de vérité.
- $\lambda x\,\textbf{dormir}(x)$ attend un argument de type e et donne comme résultat une expression qui est de type t, i.e. une formule.

C'est une fonction de type $\langle \mathrm{e}, \mathrm{t}\rangle$

$\langle \mathrm{e}, \mathrm{t}\rangle$ = type des expressions qui dénotent une fonction dont l'argument est de type e et le résultat est de type t.

Type fonctionnel

Type fonctionnel

Si a et b sont deux types, alors $\langle a, b \rangle$ est aussi un type.

Type fonctionnel

Type fonctionnel

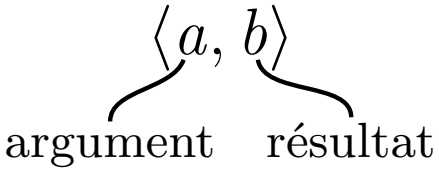
Si a et b sont deux types, alors $\langle a, b\rangle$ est aussi un type. C'est le type des expressions qui dénotent des fonctions dont l'argument est de type a et le résultat de type b.

Type fonctionnel

Type fonctionnel

Si a et b sont deux types, alors $\langle a, b\rangle$ est aussi un type. C'est le type des expressions qui dénotent des fonctions dont l'argument est de type a et le résultat de type b.

Les types fonctionnels sont toujours binaires (malgré les apparences).

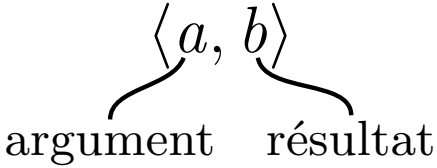


Type fonctionnel

Type fonctionnel

Si a et b sont deux types, alors $\langle a, b\rangle$ est aussi un type. C'est le type des expressions qui dénotent des fonctions dont l'argument est de type a et le résultat de type b.

Les types fonctionnels sont toujours binaires (malgré les apparences).



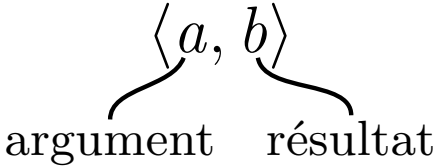
Donc techniquement les expressions fonctionnelles (i.e. les prédicats) de LO dénotent toujours des fonctions à 1 argument

Type fonctionnel

Type fonctionnel

Si a et b sont deux types, alors $\langle a, b\rangle$ est aussi un type. C'est le type des expressions qui dénotent des fonctions dont l'argument est de type a et le résultat de type b.

Les types fonctionnels sont toujours binaires (malgré les apparences).



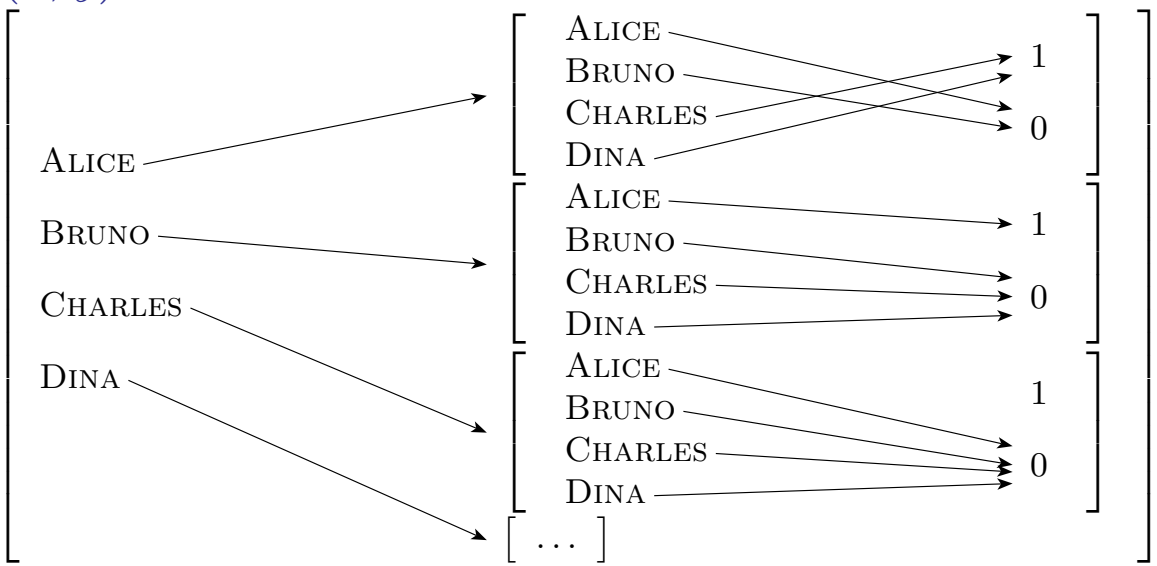
Donc techniquement les expressions fonctionnelles (i.e. les prédicats) de LO dénotent toujours des fonctions à 1 argument (... à la fois).

Prédicats à deux arguments

Un prédicat à deux arguments dénote une fonction qui lorsqu'on lui donne un argument de type e, renvoie une fonction de type $\langle e, t \rangle$.

$\lambda y \lambda x \, \textbf{regarder}(x, y)$

$[\![\textbf{regarder}]\!]^{\mathcal{M},w,g} =$

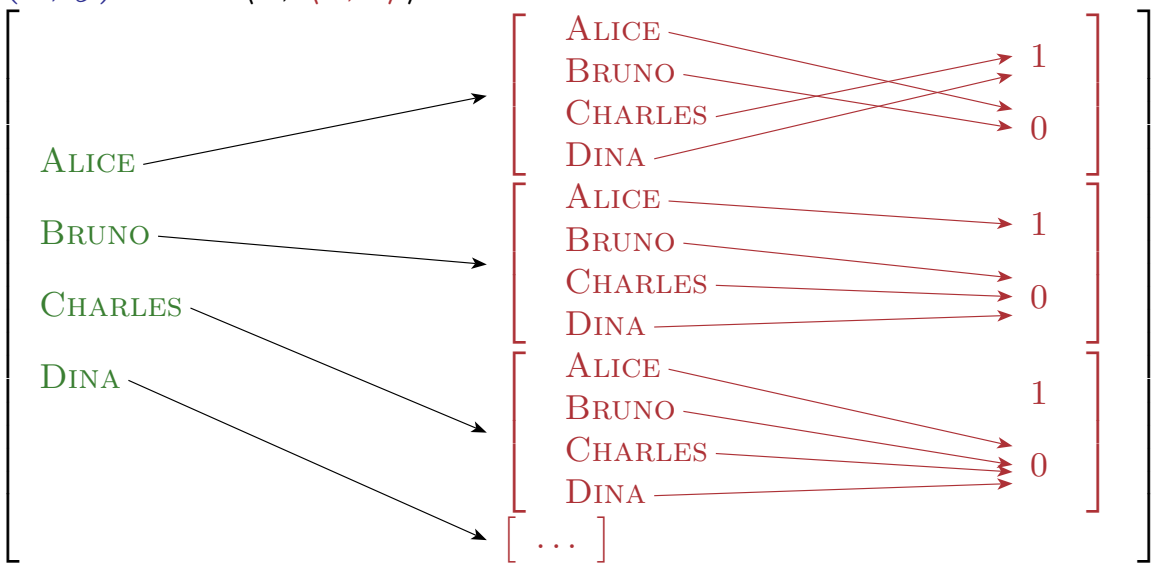


Prédicats à deux arguments

Un prédicat à deux arguments dénote une fonction qui lorsqu'on lui donne un argument de type e, renvoie une fonction de type $\langle e, t\rangle$.

$\lambda y \lambda x\, \mathbf{regarder}(x, y)$ $\langle e, \langle e, t\rangle\rangle$

$[\![\mathbf{regarder}]\!]^{\mathcal{M},w,g} =$



Des fonctions dans le modèle

On conçoit une grande variété de fonctions dans un modèle donné.

type	dénotation : fonction	expressions de LO
$\langle e, t\rangle$	$\mathcal{A} \longrightarrow \{0\ ;\ 1\}$	prédicat à 1 place
$\langle e, \langle e, t\rangle\rangle$	$\mathcal{A} \longrightarrow (\mathcal{A} \longrightarrow \{0\ ;\ 1\})$	prédicat à 2 places
$\langle e, \langle e, \langle e, t\rangle\rangle\rangle$	$\mathcal{A} \longrightarrow (\mathcal{A} \longrightarrow (\mathcal{A} \longrightarrow \{0\ ;\ 1\}))$	prédicat à 3 places
$\langle t, t\rangle$	$\{0\ ;\ 1\} \longrightarrow \{0\ ;\ 1\}$	opérateur logique (comme $\neg$)
$\langle\langle e, t\rangle, t\rangle$	$(\mathcal{A} \longrightarrow \{0\ ;\ 1\}) \longrightarrow \{0\ ;\ 1\}$	ensemble de prédicats à 1 place

Des fonctions dans le modèle

On conçoit une grande variété de fonctions dans un modèle donné.

type	dénotation : fonction	expressions de LO
$\langle e, t\rangle$	$\mathcal{A} \longrightarrow \{0\ ;\ 1\}$	prédicat à 1 place
$\langle e, \langle e, t\rangle\rangle$	$\mathcal{A} \longrightarrow (\mathcal{A} \longrightarrow \{0\ ;\ 1\})$	prédicat à 2 places
$\langle e, \langle e, \langle e, t\rangle\rangle\rangle$	$\mathcal{A} \longrightarrow (\mathcal{A} \longrightarrow (\mathcal{A} \longrightarrow \{0\ ;\ 1\}))$	prédicat à 3 places
$\langle t, t\rangle$	$\{0\ ;\ 1\} \longrightarrow \{0\ ;\ 1\}$	opérateur logique (comme $\neg$)
$\langle\langle e, t\rangle, t\rangle$	$(\mathcal{A} \longrightarrow \{0\ ;\ 1\}) \longrightarrow \{0\ ;\ 1\}$	ensemble de prédicats à 1 place

Rappel : la dénotation d'une expression de type $\langle e, t\rangle$ peut aussi être vue comme un **ensemble** d'individus.

Des fonctions dans le modèle

On conçoit une grande variété de fonctions dans un modèle donné.

type	dénotation : fonction	expressions de LO
$\langle e, t\rangle$	$\mathcal{A} \longrightarrow \{0\ ;\ 1\}$	prédicat à 1 place
$\langle e, \langle e, t\rangle\rangle$	$\mathcal{A} \longrightarrow (\mathcal{A} \longrightarrow \{0\ ;\ 1\})$	prédicat à 2 places
$\langle e, \langle e, \langle e, t\rangle\rangle\rangle$	$\mathcal{A} \longrightarrow (\mathcal{A} \longrightarrow (\mathcal{A} \longrightarrow \{0\ ;\ 1\}))$	prédicat à 3 places
$\langle t, t\rangle$	$\{0\ ;\ 1\} \longrightarrow \{0\ ;\ 1\}$	opérateur logique (comme $\neg$)
$\langle\langle e, t\rangle, t\rangle$	$(\mathcal{A} \longrightarrow \{0\ ;\ 1\}) \longrightarrow \{0\ ;\ 1\}$	ensemble de prédicats à 1 place

Rappel : la dénotation d'une expression de type $\langle e, t\rangle$ peut aussi être vue comme un **ensemble** d'individus.

De manière générale : on assimile la dénotation de toute expression de type $\langle a, t\rangle$ à un ensemble d'objets qui sont de type a.

La fabrique de fonctions

Anatomie d'un λ-terme

Révision de la règle syntaxique (cf. règle (Syn.7))

λ-abstraction

Si α est une expression de type a dans LO et si v est une variable de type b, alors $\lambda v \alpha$ est une expression bien formée de type $\langle b, a \rangle$.

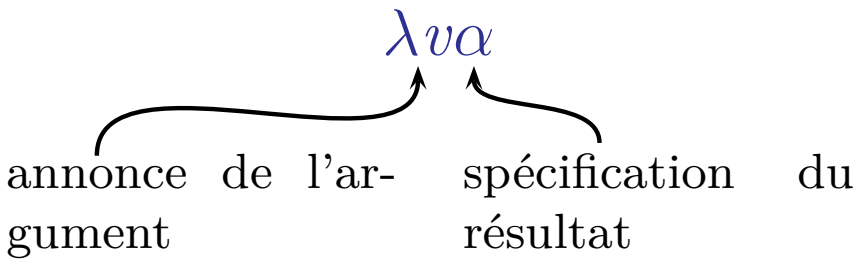
La fabrique de fonctions

Anatomie d'un λ-terme

Révision de la règle syntaxique (cf. règle (Syn.7))

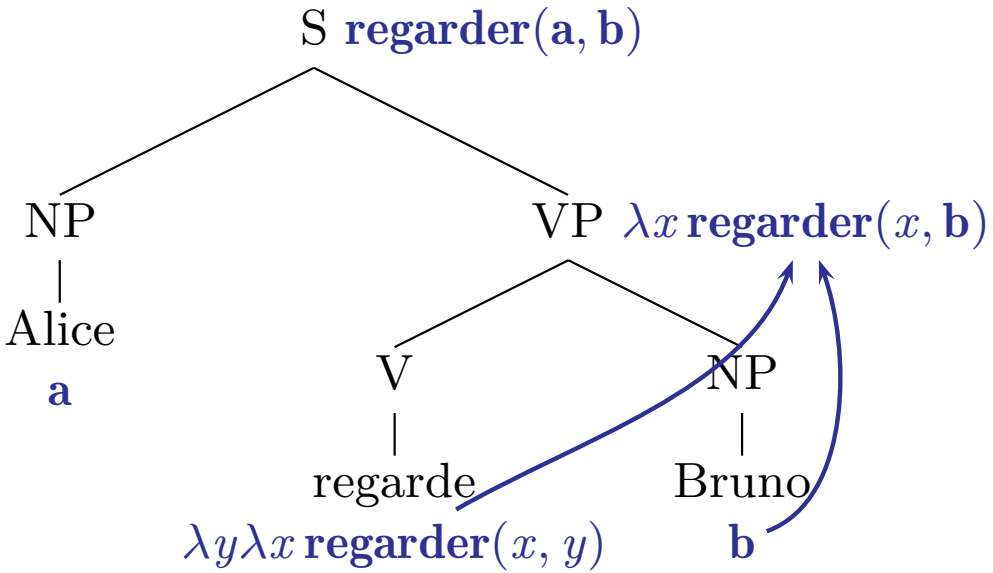
λ-abstraction

Si α est une expression de type a dans LO et si v est une variable de type b, alors $\lambda v \alpha$ est une expression bien formée de type $\langle b, a \rangle$.



C'est une règle qui complexifie les types.

Composition des formules



Méthode : on construit la représentation sémantique d'un nœud à partir de la représentation des nœuds qu'il domine directement.

Application fonctionnelle

Nourrir les fonctions

Une fonction, par essence, attend qu'on lui donne un argument.
On va lui en donner.

Application fonctionnelle

Nourrir les fonctions

Une fonction, par essence, attend qu'on lui donne un argument.
On va lui en donner.

γ

fonction γ

Application fonctionnelle

Nourrir les fonctions

Une fonction, par essence, attend qu'on lui donne un argument.
On va lui en donner.

$$[\gamma(\beta)]$$

On fournit β comme argument à la fonction γ

Application fonctionnelle

Nourrir les fonctions

Une fonction, par essence, attend qu'on lui donne un argument.
On va lui en donner.

Application fonctionnelle (dans LO)

Si γ est de type $\langle b, a\rangle$ et si β est de type b, alors $[\gamma(\beta)]$ est de type a.

$$[\gamma(\beta)]$$

On fournit β comme argument à la fonction γ

Je ne peux pas donner n'importe quel type d'argument à n'importe quel type de fonction.

Application fonctionnelle

Plusieurs arguments

Si γ attend plusieurs arguments :

Application fonctionnelle

Plusieurs arguments

Si γ attend plusieurs arguments :

$$\gamma$$

Fonction à 2 arguments

Application fonctionnelle

Plusieurs arguments

Si γ attend plusieurs arguments :

$$[\gamma(\beta)]$$

Fonction à 1 argument

Application fonctionnelle

Plusieurs arguments

Si γ attend plusieurs arguments :

$$[[\gamma(\beta)](\delta)]$$

Fonction à 0 argument (saturée)

Application fonctionnelle

Plusieurs arguments

Si γ attend plusieurs arguments :

$$[[\gamma(\beta)](\delta)]$$

Fonction à 0 argument (saturée)

Simplification d'écriture

On s'autorisera à simplifier toute notation $[[\gamma(\beta)](\delta)]$ en $\gamma(\delta, \beta)$.

Application fonctionnelle

Plusieurs arguments

Si γ attend plusieurs arguments :

$$[[\gamma(\beta)](\delta)]$$

Fonction à 0 argument (saturée)

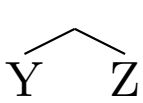
Simplification d'écriture

On s'autorisera à simplifier toute notation $[[\gamma(\beta)](\delta)]$ en $\gamma(\delta, \beta)$.
De manière générale, on simplifie $[[[[\gamma(\beta_1)](\beta_2)]\dots](\beta_n)]$ en $\gamma(\beta_n, \dots, \beta_2, \beta_1)$.

Application fonctionnelle

Opération à tout faire

Hypothèse : pour tout sous-arbre

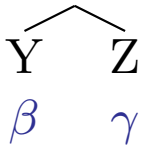


un des constituants est une fonction et l'autre son argument

Application fonctionnelle

Opération à tout faire

Hypothèse : pour tout sous-arbre X $[\beta(\gamma)]$

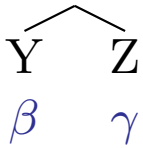


un des constituants est une fonction et l'autre son argument

Application fonctionnelle

Opération à tout faire

Hypothèse : pour tout sous-arbre X $[\gamma(\beta)]$

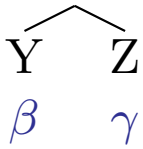


un des constituants est une fonction et l'autre son argument

Application fonctionnelle

Opération à tout faire

Hypothèse : pour tout sous-arbre X $[\gamma(\beta)]$



un des constituants est une fonction et l'autre son argument

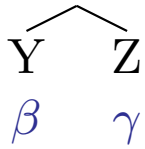
Schéma général

Règle syntaxique :	X	$\rightarrow$	Y	Z
Règle sémantique :	$[\beta(\gamma)]$	$\leftarrow$	β	γ
ou	$[\gamma(\beta)]$	$\leftarrow$	β	γ

Application fonctionnelle

Opération à tout faire

Hypothèse : pour tout sous-arbre $\quad X\ [\gamma(\beta)]$



un des constituants est une fonction et l'autre son argument

Schéma général

Règle syntaxique :	X	→	Y	Z
Règle sémantique :	$[\beta(\gamma)]$	←	β	γ
ou	$[\gamma(\beta)]$	←	β	γ

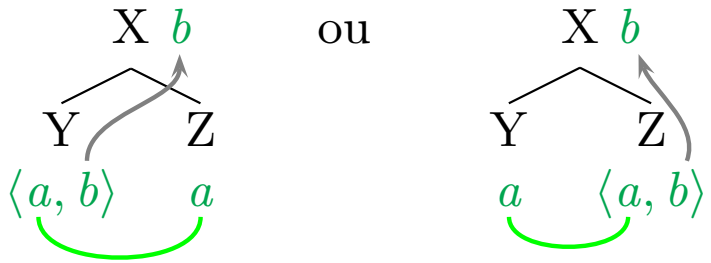
Exemples :

VP	→	V	NP
$[\beta(\gamma)]$	←	β	γ
S	→	NP	VP
$[\gamma(\beta)]$	←	β	γ

C'est grâce aux types que l'on sait qui est la fonction et qui est l'argument.

Composition sémantique guidée par les types

Puisqu'il faut respecter la règle (Syn.2), la composition sémantique ne peut se faire que dans les configurations suivantes :



Sauf si on se donne une autre règle de composition...

Exemples de calcul de types

$$\lambda P \lambda x [\mathbf{dormir}(x) \wedge P(\mathbf{a})]$$

$$\underbrace{\lambda \underbrace{P}_{\langle e,t\rangle} \underbrace{\lambda \underbrace{x}_{e} \underbrace{[\mathbf{dormir}(x) \wedge P(\mathbf{a})]}_{t}}_{\langle e,t\rangle}}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$$

Exemples de calcul de types

$$\lambda P \lambda x [\mathbf{dormir}(x) \wedge P(\mathbf{a})]$$

$$\underbrace{\lambda \underbrace{P}_{\langle e,t\rangle} \underbrace{\lambda \underbrace{x}_{e} \underbrace{[\mathbf{dormir}(x) \wedge P(\mathbf{a})]}_{t}}_{\langle e,t\rangle}}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$$

$$\lambda x \lambda P [\mathbf{dormir}(x) \wedge P(\mathbf{a})]$$

$$\underbrace{\lambda \underbrace{x}_{e} \underbrace{\lambda \underbrace{P}_{\langle e,t\rangle} \underbrace{[\mathbf{dormir}(x) \wedge P(\mathbf{a})]}_{t}}_{\langle\langle e,t\rangle,t\rangle}}_{\langle e,\langle\langle e,t\rangle,t\rangle\rangle}$$

Exemples de calcul de types

$$\lambda P \lambda x [\mathbf{dormir}(x) \wedge P(\mathbf{a})]$$

$$\underbrace{\lambda \underbrace{P}_{\langle e,t\rangle} \underbrace{\lambda \underbrace{x}_{e} \underbrace{[\mathbf{dormir}(x) \wedge P(\mathbf{a})]}_{t}}_{\langle e,t\rangle}}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$$

$$\lambda x \lambda P [\mathbf{dormir}(x) \wedge P(\mathbf{a})]$$

$$\underbrace{\lambda \underbrace{x}_{e} \underbrace{\lambda \underbrace{P}_{\langle e,t\rangle} \underbrace{[\mathbf{dormir}(x) \wedge P(\mathbf{a})]}_{t}}_{\langle\langle e,t\rangle,t\rangle}}_{\langle e,\langle\langle e,t\rangle,t\rangle\rangle}$$

$$\lambda P \lambda x \hat{} [\mathbf{dormir}(x) \wedge P(\mathbf{a})]$$

$$\underbrace{\lambda \underbrace{P}_{\langle e,t\rangle} \underbrace{\lambda \underbrace{x}_{e} \underbrace{\hat{} \underbrace{[\mathbf{dormir}(x) \wedge P(\mathbf{a})]}_{t}}_{\langle s,t\rangle}}_{\langle e,\langle s,t\rangle\rangle}}_{\langle\langle e,t\rangle,\langle e,\langle s,t\rangle\rangle\rangle}$$

Exemples de calcul de types

$\lambda P \lambda x [\mathbf{dormir}(x) \wedge P(\mathbf{a})]$

$$\underbrace{\lambda \underbrace{P}_{\langle e,t\rangle} \underbrace{\lambda \underbrace{x}_{e} \underbrace{[\mathbf{dormir}(x) \wedge P(\mathbf{a})]}_{t}}_{\langle e,t\rangle}}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$$

$\lambda x \lambda P [\mathbf{dormir}(x) \wedge P(\mathbf{a})]$

$$\underbrace{\lambda \underbrace{x}_{e} \underbrace{\lambda \underbrace{P}_{\langle e,t\rangle} \underbrace{[\mathbf{dormir}(x) \wedge P(\mathbf{a})]}_{t}}_{\langle\langle e,t\rangle,t\rangle}}_{\langle e,\langle\langle e,t\rangle,t\rangle\rangle}$$

$\lambda P \lambda x \hat{} [\mathbf{dormir}(x) \wedge P(\mathbf{a})]$

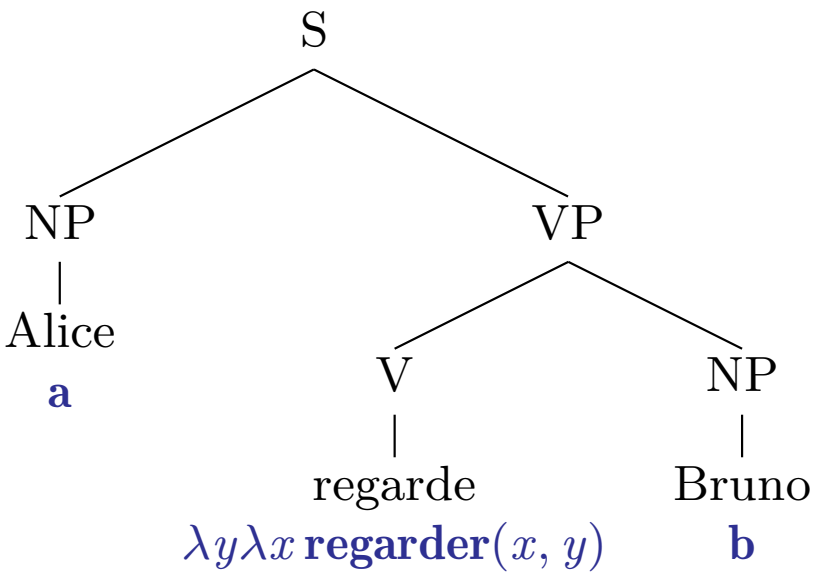
$$\underbrace{\lambda \underbrace{P}_{\langle e,t\rangle} \underbrace{\lambda \underbrace{x}_{e} \hat{} \underbrace{[\mathbf{dormir}(x) \wedge P(\mathbf{a})]}_{t}}_{\langle s,t\rangle}}_{\langle e,\langle s,t\rangle\rangle}}_{\langle\langle e,t\rangle,\langle e,\langle s,t\rangle\rangle\rangle}$$

$\lambda P \hat{} \lambda x [\mathbf{dormir}(x) \wedge P(\mathbf{a})]$

$$\underbrace{\lambda \underbrace{P}_{\langle e,t\rangle} \hat{} \underbrace{\lambda \underbrace{x}_{e} \underbrace{[\mathbf{dormir}(x) \wedge P(\mathbf{a})]}_{t}}_{\langle e,t\rangle}}_{\langle s,\langle e,t\rangle\rangle}}_{\langle\langle e,t\rangle,\langle s,\langle e,t\rangle\rangle\rangle}$$

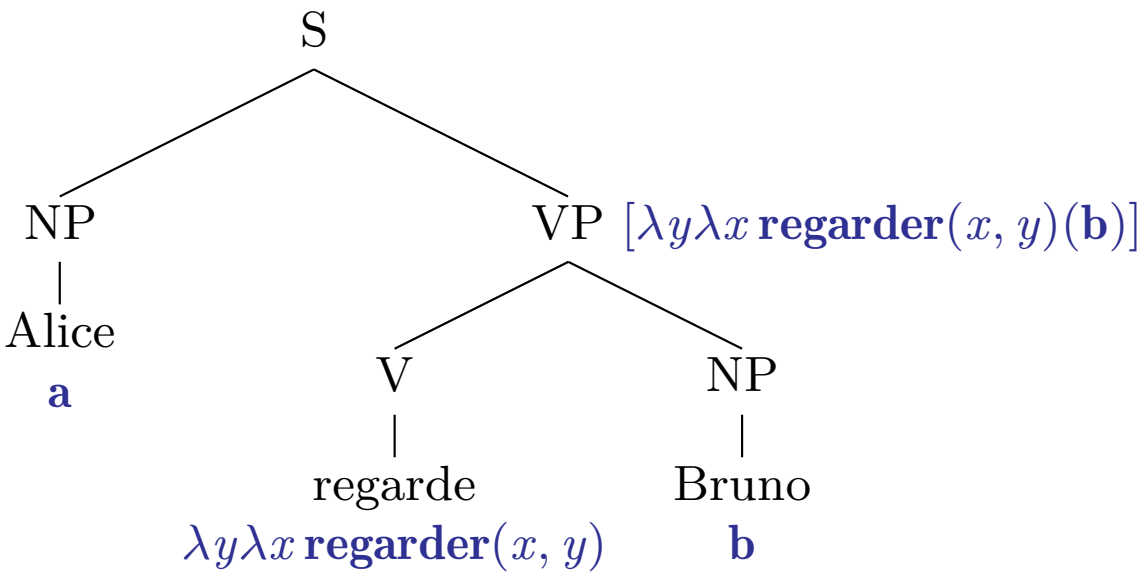
Exemple

Retour à notre arbre



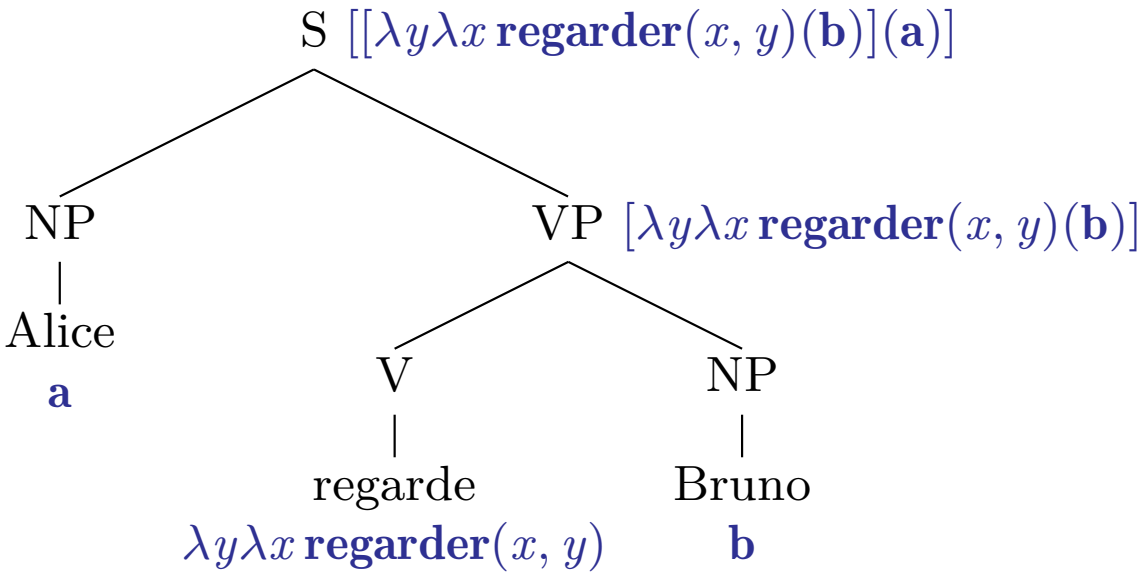
Exemple

Retour à notre arbre



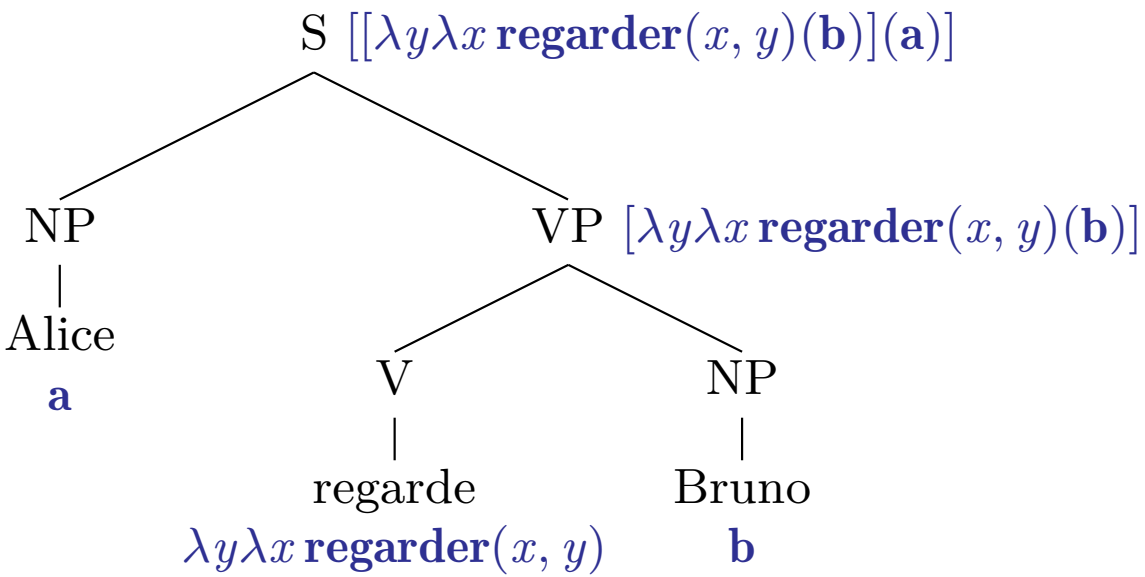
Exemple

Retour à notre arbre



Exemple

Retour à notre arbre



Pas exactement ce qu'on attendait, mais c'est l'analyse correcte.

λ-conversion / β-réduction

Une règle de simplification sémantique et graphique

β-réduction

$[\lambda v \alpha(\beta)]$ est équivalent à $[\beta/v]\alpha$
où $[\beta/v]\alpha$ est l'expression α où on a remplacé toutes les occurences libres de v par β.

λ-conversion / β-réduction

Une règle de simplification sémantique et graphique

β-réduction

$[\lambda v \alpha(\beta)]$ est équivalent à $[\beta/v]\alpha$
où $[\beta/v]\alpha$ est l'expression α où on a remplacé toutes les occurences libres de v par β.

En pratique :

$$[\lambda v \alpha(\beta)]$$

λ-conversion / β-réduction

Une règle de simplification sémantique et graphique

β-réduction

$[\lambda v \alpha(\beta)]$ est équivalent à $[\beta/v]\alpha$
où $[\beta/v]\alpha$ est l'expression α où on a remplacé toutes les occurences libres de v par β.

En pratique :

$$\lambda v \alpha(\beta)$$

λ-conversion / β-réduction

Une règle de simplification sémantique et graphique

β-réduction

$[\lambda v \alpha(\beta)]$ est équivalent à $[\beta/v]\alpha$
où $[\beta/v]\alpha$ est l'expression α où on a remplacé toutes les occurences libres de v par β.

En pratique :

$$\alpha(\beta)$$

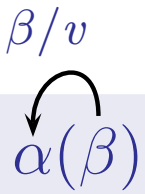
λ-conversion / β-réduction

Une règle de simplification sémantique et graphique

β-réduction

$[\lambda v \alpha(\beta)]$ est équivalent à $[\beta/v]\alpha$
où $[\beta/v]\alpha$ est l'expression α où on a remplacé toutes les occurences libres de v par β.

En pratique :



λ-conversion / β-réduction

Une règle de simplification sémantique et graphique

β-réduction

$[\lambda v \alpha(\beta)]$ est équivalent à $[\beta/v]\alpha$
où $[\beta/v]\alpha$ est l'expression α où on a remplacé toutes les occurences libres de v par β.

En pratique :

λ-conversion / β-réduction

Une règle de simplification sémantique et graphique

β-réduction

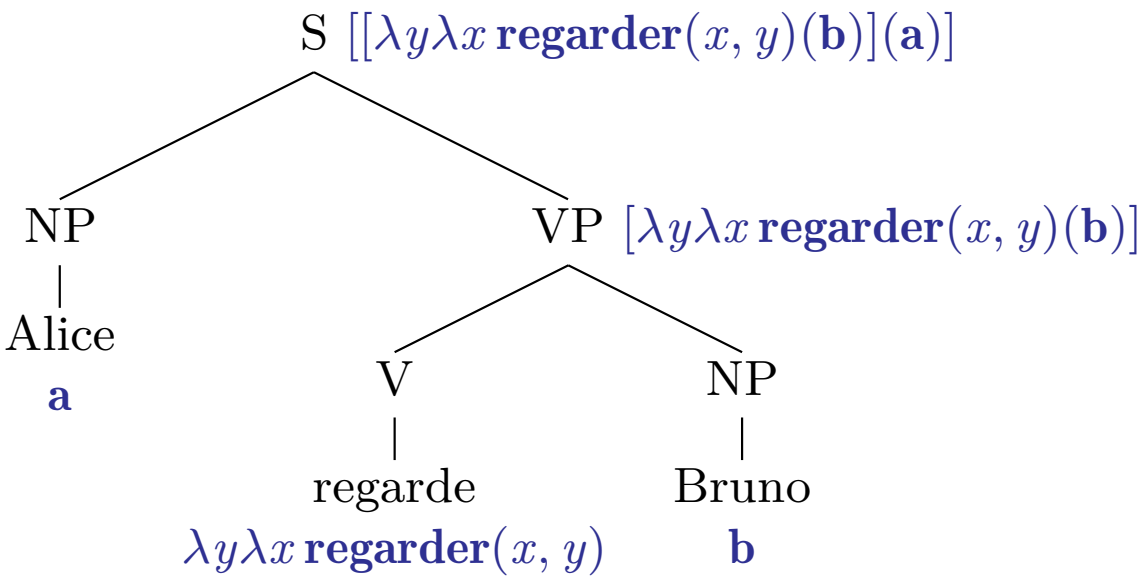
$[\lambda v \alpha(\beta)]$ est équivalent à $[\beta/v]\alpha$
où $[\beta/v]\alpha$ est l'expression α où on a remplacé toutes les occurences libres de v par β.

En pratique :

Attention : pour appliquer la β-réduction, il est indispensable de reconnaître **exactement** le schéma $[\lambda v. . . (. . .)]$

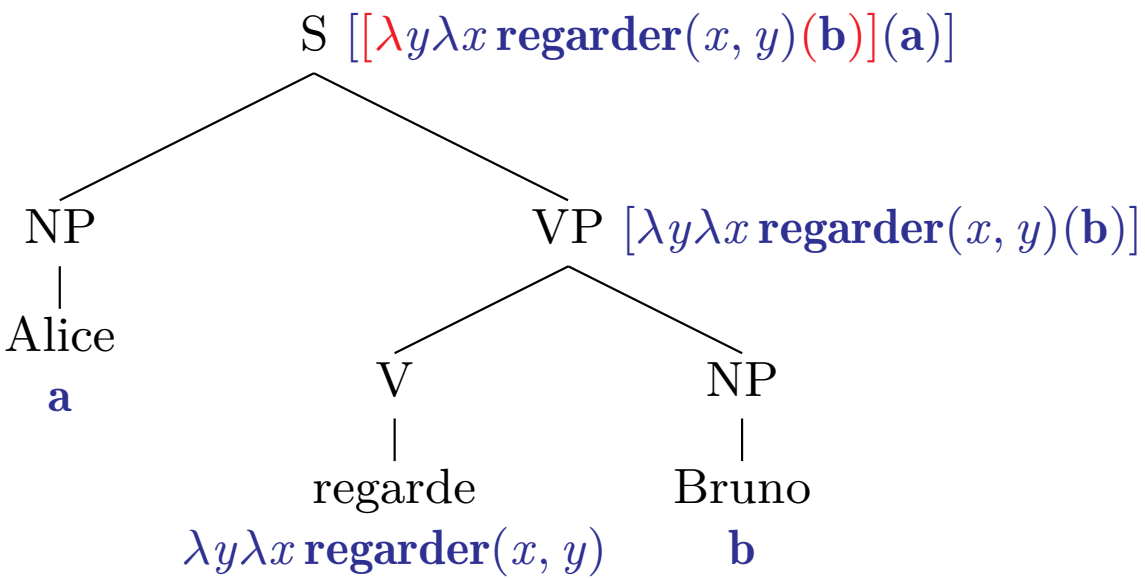
Exemple

Pour en finir avec notre arbre



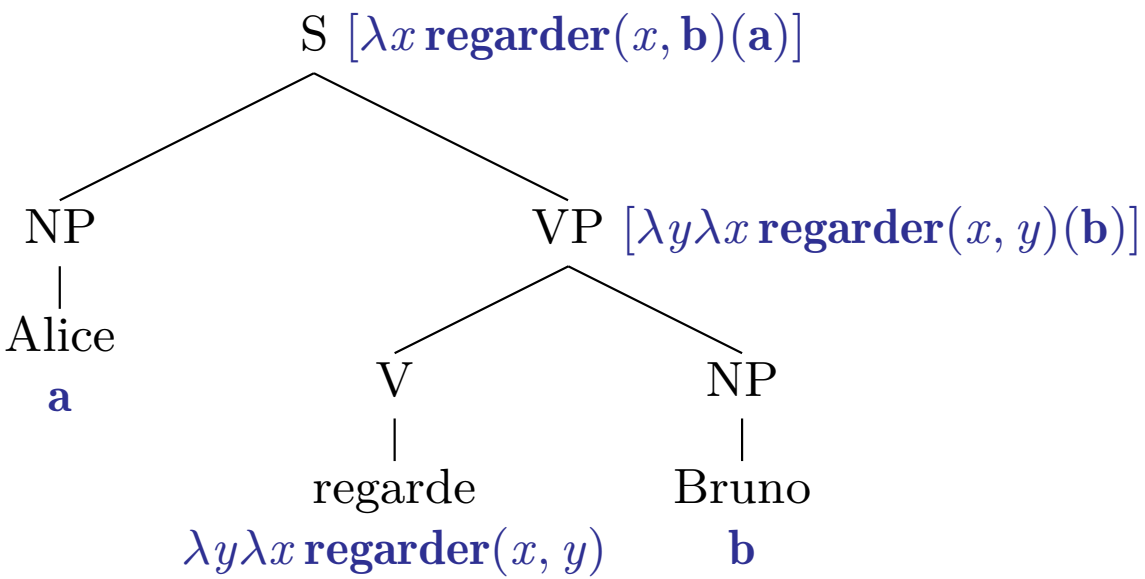
Exemple

Pour en finir avec notre arbre



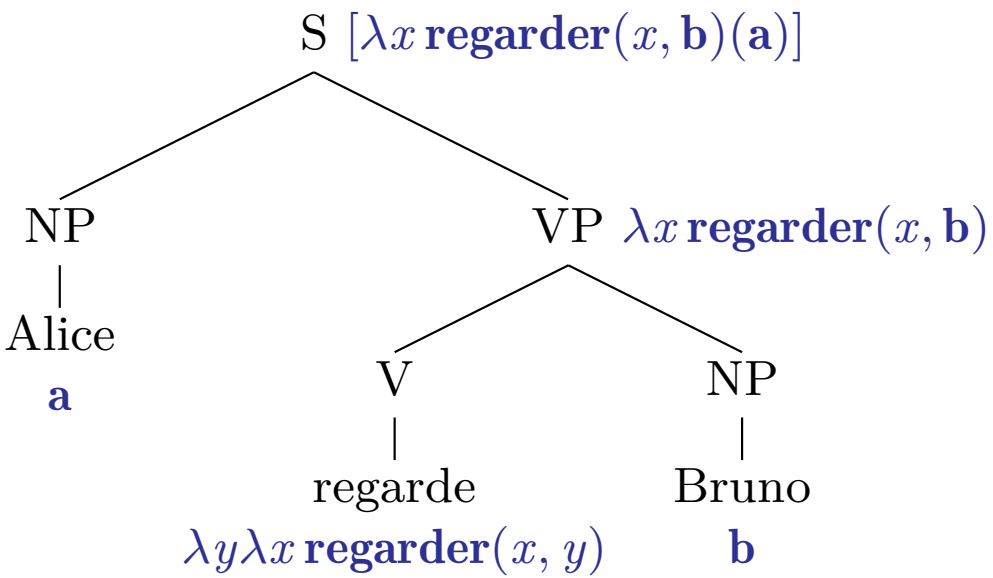
Exemple

Pour en finir avec notre arbre



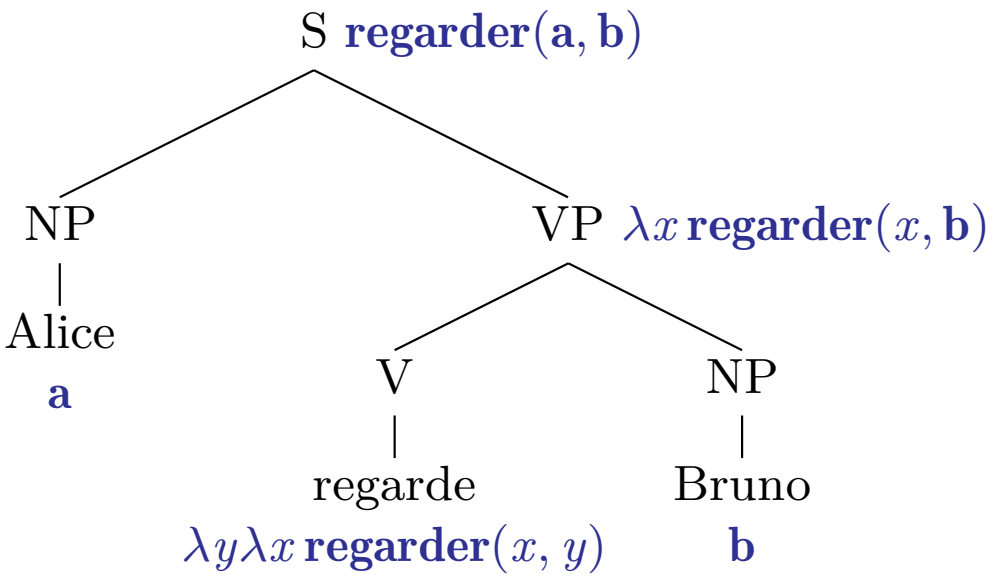
Exemple

Pour en finir avec notre arbre



Exemple

Pour en finir avec notre arbre



Limitation de la β-réduction

Un (des deux) cas de figures où la réduction est interdite

Limitation de la β-réduction

Un (des deux) cas de figures où la réduction est interdite

1. $[\![\lambda x \exists y \, \mathbf{regarder}(x, y)]\!]^{\mathcal{M},w,g} =$

Limitation de la β-réduction

Un (des deux) cas de figures où la réduction est interdite

1. $[\![\lambda x \exists y \, \mathbf{regarder}(x, y)]\!]^{\mathcal{M},w,g}$ = la fonction qui pour toute valeur de x nous renvoie la dénotation de $\exists y \, \mathbf{regarder}(x, y)$ par rapport à $\mathcal{M}$ et w

Limitation de la β-réduction

Un (des deux) cas de figures où la réduction est interdite

1. $[\![\lambda x \exists y\, \mathbf{regarder}(x, y)]\!]^{\mathcal{M},w,g}$ = la fonction qui pour toute valeur de x nous renvoie la dénotation de $\exists y\, \mathbf{regarder}(x, y)$ par rapport à $\mathcal{M}$ et w (elle retourne 1 ssi x regarde quelqu'un)

Limitation de la β-réduction

Un (des deux) cas de figures où la réduction est interdite

1. $[\![\lambda x \exists y \, \textbf{regarder}(x, y)]\!]^{\mathcal{M},w,g}$ = la fonction qui pour toute valeur de x nous renvoie la dénotation de $\exists y \, \textbf{regarder}(x, y)$ par rapport à $\mathcal{M}$ et w (elle retourne 1 ssi x regarde quelqu'un)
2. $[\![[\lambda x \exists y \, \textbf{regarder}(x, y)(z)]]\!]^{\mathcal{M},w,g} =$

Limitation de la β-réduction

Un (des deux) cas de figures où la réduction est interdite

1. $[\![\lambda x \exists y \, \textbf{regarder}(x, y)]\!]^{\mathcal{M},w,g}$ = la fonction qui pour toute valeur de x nous renvoie la dénotation de $\exists y \, \textbf{regarder}(x, y)$ par rapport à $\mathcal{M}$ et w (elle retourne 1 ssi x regarde quelqu'un)
2. $[\![[\lambda x \exists y \, \textbf{regarder}(x, y)(z)]]\!]^{\mathcal{M},w,g} =$ 1 ssi la dénotation de z est un individu qui regarde quelqu'un dans w.

Limitation de la β-réduction

Un (des deux) cas de figures où la réduction est interdite

1. $[\![\lambda x \exists y\, \textbf{regarder}(x, y)]\!]^{\mathcal{M},w,g}$ = la fonction qui pour toute valeur de x nous renvoie la dénotation de $\exists y\, \textbf{regarder}(x, y)$ par rapport à $\mathcal{M}$ et w (elle retourne 1 ssi x regarde quelqu'un)
2. $[\![[\lambda x \exists y\, \textbf{regarder}(x, y)(z)]]\!]^{\mathcal{M},w,g}$ = 1 ssi la dénotation de z est un individu qui regarde quelqu'un dans w.
3. Confirmé par β-réduction : $\exists y\, \textbf{regarder}(z, y)$

Limitation de la β-réduction

Un (des deux) cas de figures où la réduction est interdite

1. $[\![\lambda x \exists y\, \textbf{regarder}(x, y)]\!]^{\mathcal{M},w,g}$ = la fonction qui pour toute valeur de x nous renvoie la dénotation de $\exists y\, \textbf{regarder}(x, y)$ par rapport à $\mathcal{M}$ et w (elle retourne 1 ssi x regarde quelqu'un)
2. $[\![[\lambda x \exists y\, \textbf{regarder}(x, y)(z)]]\!]^{\mathcal{M},w,g} =$ 1 ssi la dénotation de z est un individu qui regarde quelqu'un dans w.
3. Confirmé par β-réduction : $\exists y\, \textbf{regarder}(z, y)$
4. Le calcul (2) vaut pour n'importe quel argument donné au λ-terme : $[\![[\lambda x \exists y\, \textbf{regarder}(x, y)(y)]]\!]^{\mathcal{M},w,g} = 1$ ssi la dénotation de y est un individu qui regarde quelqu'un dans w.

Limitation de la β-réduction

Un (des deux) cas de figures où la réduction est interdite

1. $[\![\lambda x \exists y\, \textbf{regarder}(x, y)]\!]^{\mathcal{M},w,g}$ = la fonction qui pour toute valeur de x nous renvoie la dénotation de $\exists y\, \textbf{regarder}(x, y)$ par rapport à $\mathcal{M}$ et w (elle retourne 1 ssi x regarde quelqu'un)
2. $[\![[\lambda x \exists y\, \textbf{regarder}(x, y)(z)]]\!]^{\mathcal{M},w,g}$ = 1 ssi la dénotation de z est un individu qui regarde quelqu'un dans w.
3. Confirmé par β-réduction : $\exists y\, \textbf{regarder}(z, y)$
4. Le calcul (2) vaut pour n'importe quel argument donné au λ-terme : $[\![[\lambda x \exists y\, \textbf{regarder}(x, y)(y)]]\!]^{\mathcal{M},w,g} = 1$ ssi la dénotation de y est un individu qui regarde quelqu'un dans w.
5. **Mais** par β-réduction : $\exists y\, \textbf{regarder}(y, y)$

Limitation de la β-réduction

Un (des deux) cas de figures où la réduction est interdite

1. $[\![\lambda x \exists y\, \textbf{regarder}(x, y)]\!]^{\mathcal{M},w,g}$ = la fonction qui pour toute valeur de x nous renvoie la dénotation de $\exists y\, \textbf{regarder}(x, y)$ par rapport à $\mathcal{M}$ et w (elle retourne 1 ssi x regarde quelqu'un)
2. $[\![[\lambda x \exists y\, \textbf{regarder}(x, y)(z)]]\!]^{\mathcal{M},w,g} =$ 1 ssi la dénotation de z est un individu qui regarde quelqu'un dans w.
3. Confirmé par β-réduction : $\exists y\, \textbf{regarder}(z, y)$
4. Le calcul (2) vaut pour n'importe quel argument donné au λ-terme : $[\![[\lambda x \exists y\, \textbf{regarder}(x, y)(y)]]\!]^{\mathcal{M},w,g} = 1$ ssi la dénotation de y est un individu qui regarde quelqu'un dans w.
5. **Mais** par β-réduction : $\exists y\, \textbf{regarder}(y, y)$
   ce qui signifie que *quelqu'un se regarde lui-même*!

Limitation de la β-réduction

Un (des deux) cas de figures où la réduction est interdite

1. $[\![\lambda x \exists y\, \textbf{regarder}(x, y)]\!]^{\mathcal{M},w,g}$ = la fonction qui pour toute valeur de x nous renvoie la dénotation de $\exists y\, \textbf{regarder}(x, y)$ par rapport à $\mathcal{M}$ et w (elle retourne 1 ssi x regarde quelqu'un)
2. $[\![[\lambda x \exists y\, \textbf{regarder}(x, y)(z)]]\!]^{\mathcal{M},w,g}$ = 1 ssi la dénotation de z est un individu qui regarde quelqu'un dans w.
3. Confirmé par β-réduction : $\exists y\, \textbf{regarder}(z, y)$
4. Le calcul (2) vaut pour n'importe quel argument donné au λ-terme : $[\![[\lambda x \exists y\, \textbf{regarder}(x, y)(y)]]\!]^{\mathcal{M},w,g} = 1$ ssi la dénotation de y est un individu qui regarde quelqu'un dans w.
5. **Mais** par β-réduction : $\exists y\, \textbf{regarder}(y, y)$
   ce qui signifie que *quelqu'un se regarde lui-même*!

Contrainte sur la β-réduction

Soit $[\alpha(\gamma)]$ une application fonctionnelle. Une variable libre dans γ ne doit pas se retrouver liée dans α après β-réduction.

Limitation de la β-réduction

Un (des deux) cas de figures où la réduction est interdite

1. $[\![\lambda x \exists y\, \mathbf{regarder}(x, y)]\!]^{\mathcal{M},w,g}$ = la fonction qui pour toute valeur de x nous renvoie la dénotation de $\exists y\, \mathbf{regarder}(x, y)$ par rapport à $\mathcal{M}$ et w (elle retourne 1 ssi x regarde quelqu'un)
2. $[\![[\lambda x \exists y\, \mathbf{regarder}(x, y)(z)]]\!]^{\mathcal{M},w,g}$ = 1 ssi la dénotation de z est un individu qui regarde quelqu'un dans w.
3. Confirmé par β-réduction : $\exists y\, \mathbf{regarder}(z, y)$
4. Le calcul (2) vaut pour n'importe quel argument donné au λ-terme : $[\![[\lambda x \exists y\, \mathbf{regarder}(x, y)(y)]]\!]^{\mathcal{M},w,g} = 1$ ssi la dénotation de y est un individu qui regarde quelqu'un dans w.
5. **Mais** par β-réduction : $\exists y\, \mathbf{regarder}(y, y)$
   ce qui signifie que *quelqu'un se regarde lui-même*!

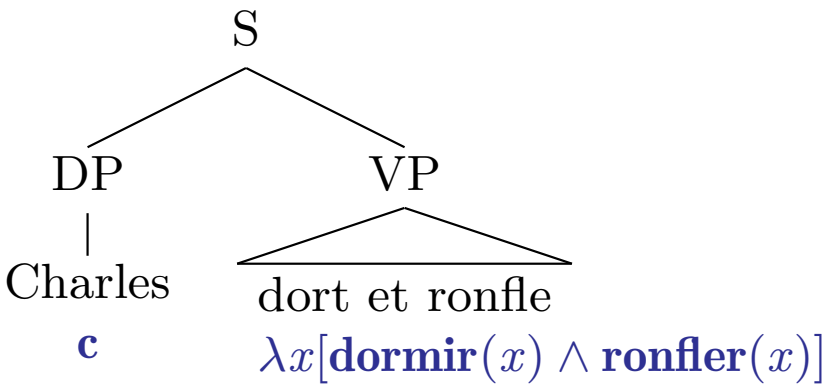
Contrainte sur la β-réduction

Soit $[\alpha(\gamma)]$ une application fonctionnelle. Une variable libre dans γ ne doit pas se retrouver liée dans α après β-réduction.

Heureusement : $\lambda x \exists y\, \mathbf{regarder}(x, y) \equiv \lambda x \exists u\, \mathbf{regarder}(x, u)$

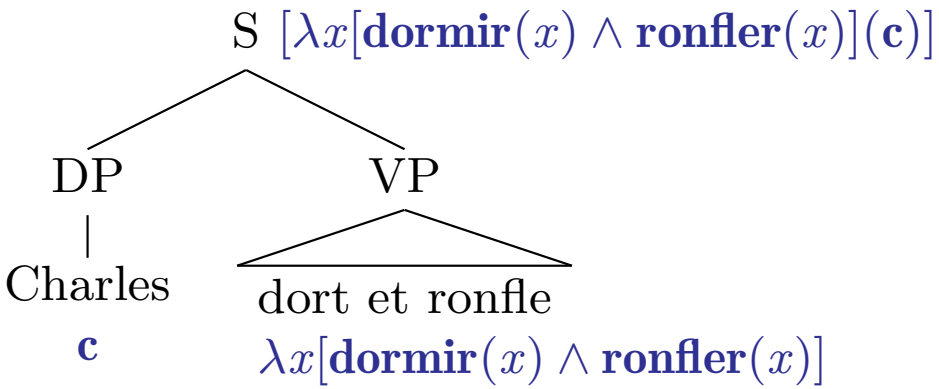
Application à la coordination de VP

(1) Charles dort et ronfle.



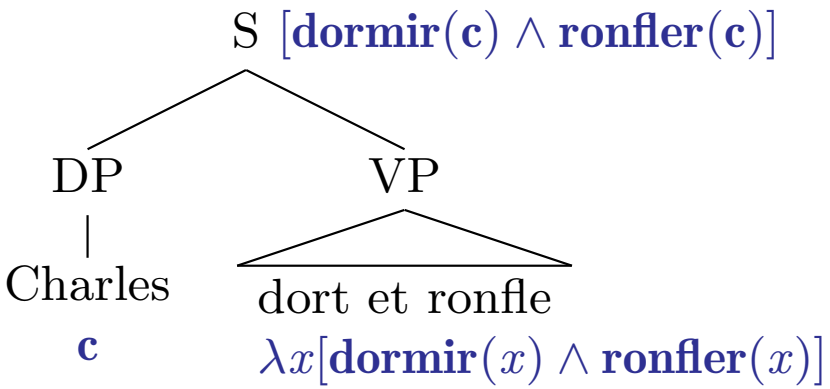
Application à la coordination de VP

(1) Charles dort et ronfle.



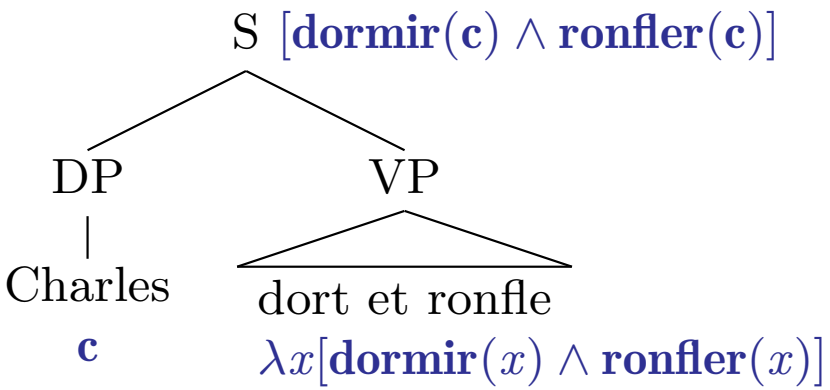
Application à la coordination de VP

(1) Charles dort et ronfle.



Application à la coordination de VP

(1) Charles dort et ronfle.



On a introduit **c** une seule fois.

Variables pour tous les types

Vers un ordre supérieur

- On va se donner des variables de n'importe quel type.

Variables pour tous les types

Vers un ordre supérieur

- On va se donner des variables de n'importe quel type.
- Ex : soit P une variable de type $\langle \mathrm{e}, \mathrm{t} \rangle$. C'est une variable de prédicat.

Variables pour tous les types

Vers un ordre supérieur

- On va se donner des variables de n'importe quel type.
- Ex : soit P une variable de type $\langle \mathrm{e}, \mathrm{t} \rangle$. C'est une variable de prédicat.
- Pour n'importe quel type, on sait quel genre d'*objet* dénote toute expression de ce type (une entité, une valeur de vérité, ou une fonction).
- Donc pour toute variable d'un type donné, g lui assignera une valeur appropriée.
  $g(P) =$ une fonction de $\mathcal{A}$ vers $\{0 ; 1\}$, ou, plus simplement, un sous-ensemble de $\mathcal{A}$.

Variables pour tous les types

Vers un ordre supérieur

- On va se donner des variables de n'importe quel type.
- Ex : soit P une variable de type $\langle \mathrm{e}, \mathrm{t} \rangle$. C'est une variable de prédicat.
- Pour n'importe quel type, on sait quel genre d'*objet* dénote toute expression de ce type (une entité, une valeur de vérité, ou une fonction).
- Donc pour toute variable d'un type donné, g lui assignera une valeur appropriée.
  $g(P) =$ une fonction de $\mathcal{A}$ vers $\{0\ ;\ 1\}$, ou, plus simplement, un sous-ensemble de $\mathcal{A}$.
- Autre exemple :
  Si p et q sont des variables de type t, $g(p) = 0$ ou 1 ; idem pour $g(q)$.

Variables pour tous les types

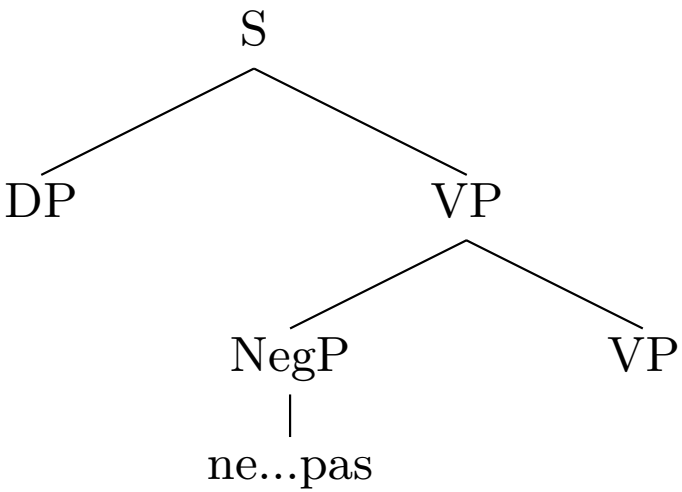
Vers un ordre supérieur

- On va se donner des variables de n'importe quel type.
- Ex : soit P une variable de type $\langle \mathrm{e}, \mathrm{t} \rangle$. C'est une variable de prédicat.
- Pour n'importe quel type, on sait quel genre d'*objet* dénote toute expression de ce type (une entité, une valeur de vérité, ou une fonction).
- Donc pour toute variable d'un type donné, g lui assignera une valeur appropriée.
  $g(P) =$ une fonction de $\mathcal{A}$ vers $\{0\,;\,1\}$, ou, plus simplement, un sous-ensemble de $\mathcal{A}$.
- Autre exemple :
  Si p et q sont des variables de type t, $g(p) = 0$ ou 1 ; idem pour $g(q)$.
  Et on peut écrire le λ-terme :
  $\lambda p \lambda q[p \wedge q]$

Exemples

La négation

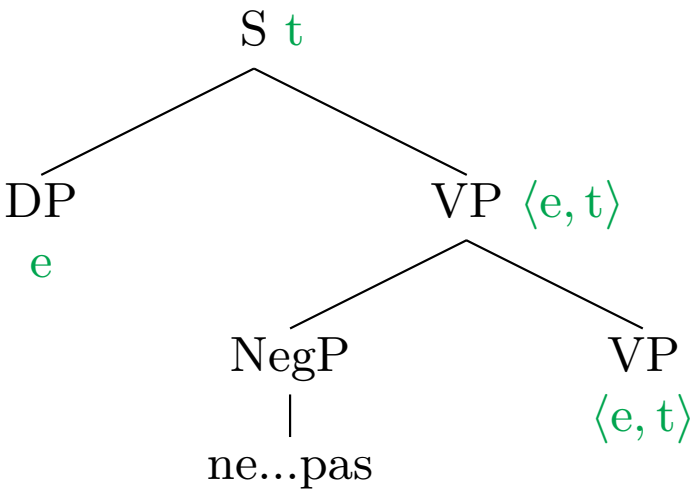
(2) Alice ne dort pas.



Exemples

La négation

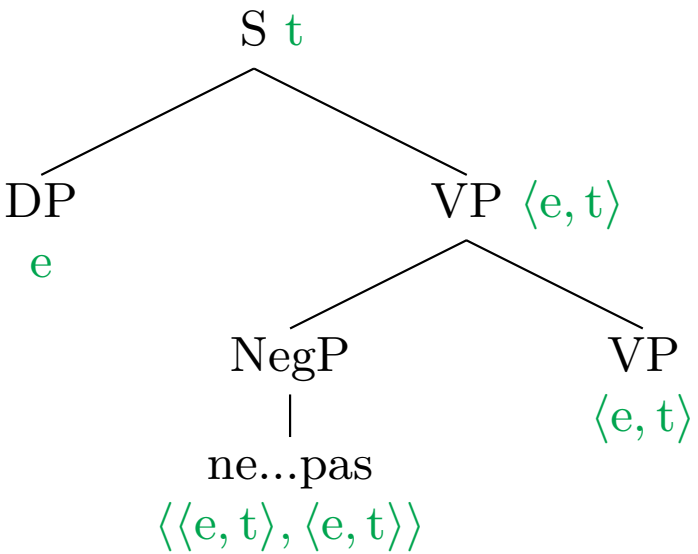
(2) Alice ne dort pas.



Exemples

La négation

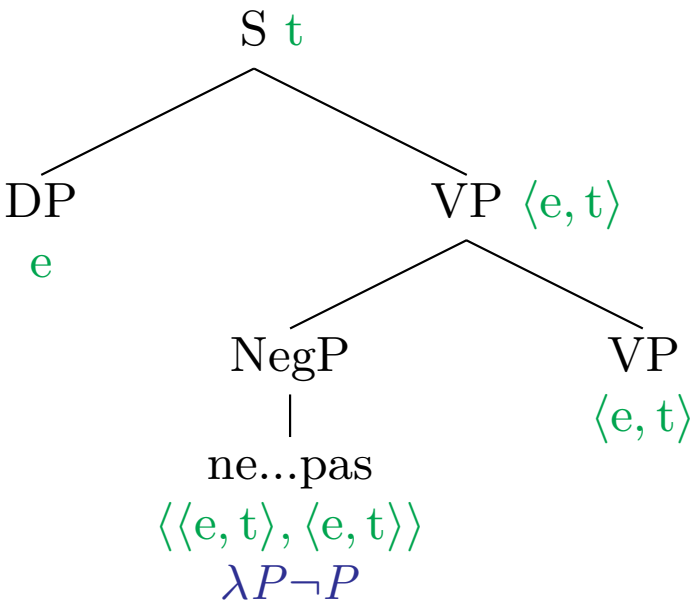
(2) Alice ne dort pas.



Exemples

La négation

(2) Alice ne dort pas.

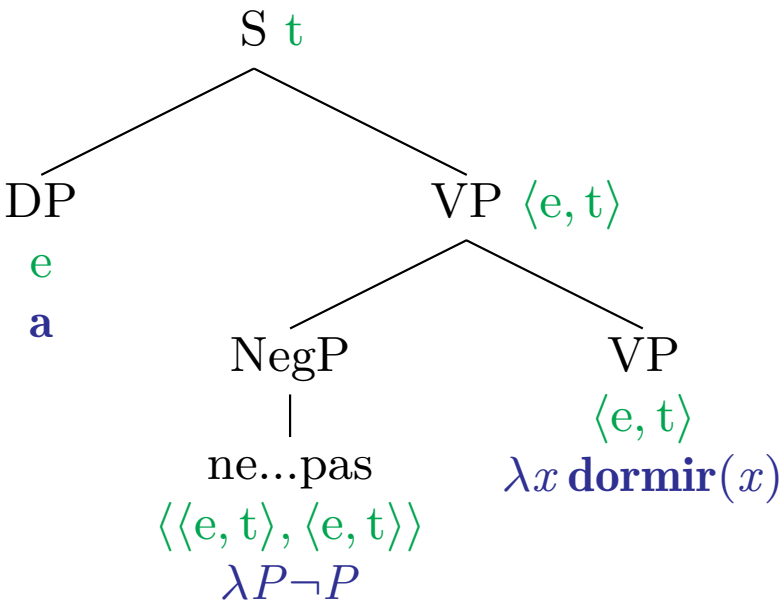


1er essai : NegP $\rightsquigarrow \lambda P \neg P$

Exemples

La négation

(2) Alice ne dort pas.

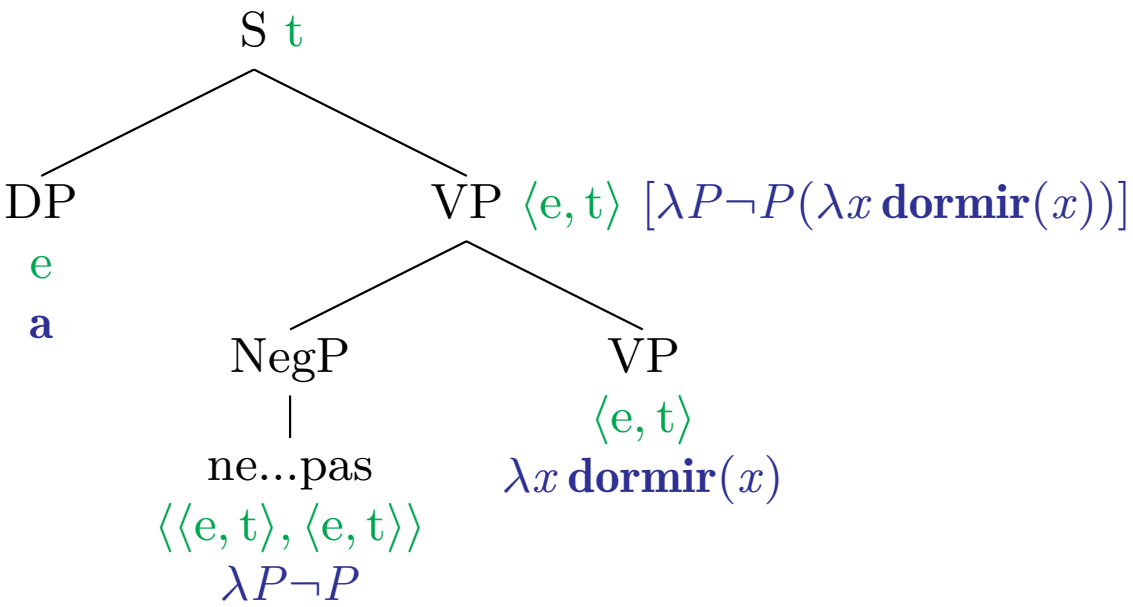


1er essai : NegP $\rightsquigarrow \lambda P \neg P$

Exemples

La négation

(2) Alice ne dort pas.

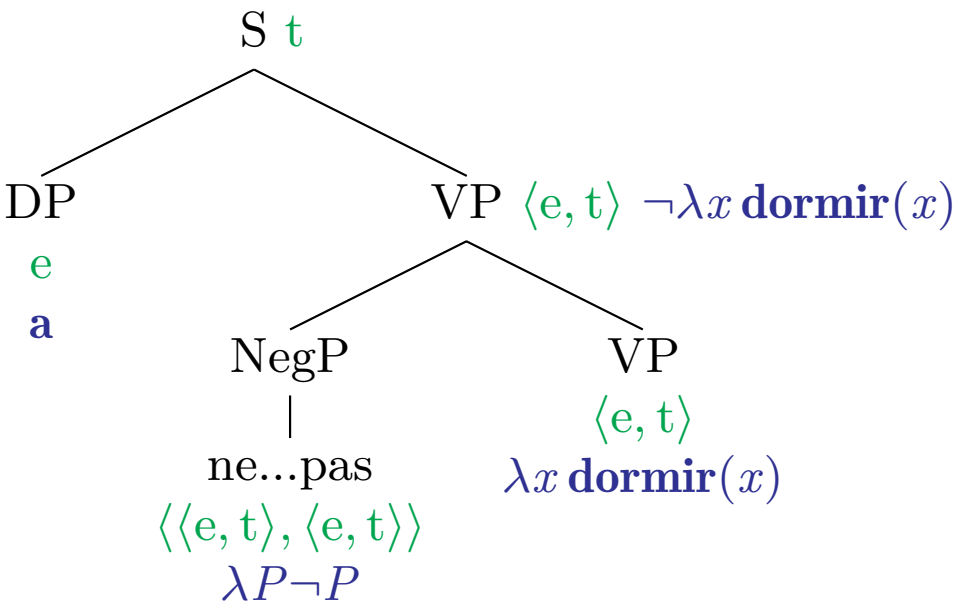


1er essai : NegP $\rightsquigarrow \lambda P \neg P$

Exemples

La négation

(2) Alice ne dort pas.

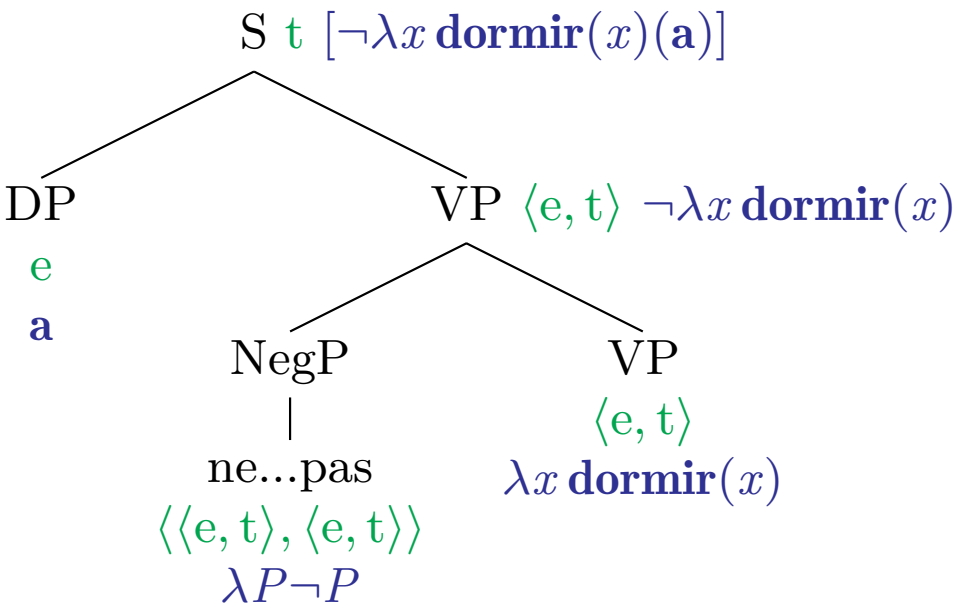


1er essai : NegP $\rightsquigarrow \lambda P \neg P$

Exemples

La négation

(2) Alice ne dort pas.

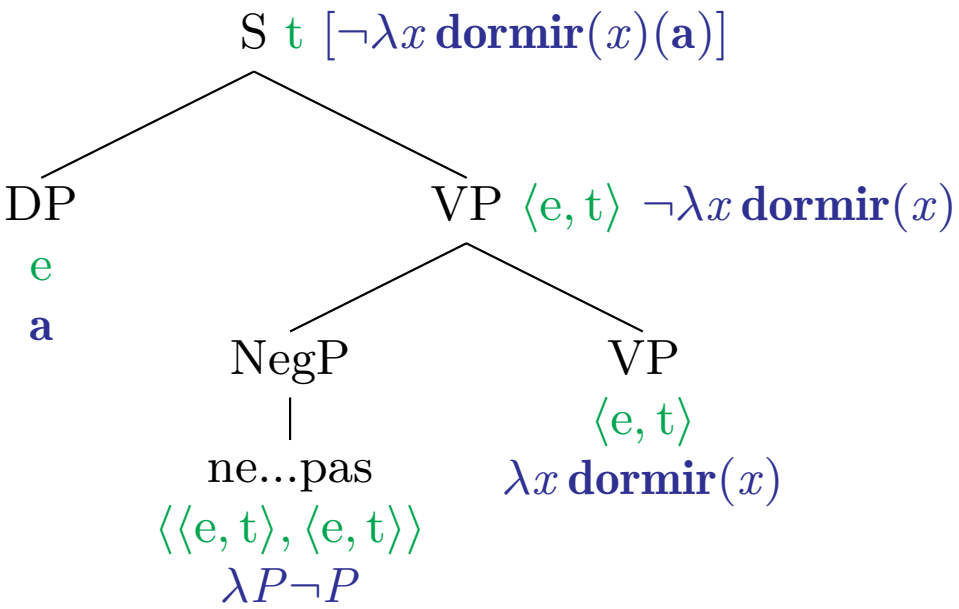


1er essai : NegP $\rightsquigarrow \lambda P \neg P$

Exemples

La négation

(2) Alice ne dort pas.

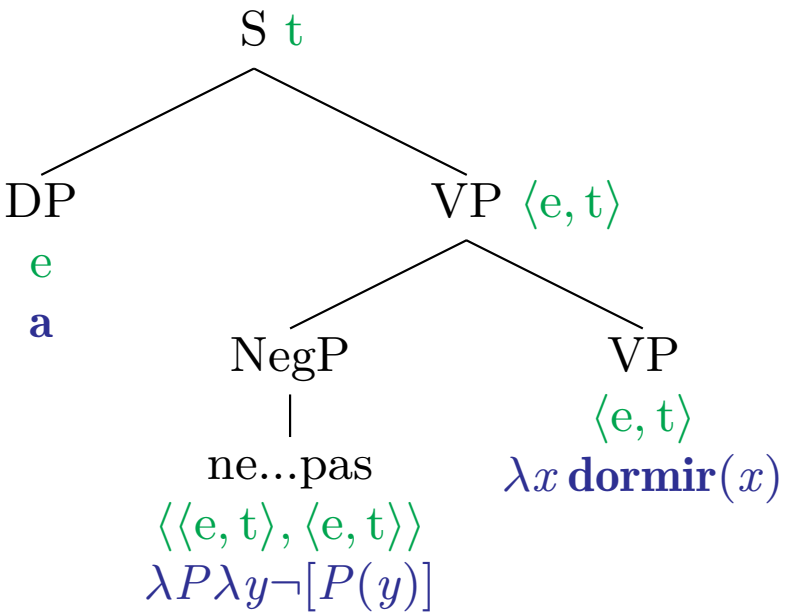


1er essai : NegP $\rightsquigarrow \lambda P \neg P$ P est de type $\langle e, t\rangle$!! Pas le droit d'écrire $\neg P$

Exemples

La négation

(2) Alice ne dort pas.

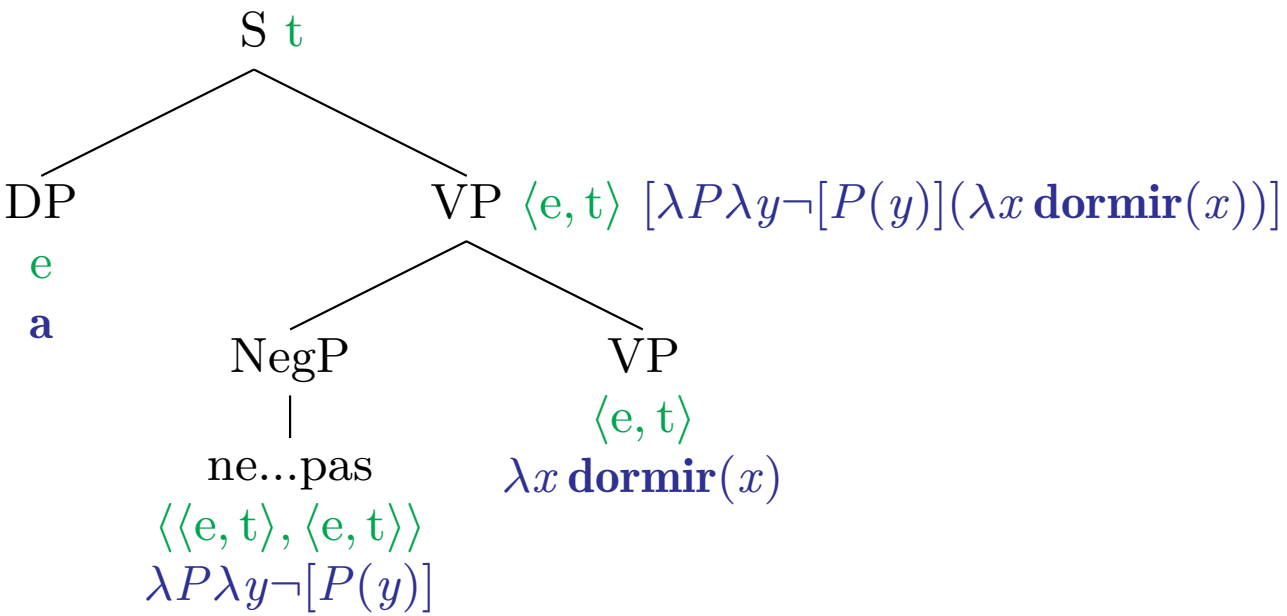


Version correcte : NegP $\rightsquigarrow \lambda P \lambda y \neg [P(y)]$

Exemples

La négation

(2) Alice ne dort pas.

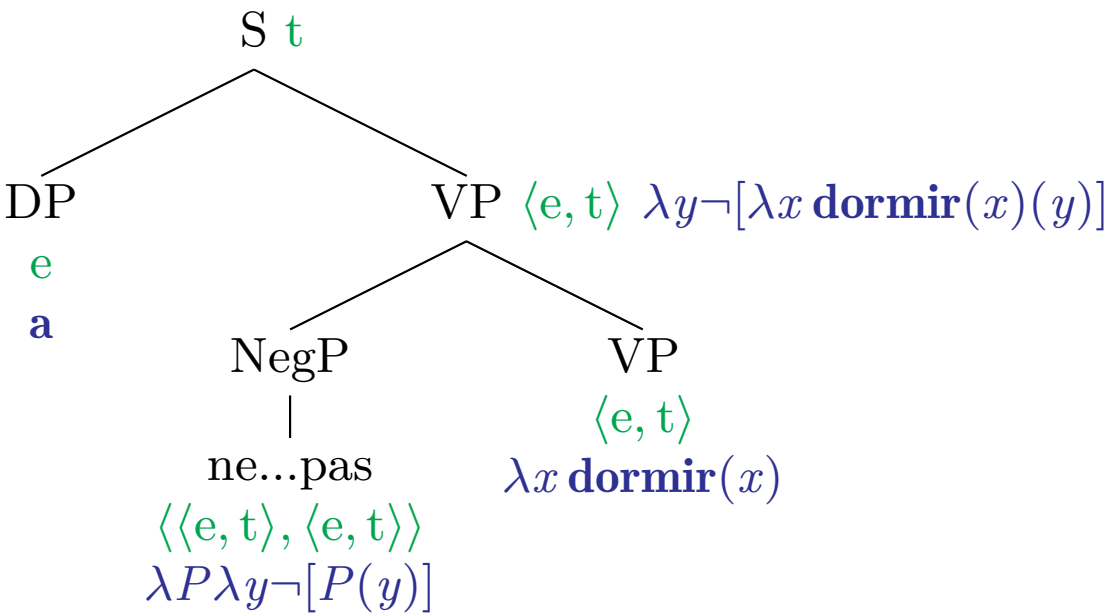


Version correcte : NegP $\rightsquigarrow \lambda P \lambda y \neg [P(y)]$

Exemples

La négation

(2) Alice ne dort pas.

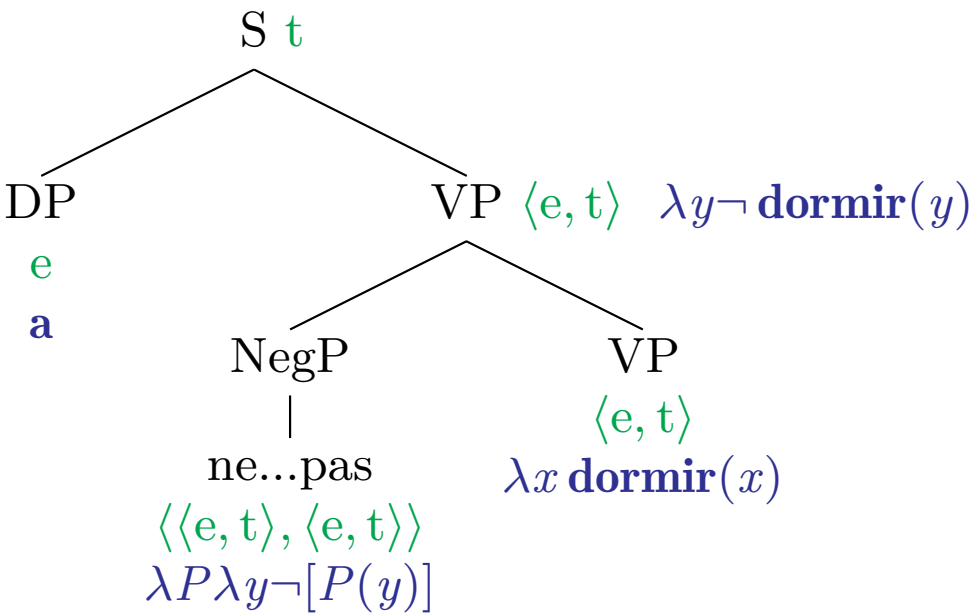


Version correcte : NegP $\rightsquigarrow \lambda P \lambda y \neg [P(y)]$

Exemples

La négation

(2) Alice ne dort pas.

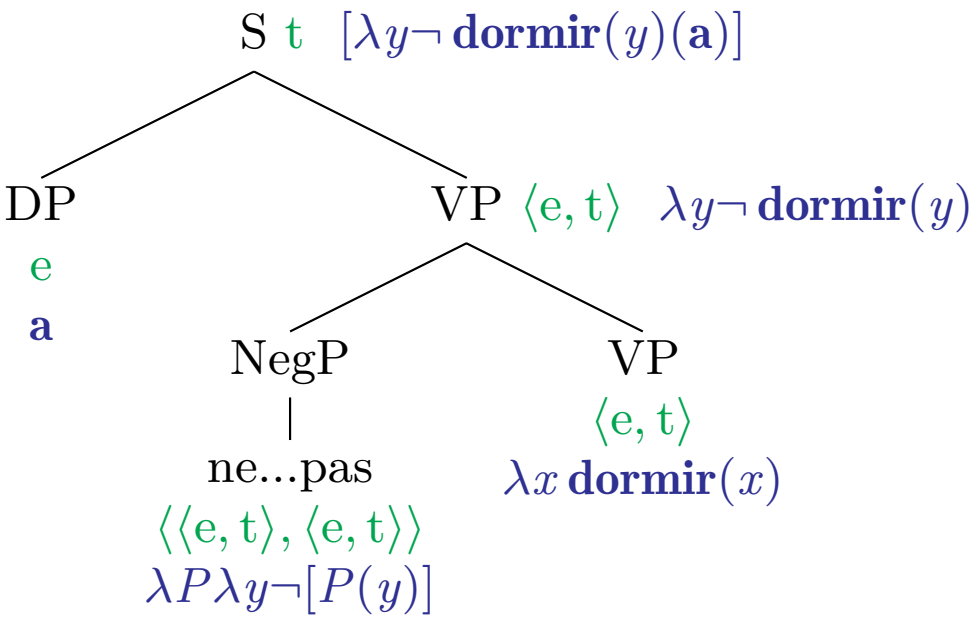


Version correcte : NegP $\rightsquigarrow \lambda P \lambda y \neg [P(y)]$

Exemples

La négation

(2) Alice ne dort pas.

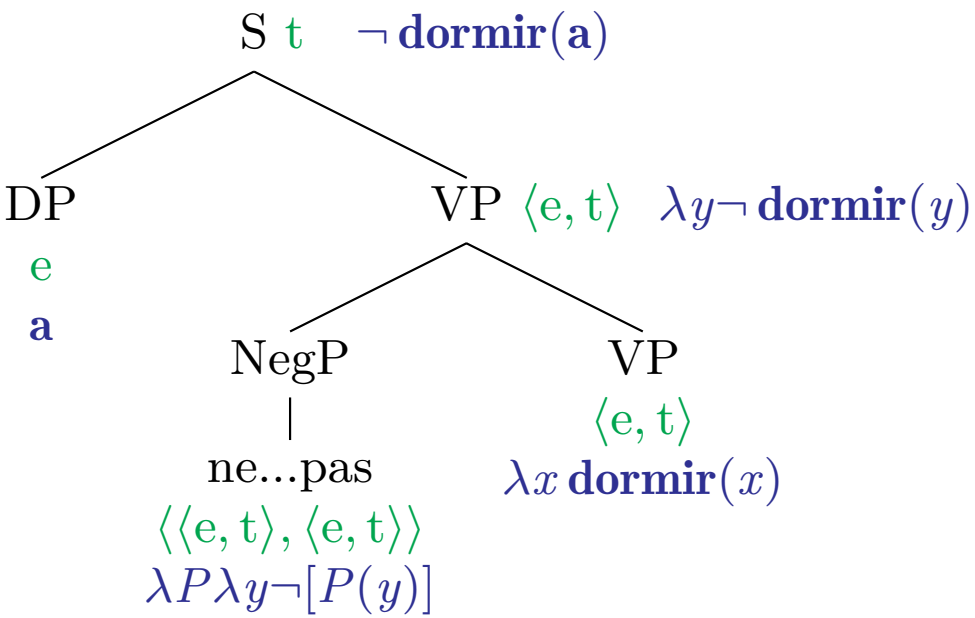


Version correcte : NegP $\rightsquigarrow \lambda P \lambda y \neg [P(y)]$

Exemples

La négation

(2) Alice ne dort pas.



Version correcte : NegP $\rightsquigarrow \lambda P \lambda y \neg [P(y)]$

Adjectifs épithètes, modifieurs de N

$$[\![tigre]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \mathbf{tigre}(x)]\!]^{\mathcal{M},w,g} =$$

Adjectifs épithètes, modifieurs de N

$[\![tigre]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \mathbf{tigre}(x)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les tigres de w.

Adjectifs épithètes, modifieurs de N

$[\![\textit{tigre}]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \mathbf{tigre}(x)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les tigres de w.
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y\, \mathbf{édenté}(y)]\!]^{\mathcal{M},w,g} =$

Adjectifs épithètes, modifieurs de N

$[\![\textit{tigre}]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \mathbf{tigre}(x)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les tigres de w.
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y\, \mathbf{édenté}(y)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus édentés de w.

Adjectifs épithètes, modifieurs de N

$[\![\textit{tigre}]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \mathbf{tigre}(x)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les tigres de w.
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y\, \mathbf{édenté}(y)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus édentés de w.
$[\![\textit{tigre édenté}]\!]^{\mathcal{M},w,g} =$

Adjectifs épithètes, modifieurs de N

$[\![\textit{tigre}]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \mathbf{tigre}(x)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les tigres de w.
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y\, \mathbf{édenté}(y)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus édentés de w.
$[\![\textit{tigre édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \mathbf{tigre}(x)]\!]^{\mathcal{M},w,g} \cap [\![\lambda y\, \mathbf{édenté}(y)]\!]^{\mathcal{M},w,g}$ $\langle \mathrm{e}, \mathrm{t} \rangle$

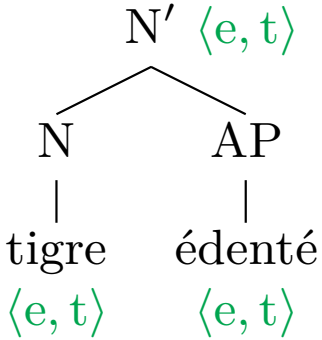
Adjectifs épithètes, modifieurs de N

$[\![\textit{tigre}]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \mathbf{tigre}(x)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les tigres de w.
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y\, \mathbf{édenté}(y)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus édentés de w.
$[\![\textit{tigre édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y[\mathbf{tigre}(y) \wedge \mathbf{édenté}(y)]]\!]^{\mathcal{M},w,g}$ $\langle \mathrm{e}, \mathrm{t} \rangle$

Adjectifs épithètes, modifieurs de N

$[\![\textit{tigre}]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \mathbf{tigre}(x)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les tigres de w.
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y\, \mathbf{édenté}(y)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus édentés de w.
$[\![\textit{tigre édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y[\mathbf{tigre}(y) \wedge \mathbf{édenté}(y)]]\!]^{\mathcal{M},w,g}$ $\langle e, t\rangle$

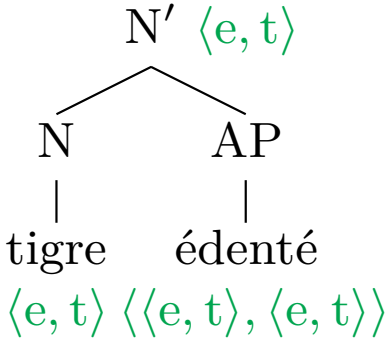
Mais ici les types ne se combinent pas par application fonctionnelle :



Adjectifs épithètes, modifieurs de N

$[\![\textit{tigre}]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \mathbf{tigre}(x)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les tigres de w.
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y\, \mathbf{édenté}(y)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus édentés de w.
$[\![\textit{tigre édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y[\mathbf{tigre}(y) \wedge \mathbf{édenté}(y)]]\!]^{\mathcal{M},w,g}$ $\langle e, t\rangle$

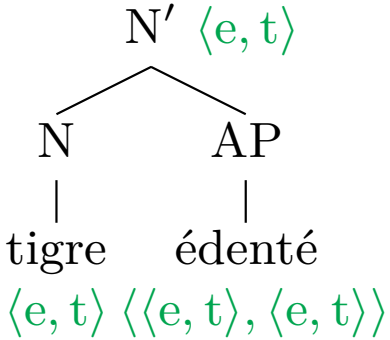
Mais ici les types ne se combinent pas par application fonctionnelle :



Si un AP modifie un N, son type devient $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$: c'est une fonction qui attend un N et qui produit un N′ ([N AP]).

Adjectifs épithètes, modifieurs de N

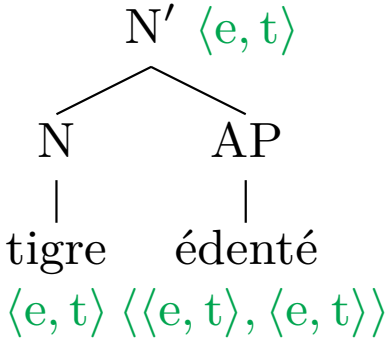
$[\![\textit{tigre}]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \textbf{tigre}(x)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les tigres de w.
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y\, \textbf{édenté}(y)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus édentés de w.
$[\![\textit{tigre édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y[\textbf{tigre}(y) \wedge \textbf{édenté}(y)]]\!]^{\mathcal{M},w,g}$ $\langle e, t\rangle$
Mais ici les types ne se combinent pas par application fonctionnelle :



Si un AP modifie un N, son type devient $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$: c'est une fonction qui attend un N et qui produit un N′ ([N AP]).
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda P \lambda y[[P(y)] \wedge \textbf{édenté}(y)]]\!]^{\mathcal{M},w,g}$
=

Adjectifs épithètes, modifieurs de N

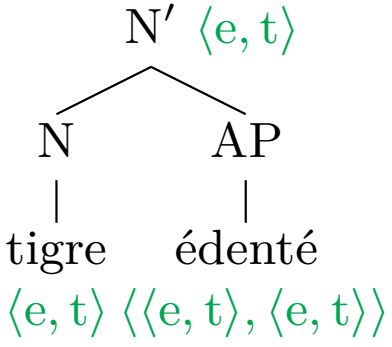
$[\![\textit{tigre}]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \textbf{tigre}(x)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les tigres de w.
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y\, \textbf{édenté}(y)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus édentés de w.
$[\![\textit{tigre édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y[\textbf{tigre}(y) \wedge \textbf{édenté}(y)]]\!]^{\mathcal{M},w,g}$ $\langle e, t\rangle$
Mais ici les types ne se combinent pas par application fonctionnelle :



Si un AP modifie un N, son type devient $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$: c'est une fonction qui attend un N et qui produit un N′ ([N AP]).
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda P \lambda y[[P(y)] \wedge \textbf{édenté}(y)]]\!]^{\mathcal{M},w,g}$
= la fonction qui attend un ensemble (P) et qui retourne l'*intersection* de $[\![P]\!]^{\mathcal{M},w,g}$ avec $[\![\textbf{édenté}]\!]^{\mathcal{M},w,g}$.

Adjectifs épithètes, modifieurs de N

$[\![\textit{tigre}]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \textbf{tigre}(x)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les tigres de w.
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y\, \textbf{édenté}(y)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus édentés de w.
$[\![\textit{tigre édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y[\textbf{tigre}(y) \wedge \textbf{édenté}(y)]]\!]^{\mathcal{M},w,g}$ $\langle e, t\rangle$
Mais ici les types ne se combinent pas par application fonctionnelle :

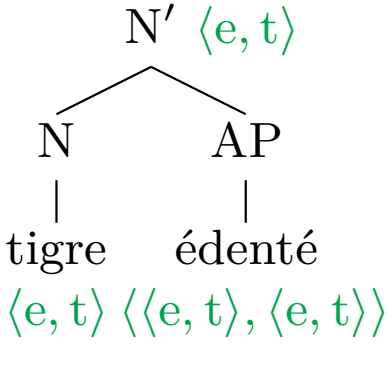


Si un AP modifie un N, son type devient $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$: c'est une fonction qui attend un N et qui produit un N′ ([N AP]).
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda P \lambda y[[P(y)] \wedge \textbf{édenté}(y)]]\!]^{\mathcal{M},w,g}$
= la fonction qui attend un ensemble (P) et qui retourne l'*intersection* de $[\![P]\!]^{\mathcal{M},w,g}$ avec $[\![\textbf{édenté}]\!]^{\mathcal{M},w,g}$.

$\textit{tigre édenté} \rightsquigarrow [\lambda P \lambda y[[P(y)] \wedge \textbf{édenté}(y)](\lambda x\, \textbf{tigre}(x))]$

Adjectifs épithètes, modifieurs de N

$[\![\textit{tigre}]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \mathbf{tigre}(x)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les tigres de w.
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y\, \mathbf{édenté}(y)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus édentés de w.
$[\![\textit{tigre édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y[\mathbf{tigre}(y) \wedge \mathbf{édenté}(y)]]\!]^{\mathcal{M},w,g}$ $\langle e, t\rangle$
Mais ici les types ne se combinent pas par application fonctionnelle :

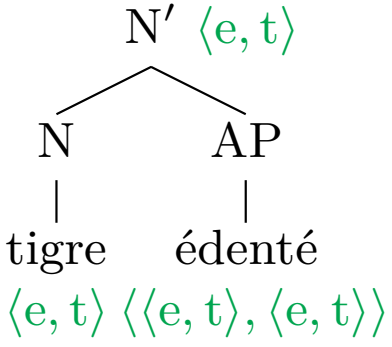


Si un AP modifie un N, son type devient $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$: c'est une fonction qui attend un N et qui produit un N′ ([N AP]).
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda P \lambda y[[P(y)] \wedge \mathbf{édenté}(y)]]\!]^{\mathcal{M},w,g}$
= la fonction qui attend un ensemble (P) et qui retourne l'*intersection* de $[\![P]\!]^{\mathcal{M},w,g}$ avec $[\![\mathbf{édenté}]\!]^{\mathcal{M},w,g}$.

$\textit{tigre édenté} \rightsquigarrow \lambda y[[\lambda x\, \mathbf{tigre}(x)(y)] \wedge \mathbf{édenté}(y)]$

Adjectifs épithètes, modifieurs de N

$[\![\textit{tigre}]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \textbf{tigre}(x)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les tigres de w.
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y\, \textbf{édenté}(y)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus édentés de w.
$[\![\textit{tigre édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y[\textbf{tigre}(y) \wedge \textbf{édenté}(y)]]\!]^{\mathcal{M},w,g}$ $\langle e, t\rangle$
Mais ici les types ne se combinent pas par application fonctionnelle :

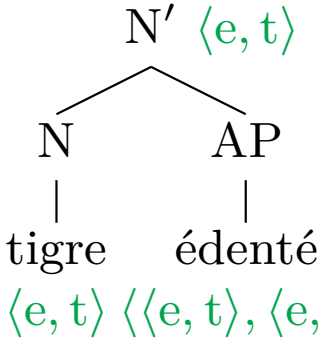


Si un AP modifie un N, son type devient $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$: c'est une fonction qui attend un N et qui produit un N′ ([N AP]).
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda P \lambda y[[P(y)] \wedge \textbf{édenté}(y)]]\!]^{\mathcal{M},w,g}$
= la fonction qui attend un ensemble (P) et qui retourne l'*intersection* de $[\![P]\!]^{\mathcal{M},w,g}$ avec $[\![\textbf{édenté}]\!]^{\mathcal{M},w,g}$.

$\textit{tigre édenté} \rightsquigarrow \lambda y[\textbf{tigre}(y) \wedge \textbf{édenté}(y)]$

Adjectifs épithètes, modifieurs de N

$[\![\textit{tigre}]\!]^{\mathcal{M},w,g} = [\![\lambda x\, \textbf{tigre}(x)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les tigres de w.
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y\, \textbf{édenté}(y)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus édentés de w.
$[\![\textit{tigre édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda y[\textbf{tigre}(y) \wedge \textbf{édenté}(y)]]\!]^{\mathcal{M},w,g}$ $\langle e, t\rangle$
Mais ici les types ne se combinent pas par application fonctionnelle :



Si un AP modifie un N, son type devient $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$: c'est une fonction qui attend un N et qui produit un N′ ([N AP]).
$[\![\textit{édenté}]\!]^{\mathcal{M},w,g} = [\![\lambda P \lambda y[[P(y)] \wedge \textbf{édenté}(y)]]\!]^{\mathcal{M},w,g}$
= la fonction qui attend un ensemble (P) et qui retourne l'*intersection* de $[\![P]\!]^{\mathcal{M},w,g}$ avec $[\![\textbf{édenté}]\!]^{\mathcal{M},w,g}$.

$\textit{tigre édenté} \rightsquigarrow \lambda y[\textbf{tigre}(y) \wedge \textbf{édenté}(y)]$

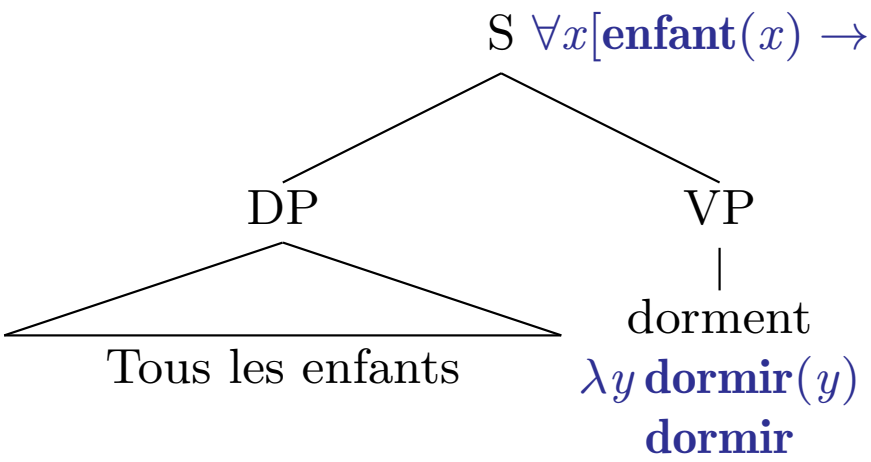
Règle syntaxe–sémantique spéciale :

	N′	→ N	AP
$[\lambda P \lambda y[[P(y)] \wedge \alpha(y)](\beta)]$	←	β	α

(équivalent à la règle PM de Heim & Kratzer (1997))

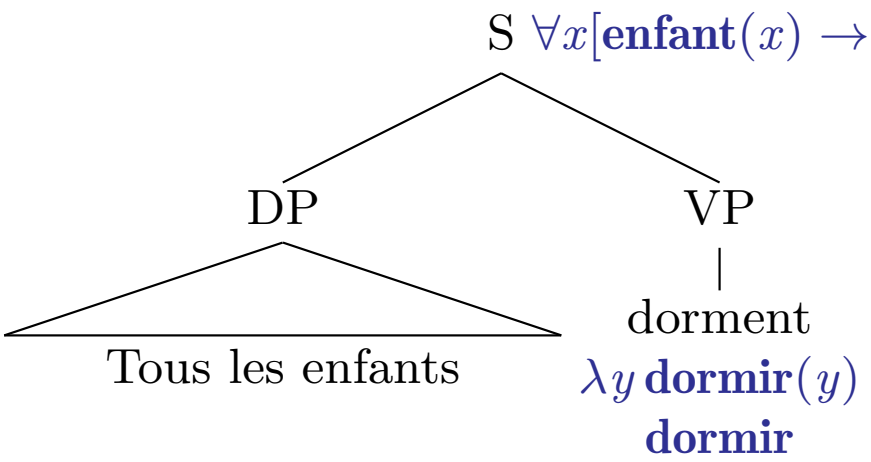
Analyse générale des DP

DP et VP sont dans un bateau



Analyse générale des DP

DP et VP sont dans un bateau

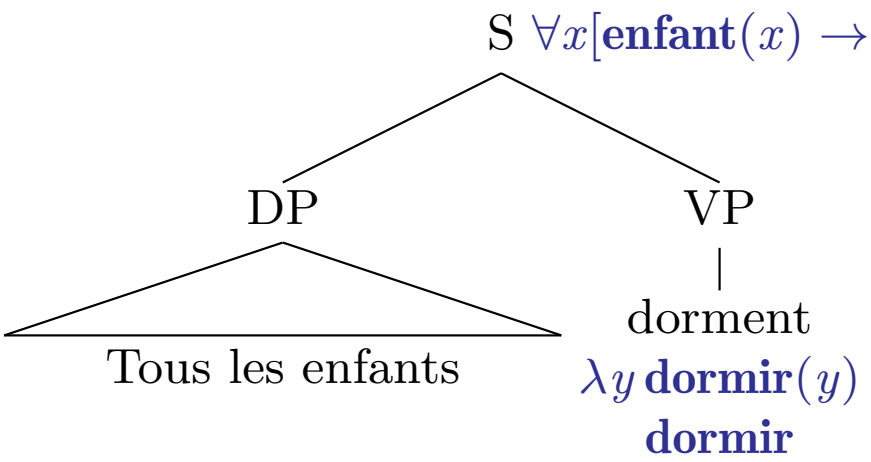


S = DP + VP

DP + VP $\rightsquigarrow$ $\forall x[\mathbf{enfant}(x) \rightarrow \mathbf{dormir}(x)]$

Analyse générale des DP

DP et VP sont dans un bateau

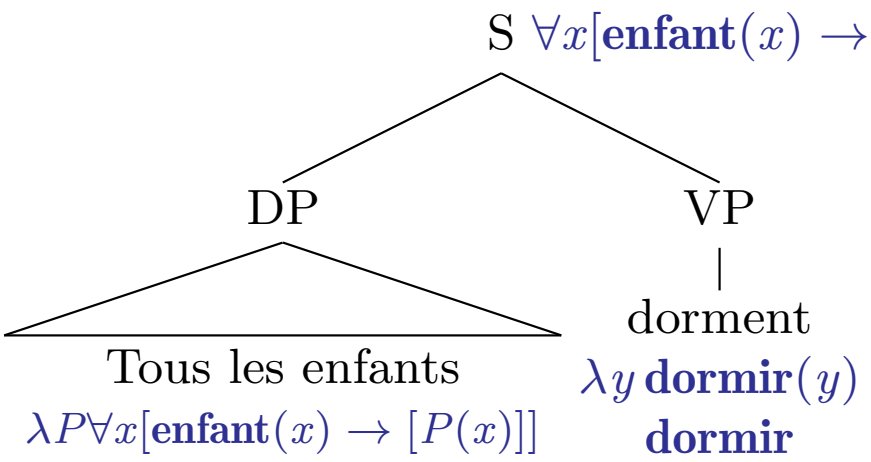


DP = S − VP

DP $\rightsquigarrow$ $\forall x[\mathbf{enfant}(x) \rightarrow \quad (x)]$

Analyse générale des DP

DP et VP sont dans un bateau

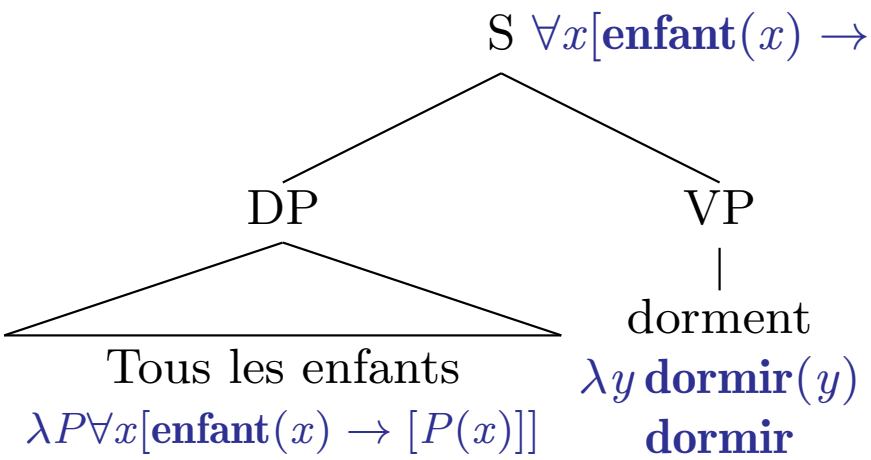


$\text{DP} = \text{S} - \text{VP}$

$\text{DP} \rightsquigarrow \lambda P \forall x[\textbf{enfant}(x) \rightarrow P(x)]$ P de type $\langle \text{e}, \text{t} \rangle$

Analyse générale des DP

DP et VP sont dans un bateau



DP = S − VP

DP $\rightsquigarrow \lambda P \forall x[\textbf{enfant}(x) \rightarrow P(x)]$ P de type $\langle \mathrm{e}, \mathrm{t} \rangle$

Le DP est de type $\langle\langle \mathrm{e}, \mathrm{t} \rangle, \mathrm{t} \rangle$, il est la fonction principale de la phrase.

Analyse générale des DP

$\langle\langle e, t\rangle, t\rangle$

Tous les DP sont de type $\langle\langle e, t\rangle, t\rangle$.

- *Tous les enfants* $\rightsquigarrow \lambda P \forall x[\mathbf{enfant}(x) \rightarrow [P(x)]]$
- *Un enfant* $\rightsquigarrow \lambda P \exists x[\mathbf{enfant}(x) \wedge [P(x)]]$
- *Alice* $\rightsquigarrow \lambda P[P(\mathbf{a})]$

Analyse générale des DP

$\langle\langle e, t\rangle, t\rangle$

Tous les DP sont de type $\langle\langle e, t\rangle, t\rangle$. (Ensembles de propriétés)

- *Tous les enfants* $\rightsquigarrow \lambda P \forall x[\mathbf{enfant}(x) \rightarrow [P(x)]]$
- *Un enfant* $\rightsquigarrow \lambda P \exists x[\mathbf{enfant}(x) \wedge [P(x)]]$
- *Alice* $\rightsquigarrow \lambda P[P(\mathbf{a})]$
  $[\![\lambda P[P(\mathbf{a})]]\!]^{\mathcal{M},w,g} \approx$ ensemble de tous les prédicats/propriétés qui s'appliquent à Alice.

Analyse générale des DP

$\langle\langle e, t\rangle, t\rangle$

Tous les DP sont de type $\langle\langle e, t\rangle, t\rangle$. (Ensembles de propriétés)

- *Tous les enfants* $\rightsquigarrow \lambda P \forall x[\textbf{enfant}(x) \rightarrow [P(x)]]$

- *Un enfant* $\rightsquigarrow \lambda P \exists x[\textbf{enfant}(x) \wedge [P(x)]]$
  $[\![\lambda P \exists x[\textbf{enfant}(x) \wedge [P(x)]]]\!]^{\mathcal{M},w,g} \approx$ ensemble des propriétés qui s'appliquent à au moins un enfant.
- *Alice* $\rightsquigarrow \lambda P[P(\textbf{a})]$
  $[\![\lambda P[P(\textbf{a})]]\!]^{\mathcal{M},w,g} \approx$ ensemble de tous les prédicats/propriétés qui s'appliquent à Alice.

Analyse générale des DP

$\langle\langle e, t\rangle, t\rangle$

Tous les DP sont de type $\langle\langle e, t\rangle, t\rangle$. (Ensembles de propriétés)

- *Tous les enfants* $\rightsquigarrow \lambda P \forall x[\mathbf{enfant}(x) \rightarrow [P(x)]]$
  $[\![\lambda P \forall x[\mathbf{enfant}(x) \rightarrow [P(x)]]]\!]^{\mathcal{M},w,g} \approx$ ensemble des propriétés qui s'appliquent à tous les enfants.
- *Un enfant* $\rightsquigarrow \lambda P \exists x[\mathbf{enfant}(x) \wedge [P(x)]]$
  $[\![\lambda P \exists x[\mathbf{enfant}(x) \wedge [P(x)]]]\!]^{\mathcal{M},w,g} \approx$ ensemble des propriétés qui s'appliquent à au moins un enfant.
- *Alice* $\rightsquigarrow \lambda P[P(\mathbf{a})]$
  $[\![\lambda P[P(\mathbf{a})]]\!]^{\mathcal{M},w,g} \approx$ ensemble de tous les prédicats/propriétés qui s'appliquent à Alice.

Analyse générale des DP

$\langle\langle e, t\rangle, t\rangle$

Tous les DP sont de type $\langle\langle e, t\rangle, t\rangle$. (Ensembles de propriétés)

- *Tous les enfants* $\rightsquigarrow \lambda P \forall x[\mathbf{enfant}(x) \rightarrow [P(x)]]$
  $[\![\lambda P \forall x[\mathbf{enfant}(x) \rightarrow [P(x)]]]\!]^{\mathcal{M},w,g} \approx$ ensemble des propriétés qui s'appliquent à tous les enfants.
- *Un enfant* $\rightsquigarrow \lambda P \exists x[\mathbf{enfant}(x) \wedge [P(x)]]$
  $[\![\lambda P \exists x[\mathbf{enfant}(x) \wedge [P(x)]]]\!]^{\mathcal{M},w,g} \approx$ ensemble des propriétés qui s'appliquent à au moins un enfant.
- *Alice* $\rightsquigarrow \lambda P[P(\mathbf{a})]$
  $[\![\lambda P[P(\mathbf{a})]]\!]^{\mathcal{M},w,g} \approx$ ensemble de tous les prédicats/propriétés qui s'appliquent à Alice.

$[\lambda P[P(\mathbf{a})](\lambda y\, \mathbf{dormir}(y))] \approx$ la propriété de dormir est une des propriétés possédées par Alice.

Analyse générale des DP

$\langle\langle e, t\rangle, t\rangle$

Tous les DP sont de type $\langle\langle e, t\rangle, t\rangle$. (Ensembles de propriétés)

- *Tous les enfants* $\rightsquigarrow \lambda P \forall x[\mathbf{enfant}(x) \rightarrow [P(x)]]$
  $[\![\lambda P \forall x[\mathbf{enfant}(x) \rightarrow [P(x)]]]\!]^{\mathcal{M},w,g} \approx$ ensemble des propriétés qui s'appliquent à tous les enfants.
- *Un enfant* $\rightsquigarrow \lambda P \exists x[\mathbf{enfant}(x) \wedge [P(x)]]$
  $[\![\lambda P \exists x[\mathbf{enfant}(x) \wedge [P(x)]]]\!]^{\mathcal{M},w,g} \approx$ ensemble des propriétés qui s'appliquent à au moins un enfant.
- *Alice* $\rightsquigarrow \lambda P[P(\mathbf{a})]$
  $[\![\lambda P[P(\mathbf{a})]]\!]^{\mathcal{M},w,g} \approx$ ensemble de tous les prédicats/propriétés qui s'appliquent à Alice.

$[\lambda y\, \mathbf{dormir}(y)(\mathbf{a})] \approx$ Alice vérifie la propriété de dormir

Analyse générale des DP

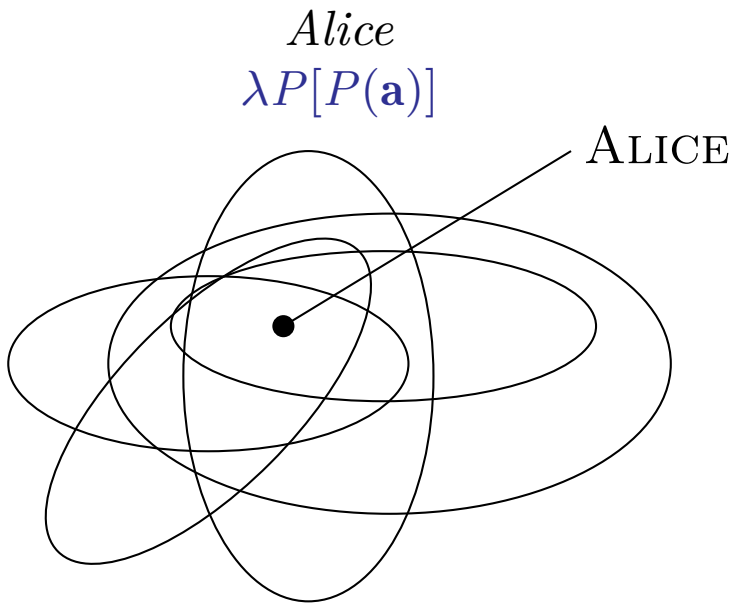
$\langle\langle e, t\rangle, t\rangle$

Tous les DP sont de type $\langle\langle e, t\rangle, t\rangle$. (Ensembles de propriétés)

- *Tous les enfants* $\rightsquigarrow \lambda P \forall x[\mathbf{enfant}(x) \rightarrow [P(x)]]$
  $[\![\lambda P \forall x[\mathbf{enfant}(x) \rightarrow [P(x)]]]\!]^{\mathcal{M},w,g} \approx$ ensemble des propriétés qui s'appliquent à tous les enfants.
- *Un enfant* $\rightsquigarrow \lambda P \exists x[\mathbf{enfant}(x) \wedge [P(x)]]$
  $[\![\lambda P \exists x[\mathbf{enfant}(x) \wedge [P(x)]]]\!]^{\mathcal{M},w,g} \approx$ ensemble des propriétés qui s'appliquent à au moins un enfant.
- *Alice* $\rightsquigarrow \lambda P[P(\mathbf{a})]$
  $[\![\lambda P[P(\mathbf{a})]]\!]^{\mathcal{M},w,g} \approx$ ensemble de tous les prédicats/propriétés qui s'appliquent à Alice.

$\mathbf{dormir}(\mathbf{a}) \approx$ Alice appartient à l'ensemble des dormeurs

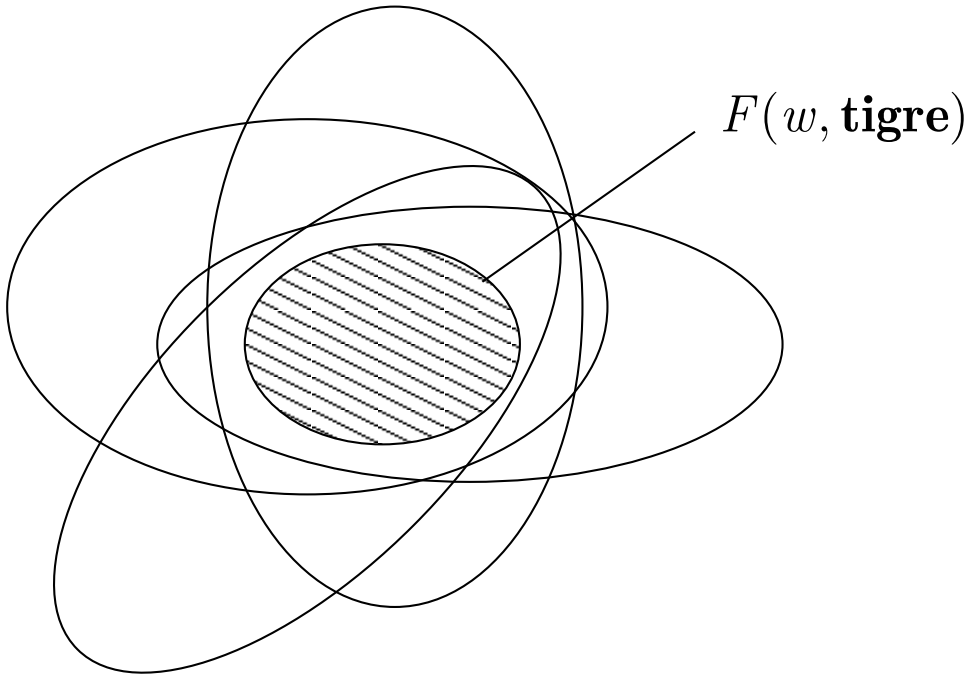
Les DP dénotent des ensembles de propriétés



Ensemble de tous les ensembles qui contiennent ALICE

Les DP dénotent des ensembles de propriétés

tous les tigres
$\lambda P \forall x[\mathbf{tigre}(x) \rightarrow P(x)]$

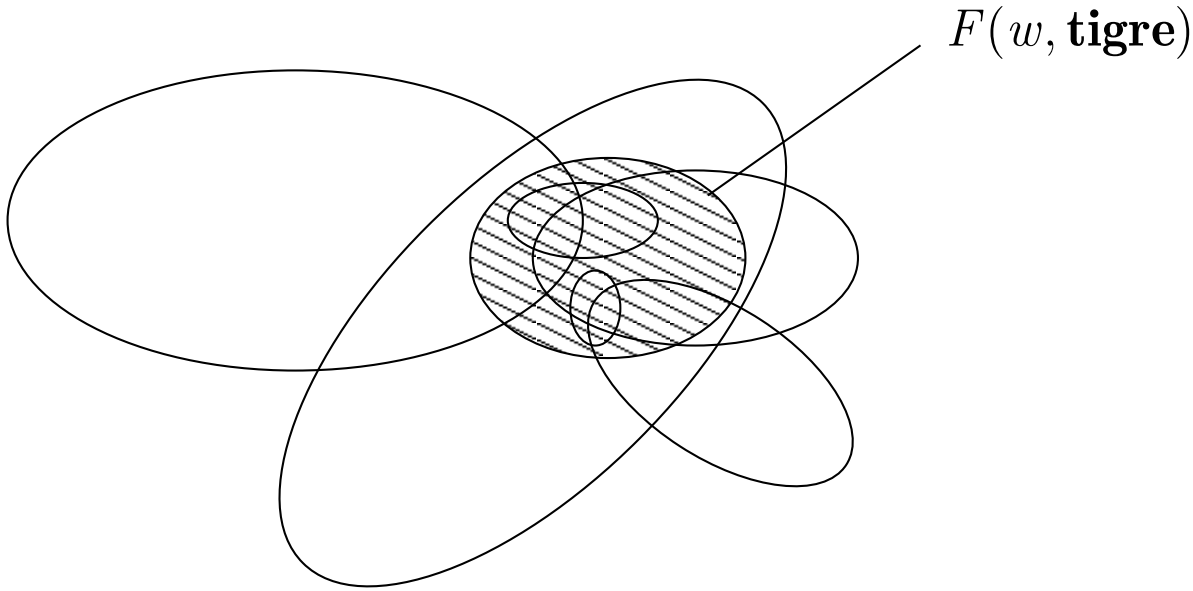


Ensemble de tous les ensembles qui contiennent tous les tigres

Les DP dénotent des ensembles de propriétés

un tigre

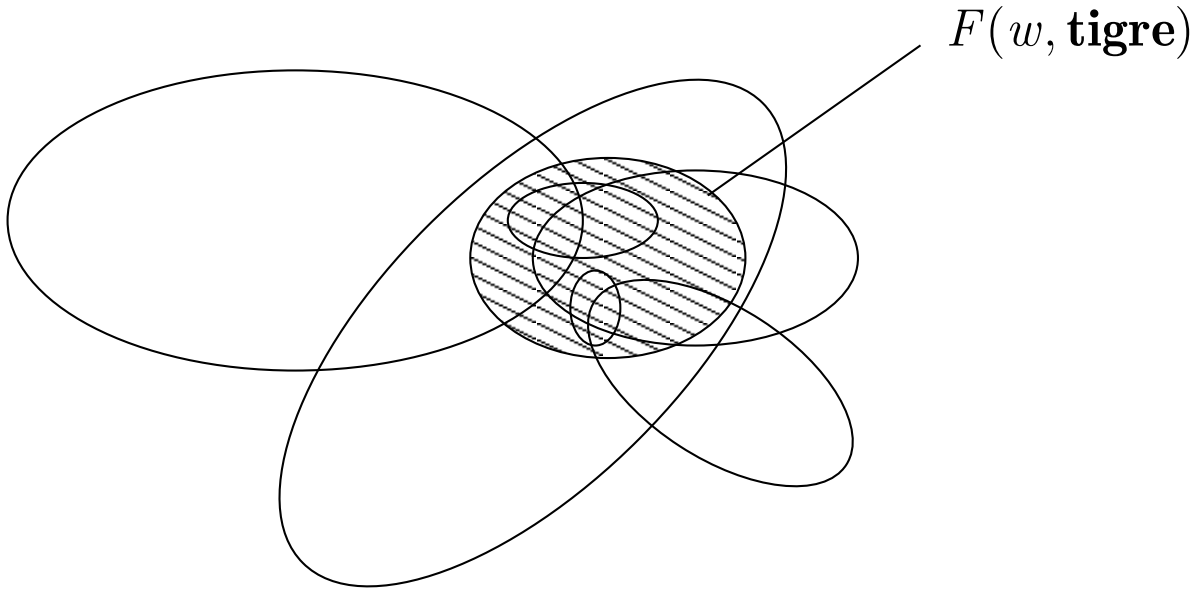
$\lambda P \exists x[\mathbf{tigre}(x) \wedge P(x)]$



Ensemble de tous les ensembles qui contiennent au moins un tigre

Les DP dénotent des ensembles de propriétés

un tigre
$\lambda P \exists x[\mathbf{tigre}(x) \wedge P(x)]$



Ensemble de tous les ensembles qui contiennent au moins un tigre

☞ Les DP ne sont pas référentiels (ils ne sont pas de type e), mais ils ont une dénotation : un ensemble de propriétés extensionnelles.

Déterminants

Les déterminants attendent un N ou NP ($\langle e, t \rangle$) pour former un DP

Déterminants

Les déterminants attendent un N ou NP ($\langle e, t \rangle$) pour former un DP qui lui-même attend un VP ($\langle e, t \rangle$) pour former une S.

Déterminants

Les déterminants attendent un N ou NP ($\langle e, t\rangle$) pour former un DP qui lui-même attend un VP ($\langle e, t\rangle$) pour former une S.
Donc les déterminants sont des fonctions qui renvoient une valeur de vérité si on leur fournit deux propriétés.
Leur type est $\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$; ils posent une relation entre 2 ensembles.

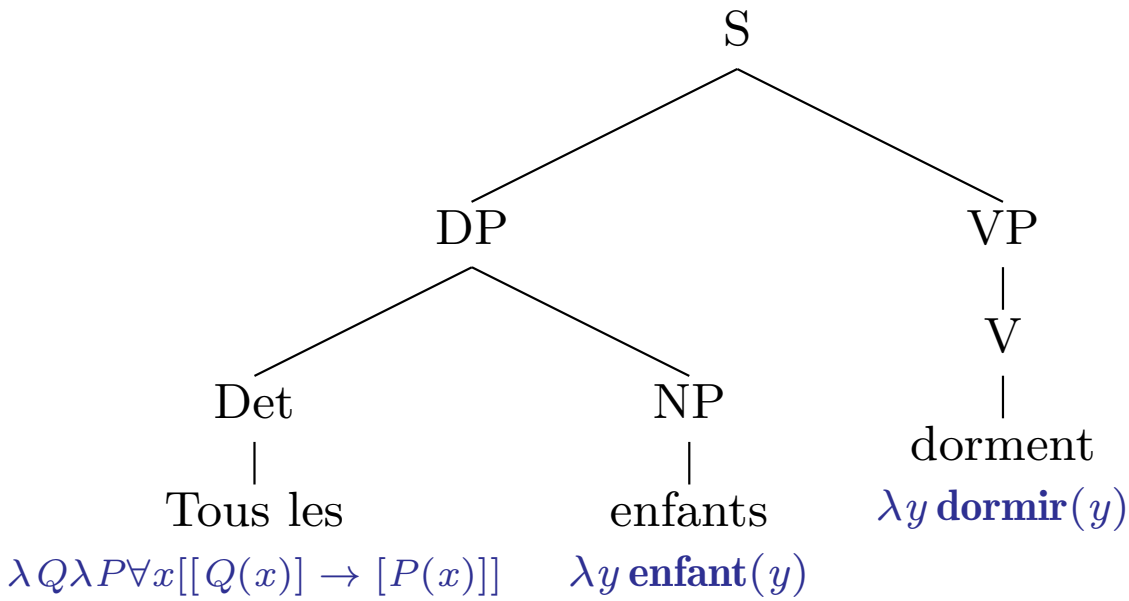
Déterminants

Les déterminants attendent un N ou NP ($\langle e, t\rangle$) pour former un DP qui lui-même attend un VP ($\langle e, t\rangle$) pour former une S.
Donc les déterminants sont des fonctions qui renvoient une valeur de vérité si on leur fournit deux propriétés.
Leur type est $\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$; ils posent une relation entre 2 ensembles.

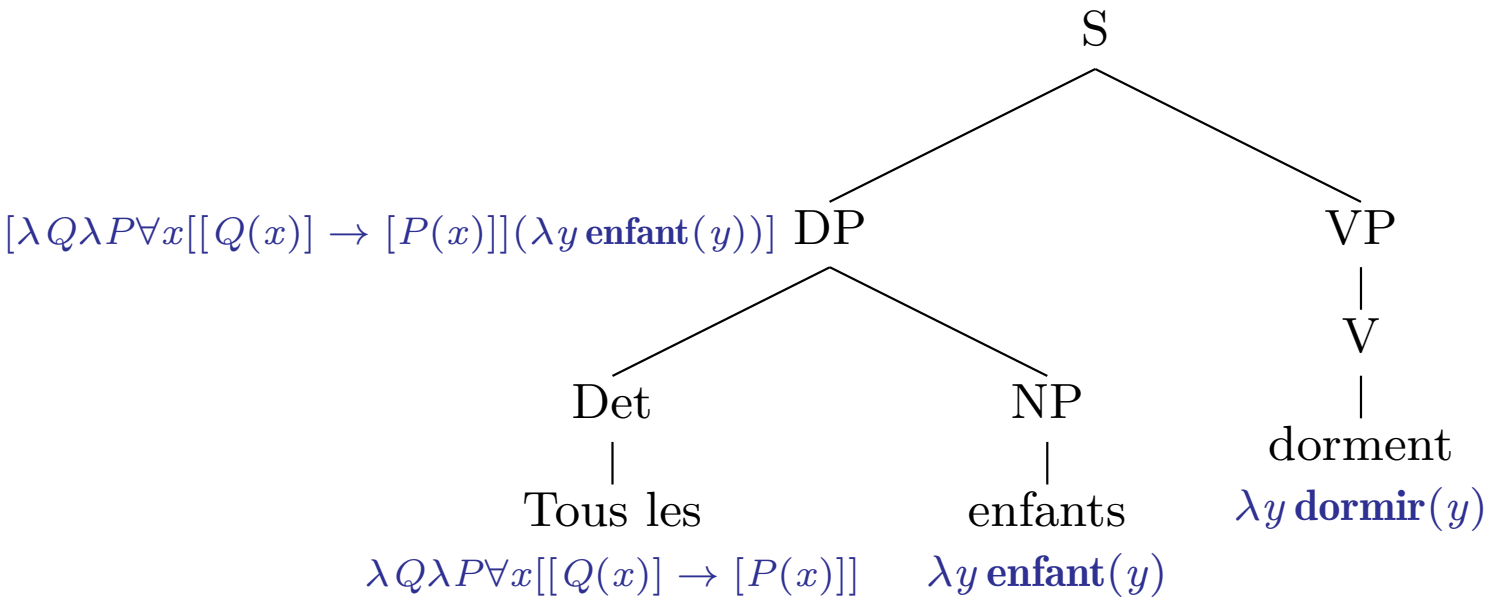
- *tous les* $\rightsquigarrow \lambda Q \lambda P \forall x[[Q(x)] \rightarrow [P(x)]]$;
  - $[\![Q]\!]^{\mathcal{M},w,g} \subset [\![P]\!]^{\mathcal{M},w,g}$
- *un* $\rightsquigarrow \lambda Q \lambda P \exists x[[Q(x)] \wedge [P(x)]]$;
  - $[\![Q]\!]^{\mathcal{M},w,g} \cap [\![P]\!]^{\mathcal{M},w,g} \neq \varnothing$
- *le* $\rightsquigarrow \lambda Q \lambda P[P(\imath x[Q(x)])]$;
  - $[\![Q]\!]^{\mathcal{M},w,g}$ est un singleton et $[\![Q]\!]^{\mathcal{M},w,g} \subset [\![P]\!]^{\mathcal{M},w,g}$

Q : argument NP $\quad$ P : argument VP

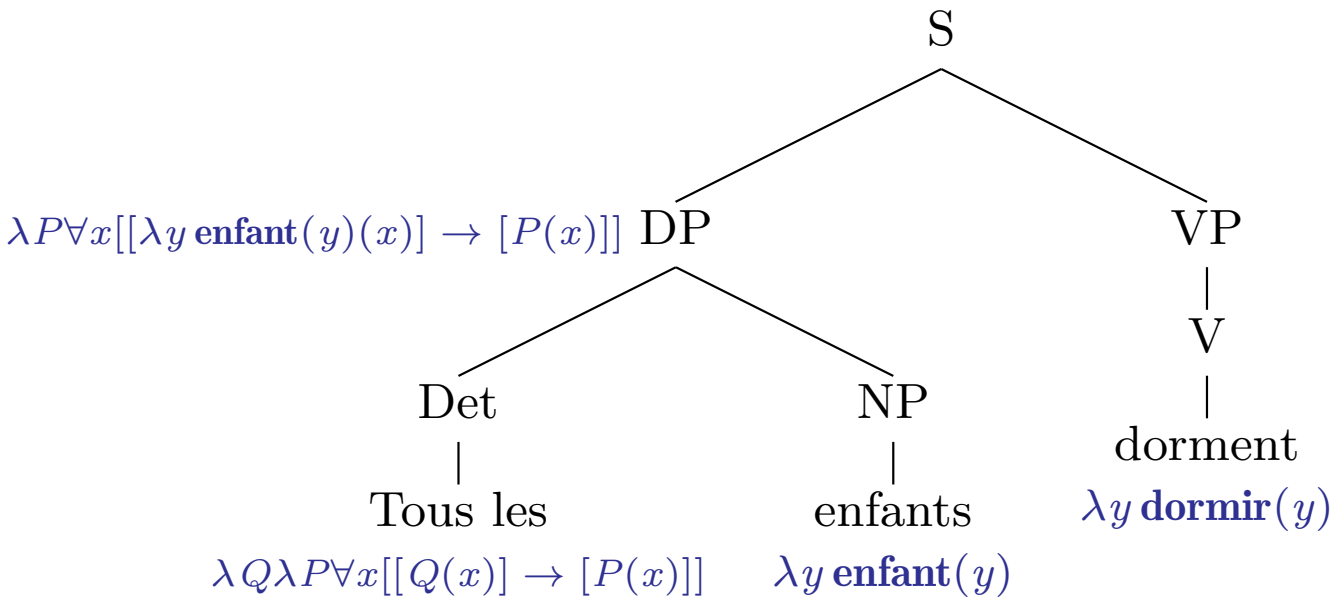
Exemple


S
DP
VP
V
Det
NP
dorment
Tous les
enfants
$\lambda Q \lambda P \forall x[[Q(x)] \rightarrow [P(x)]]$
$\lambda y\, \mathbf{enfant}(y)$
$\lambda y\, \mathbf{dormir}(y)$

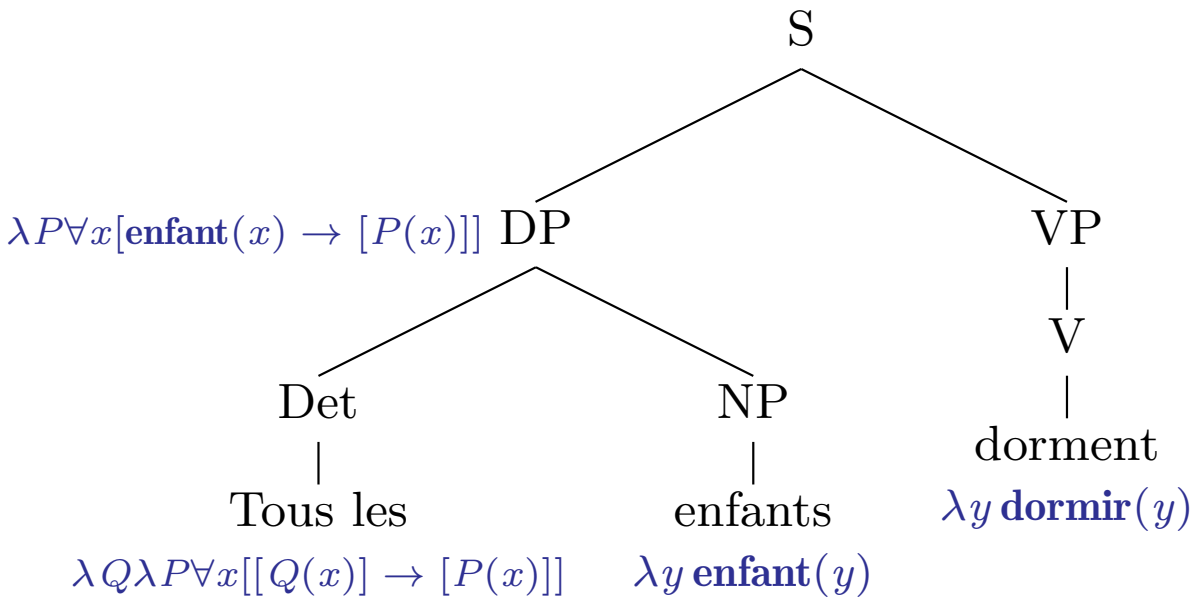
Exemple


S
$[\lambda Q \lambda P \forall x [[Q(x)] \to [P(x)]](\lambda y\, \mathbf{enfant}(y))]$
DP
VP
V
Det
NP
dorment
$\lambda y\, \mathbf{dormir}(y)$
Tous les
enfants
$\lambda Q \lambda P \forall x [[Q(x)] \to [P(x)]]$
$\lambda y\, \mathbf{enfant}(y)$

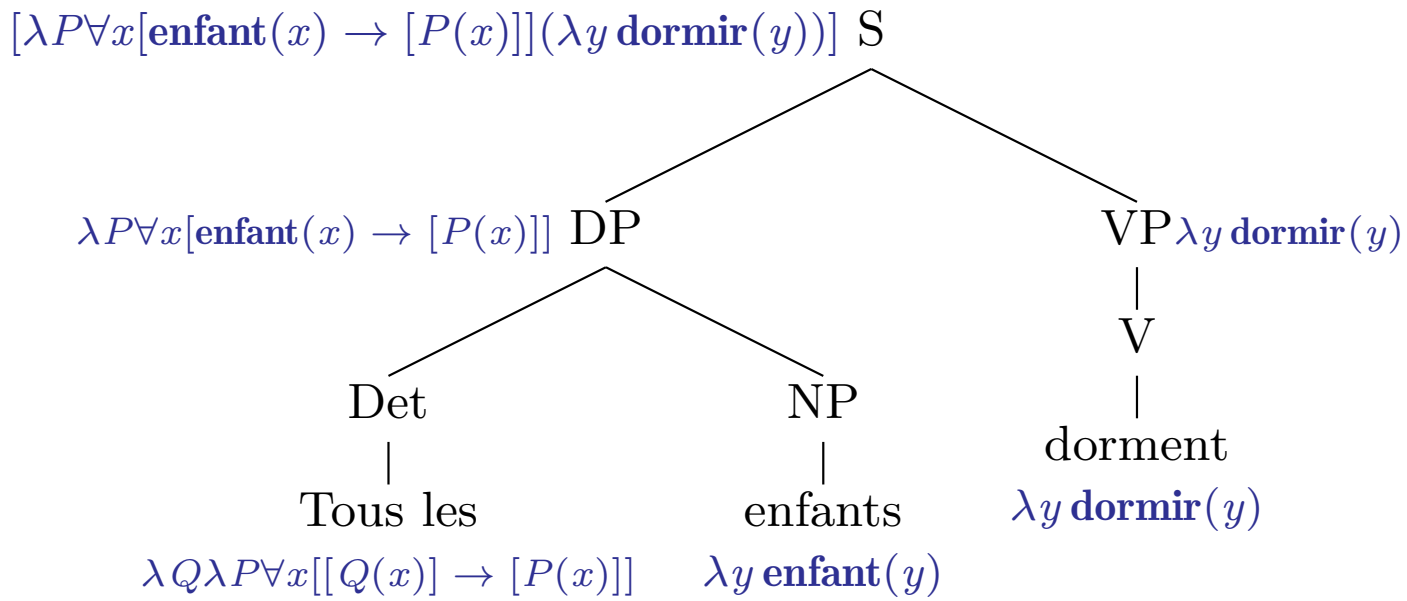
Exemple


S
$\lambda P \forall x[[\lambda y\, \mathbf{enfant}(y)(x)] \rightarrow [P(x)]]$
DP
VP
V
Det
NP
dorment
$\lambda y\, \mathbf{dormir}(y)$
Tous les
enfants
$\lambda Q \lambda P \forall x[[Q(x)] \rightarrow [P(x)]]$
$\lambda y\, \mathbf{enfant}(y)$

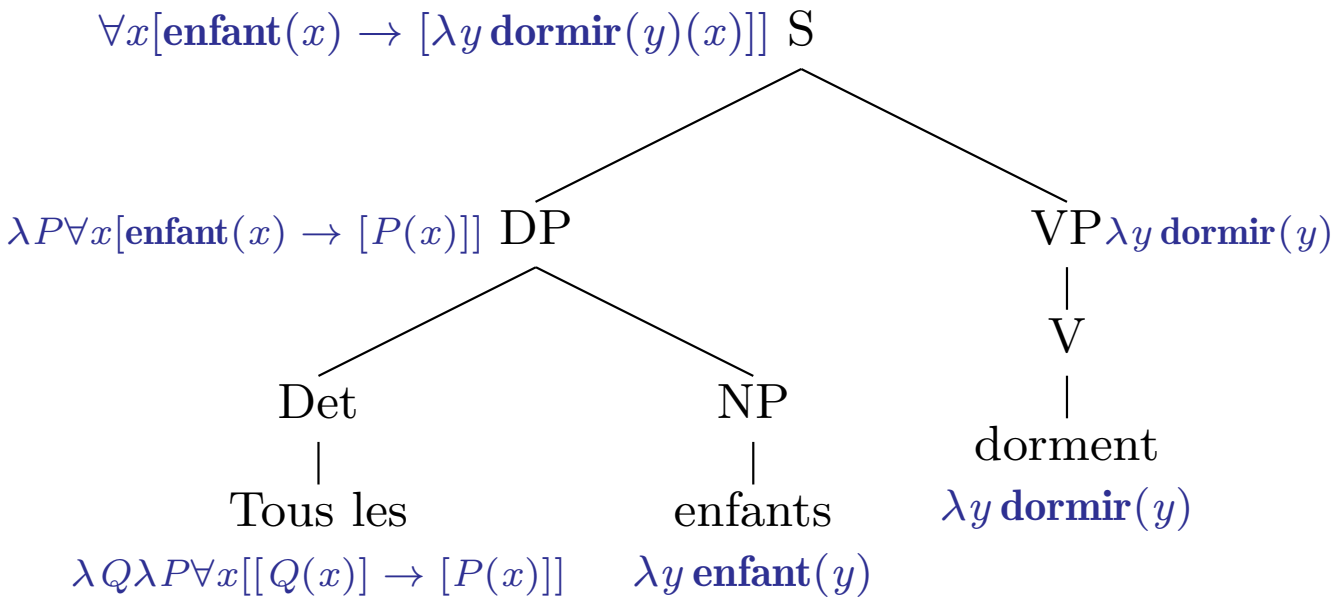
Exemple


S
$\lambda P \forall x[\textbf{enfant}(x) \rightarrow [P(x)]]$ DP
VP
V
Det
NP
dorment
$\lambda y\, \textbf{dormir}(y)$
Tous les
enfants
$\lambda Q \lambda P \forall x[[Q(x)] \rightarrow [P(x)]]$
$\lambda y\, \textbf{enfant}(y)$

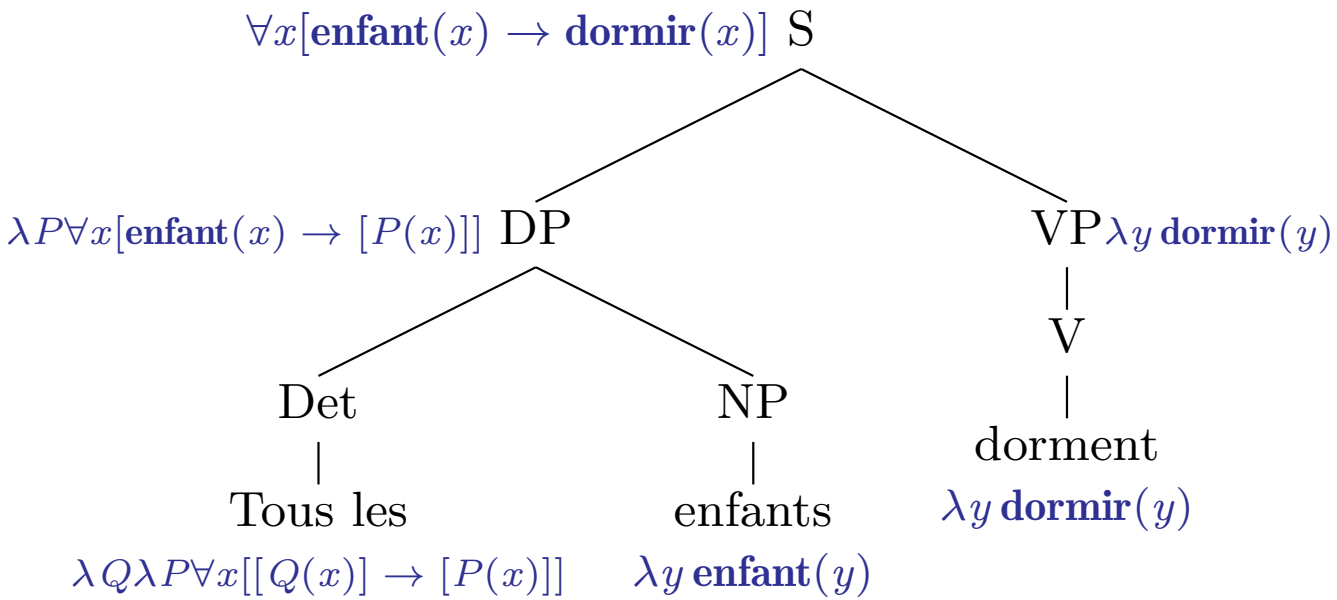
Exemple


$[\lambda P \forall x[\mathbf{enfant}(x) \rightarrow [P(x)]](\lambda y\, \mathbf{dormir}(y))]$ S
$\lambda P \forall x[\mathbf{enfant}(x) \rightarrow [P(x)]]$ DP
VP $\lambda y\, \mathbf{dormir}(y)$
V
Det
NP
dorment
$\lambda y\, \mathbf{dormir}(y)$
Tous les
$\lambda Q \lambda P \forall x[[Q(x)] \rightarrow [P(x)]]$
enfants
$\lambda y\, \mathbf{enfant}(y)$

Exemple


$\forall x[\textbf{enfant}(x) \to [\lambda y\, \textbf{dormir}(y)(x)]]$ S
$\lambda P \forall x[\textbf{enfant}(x) \to [P(x)]]$ DP
VP $\lambda y\, \textbf{dormir}(y)$
V
Det
NP
dorment
$\lambda y\, \textbf{dormir}(y)$
Tous les
enfants
$\lambda Q \lambda P \forall x[[Q(x)] \to [P(x)]]$
$\lambda y\, \textbf{enfant}(y)$

Exemple

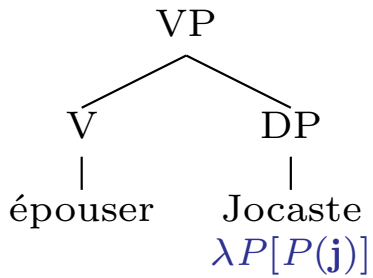

$\forall x[\textbf{enfant}(x) \rightarrow \textbf{dormir}(x)]$ S
$\lambda P \forall x[\textbf{enfant}(x) \rightarrow [P(x)]]$ DP
VP $\lambda y\, \textbf{dormir}(y)$
V
Det
NP
dorment
$\lambda y\, \textbf{dormir}(y)$
Tous les
enfants
$\lambda Q \lambda P \forall x[[Q(x)] \rightarrow [P(x)]]$
$\lambda y\, \textbf{enfant}(y)$

Verbes transitifs

- Les verbes intransitifs sont de type $\langle e, t \rangle$.
- Les verbes transitifs sont-ils de type $\langle e, \langle e, t \rangle \rangle$?

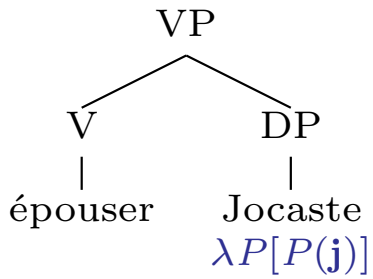
Verbes transitifs

- Les verbes intransitifs sont de type $\langle e, t\rangle$.
- Les verbes transitifs sont-ils de type $\langle e, \langle e, t\rangle\rangle$?
  Non, car leur DP complément (objet) est de type $\langle\langle e, t\rangle, t\rangle$.



Verbes transitifs

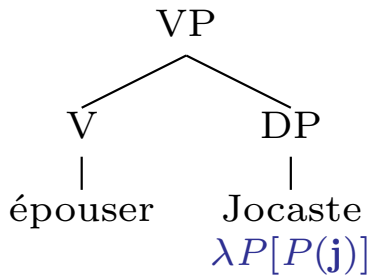
- Les verbes intransitifs sont de type $\langle e, t\rangle$.
- Les verbes transitifs sont-ils de type $\langle e, \langle e, t\rangle\rangle$?
  Non, car leur DP complément (objet) est de type $\langle\langle e, t\rangle, t\rangle$.



- On veut qu'un VT prenne un complément de type $\langle\langle e, t\rangle, t\rangle$ et produise un VP de type $\langle e, t\rangle$.
  Il est donc de type $\langle \underbrace{\langle\langle e, t\rangle, t\rangle}_{\text{DP}}, \underbrace{\langle e, t\rangle}_{\text{VP}}\rangle$.

Verbes transitifs

- Les verbes intransitifs sont de type $\langle e, t\rangle$.
- Les verbes transitifs sont-ils de type $\langle e, \langle e, t\rangle\rangle$?
  Non, car leur DP complément (objet) est de type $\langle\langle e, t\rangle, t\rangle$.

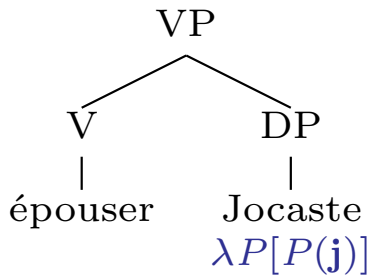


- On veut qu'un VT prenne un complément de type $\langle\langle e, t\rangle, t\rangle$ et produise un VP de type $\langle e, t\rangle$.
  Il est donc de type $\langle \underbrace{\langle\langle e, t\rangle, t\rangle}_{\text{DP}}, \underbrace{\langle e, t\rangle}_{\text{VP}}\rangle$.

épouser $\rightsquigarrow \lambda Y \lambda x[Y(\lambda y\, \mathbf{épouser}(x, y))]$ Y de type $\langle\langle e, t\rangle, t\rangle$

Verbes transitifs

- Les verbes intransitifs sont de type $\langle e, t\rangle$.
- Les verbes transitifs sont-ils de type $\langle e, \langle e, t\rangle\rangle$?
  Non, car leur DP complément (objet) est de type $\langle\langle e, t\rangle, t\rangle$.



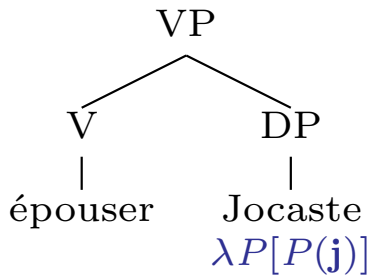
- On veut qu'un VT prenne un complément de type $\langle\langle e, t\rangle, t\rangle$ et produise un VP de type $\langle e, t\rangle$.
  Il est donc de type $\langle \underbrace{\langle\langle e, t\rangle, t\rangle}_{\text{DP}}, \underbrace{\langle e, t\rangle}_{\text{VP}}\rangle$.

$\textit{épouser} \rightsquigarrow \lambda Y \lambda x[Y(\lambda y\, \mathbf{épouser}(x, y))]$ Y de type $\langle\langle e, t\rangle, t\rangle$

$[_{\text{VP}}\, \textit{épouser}\ \textit{Jocaste}] \rightsquigarrow [\lambda Y \lambda x[Y(\lambda y\, \mathbf{épouser}(x, y))](\lambda P[P(\mathbf{j})])]$

Verbes transitifs

- Les verbes intransitifs sont de type $\langle e, t\rangle$.
- Les verbes transitifs sont-ils de type $\langle e, \langle e, t\rangle\rangle$?
  Non, car leur DP complément (objet) est de type $\langle\langle e, t\rangle, t\rangle$.



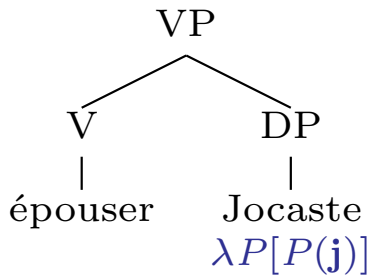
- On veut qu'un VT prenne un complément de type $\langle\langle e, t\rangle, t\rangle$ et produise un VP de type $\langle e, t\rangle$.
  Il est donc de type $\langle \underbrace{\langle\langle e, t\rangle, t\rangle}_{\text{DP}}, \underbrace{\langle e, t\rangle}_{\text{VP}}\rangle$.

$$\textit{épouser} \rightsquigarrow \lambda Y \lambda x[Y(\lambda y\, \mathbf{épouser}(x, y))] \qquad Y \text{ de type } \langle\langle e, t\rangle, t\rangle$$

$$[_{\text{VP}}\, \textit{épouser}\ \textit{Jocaste}] \rightsquigarrow \lambda x[\lambda P[P(\mathbf{j})](\lambda y\, \mathbf{épouser}(x, y))]$$

Verbes transitifs

- Les verbes intransitifs sont de type $\langle e, t\rangle$.
- Les verbes transitifs sont-ils de type $\langle e, \langle e, t\rangle\rangle$?
  Non, car leur DP complément (objet) est de type $\langle\langle e, t\rangle, t\rangle$.



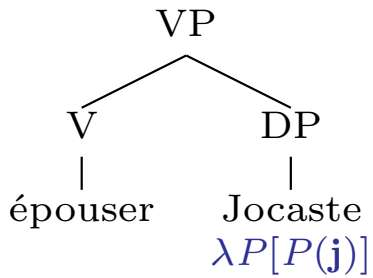
- On veut qu'un VT prenne un complément de type $\langle\langle e, t\rangle, t\rangle$ et produise un VP de type $\langle e, t\rangle$.
  Il est donc de type $\langle \underbrace{\langle\langle e, t\rangle, t\rangle}_{\text{DP}}, \underbrace{\langle e, t\rangle}_{\text{VP}}\rangle$.

$\textit{épouser} \rightsquigarrow \lambda Y \lambda x[Y(\lambda y\, \mathbf{épouser}(x, y))]$ Y de type $\langle\langle e, t\rangle, t\rangle$

$[_{\text{VP}}\, \textit{épouser}\ \textit{Jocaste}] \rightsquigarrow \lambda x[\lambda y\, \mathbf{épouser}(x, y)(\mathbf{j})]$

Verbes transitifs

- Les verbes intransitifs sont de type $\langle e, t\rangle$.
- Les verbes transitifs sont-ils de type $\langle e, \langle e, t\rangle\rangle$?
  Non, car leur DP complément (objet) est de type $\langle\langle e, t\rangle, t\rangle$.



- On veut qu'un VT prenne un complément de type $\langle\langle e, t\rangle, t\rangle$ et produise un VP de type $\langle e, t\rangle$.
  Il est donc de type $\langle \underbrace{\langle\langle e, t\rangle, t\rangle}_{\text{DP}}, \underbrace{\langle e, t\rangle}_{\text{VP}}\rangle$.

$\textit{épouser} \rightsquigarrow \lambda Y \lambda x[Y(\lambda y\, \mathbf{épouser}(x, y))]$ Y de type $\langle\langle e, t\rangle, t\rangle$

$[_{\text{VP}}\, \textit{épouser}\ \textit{Jocaste}] \rightsquigarrow \lambda x\, \mathbf{épouser}(x, \mathbf{j})$

Le verbe transitif *être*

Two *BE*s or not two *BE*s ?

Plusieurs emplois du verbe *être* :

- Transitif (+ DP)
  - Identificationnel
    Peter Parker est Spiderman.
  - Prédicatif (+ DP/NP)
    Peter Parker est un étudiant.

  être $\rightsquigarrow \lambda X \lambda x[X(\lambda y[x = y])]$
- Copule (+ AP)

  Peter Parker est amoureux.

  être $\rightsquigarrow \lambda P \lambda x[P(x)]$

Ambiguïtés de portée

Les DP sont mobiles

Hypothèse

Tout DP *peut* s'interpréter « à gauche » de la phrase où il apparaît.

Ambiguïtés de portée

Les DP sont mobiles

Hypothèse

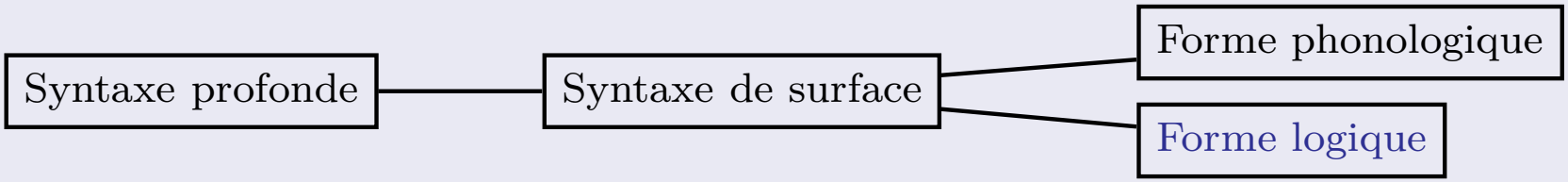
Tout DP *peut* s'interpréter « à gauche » de la phrase où il apparaît.

(3) a. Tous les élèves ont appris un poésie.
b. 'Une poésie tous les élèves ont appris.'
c. 'Tous les élèves une poésie ont appris.'

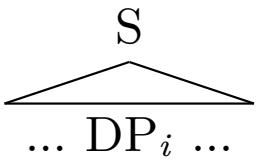
Implémentation syntaxique : *Quantifier Raising*

Les DP sont mobiles

Modèle grammatical « en Y » (e.g. GB)

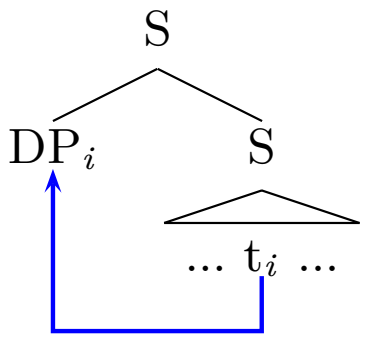


Structure de surface



$\Rightarrow$

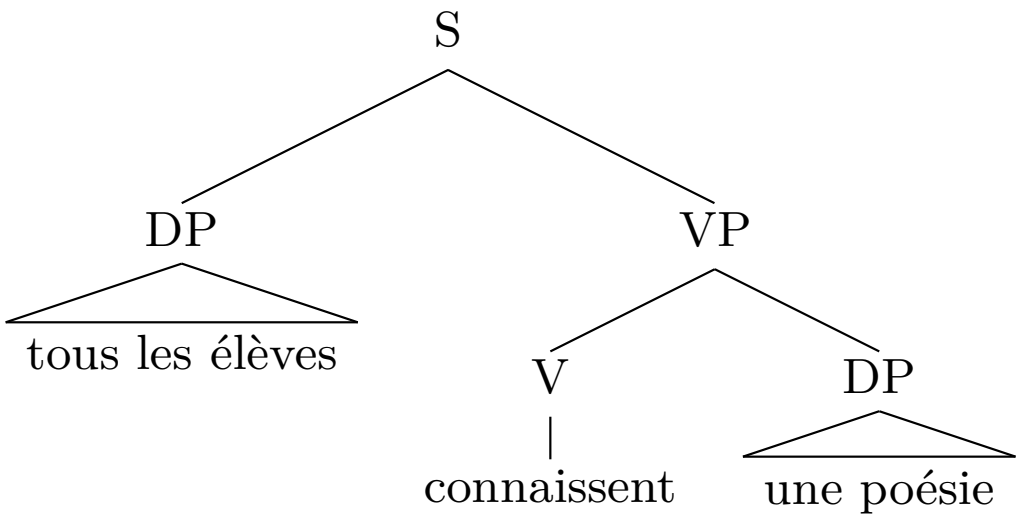
Forme logique



La trace t_i se traduit par une variable (« numérotée ») : $t_i \rightsquigarrow \lambda P[P(x_i)]$.

Quantifier Raising

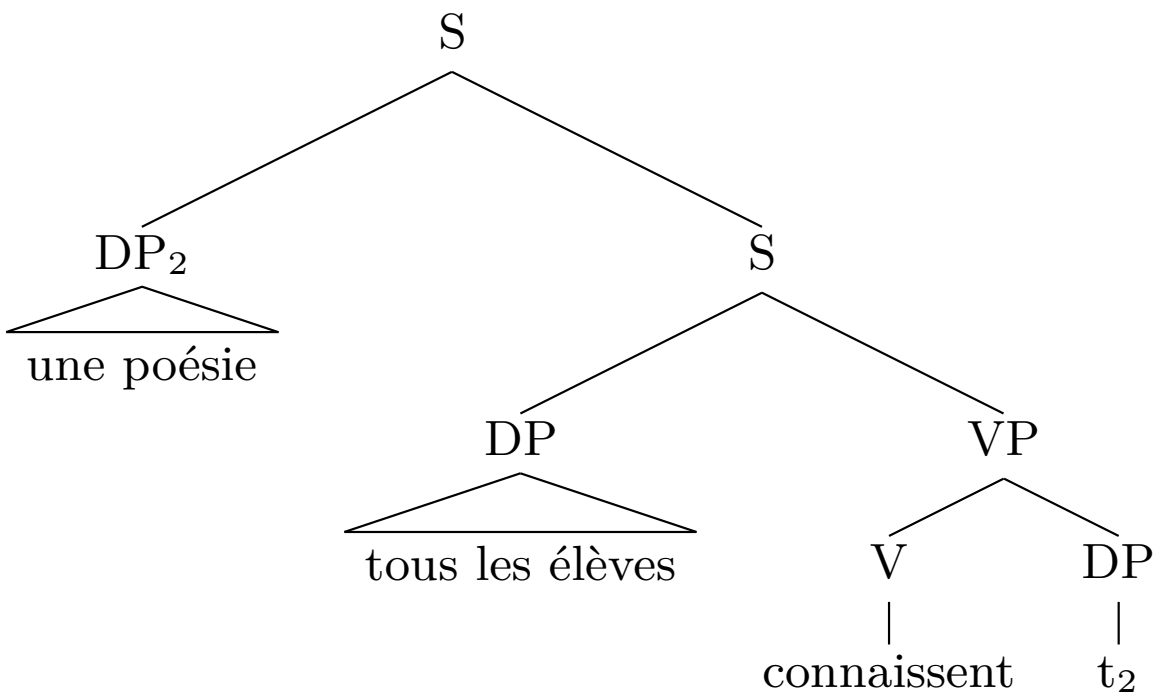
Exemples



Le mouvement est toujours possible, mais jamais obligatoire.

Quantifier Raising

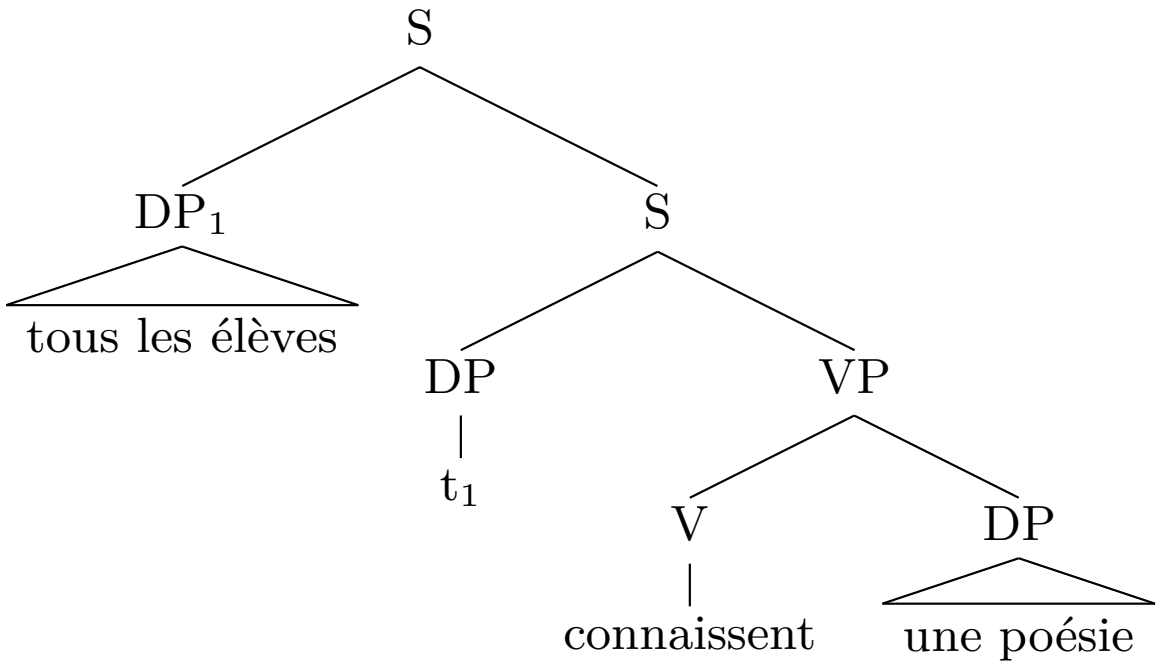
Exemples



Le mouvement est toujours possible, mais jamais obligatoire.

Quantifier Raising

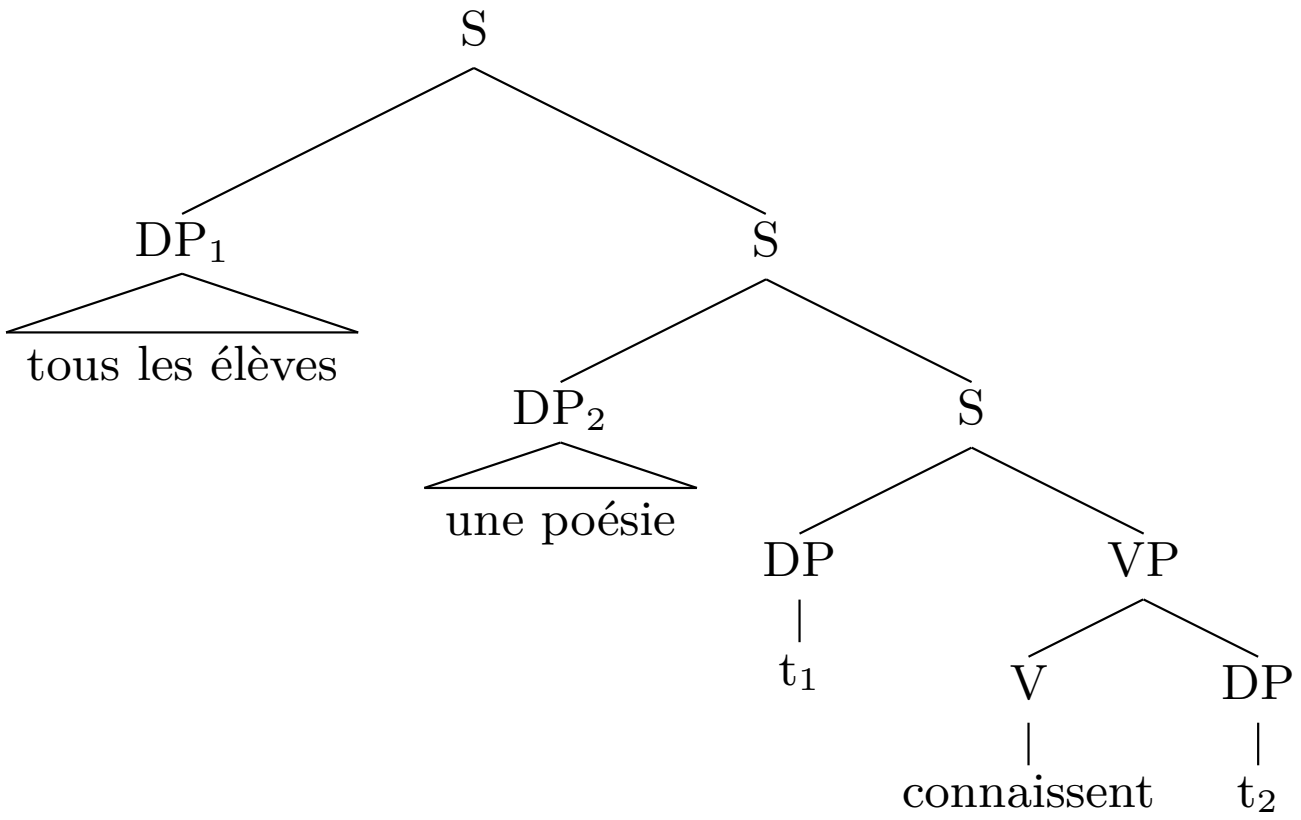
Exemples



Le mouvement est toujours possible, mais jamais obligatoire.

Quantifier Raising

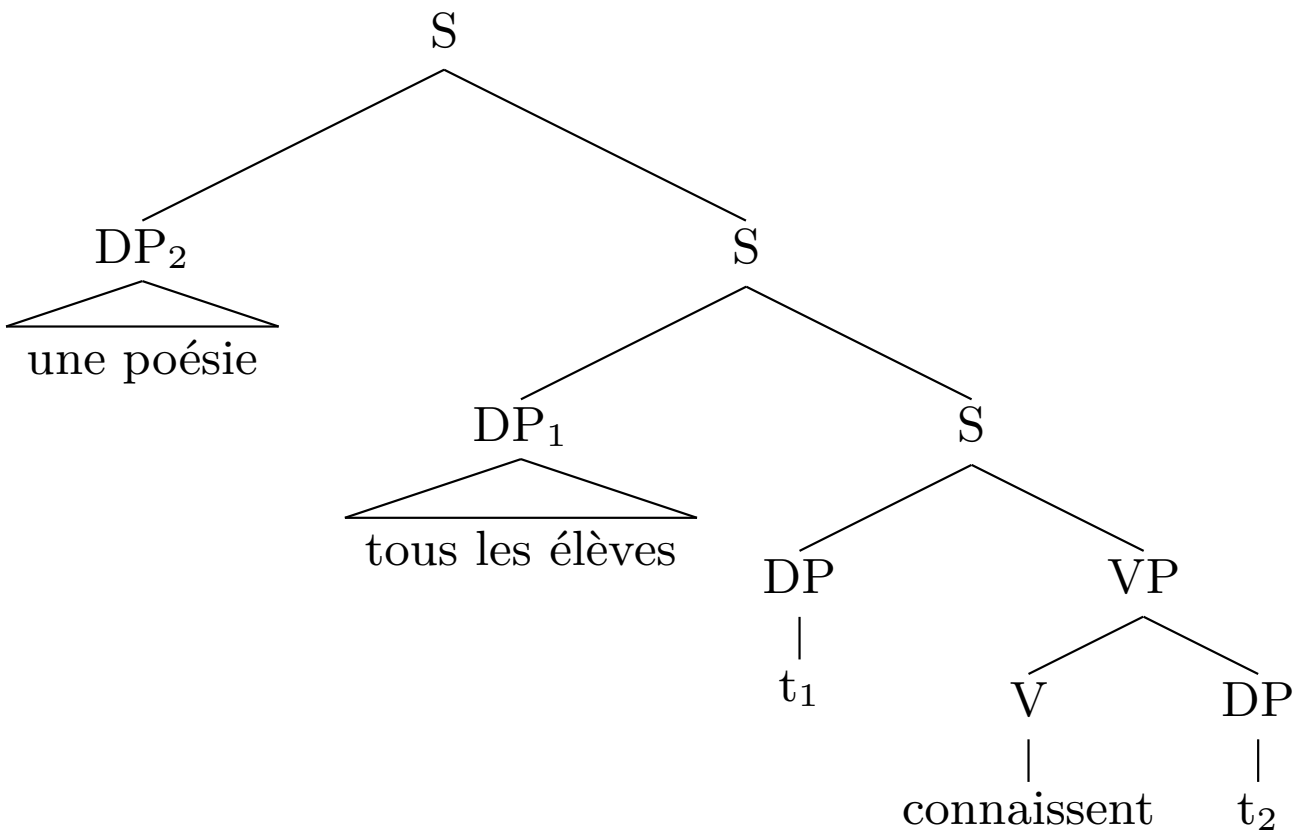
Exemples



Le mouvement est toujours possible, mais jamais obligatoire.

Quantifier Raising

Exemples

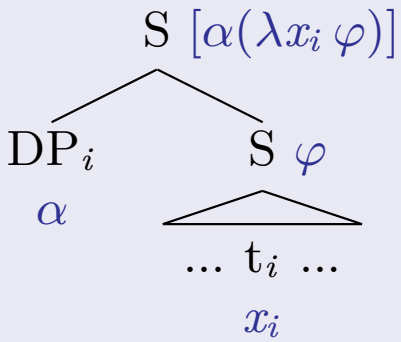


Le mouvement est toujours possible, mais jamais obligatoire.

Interprétation

Les traces sont des variables λ-abstraites (au bon moment)

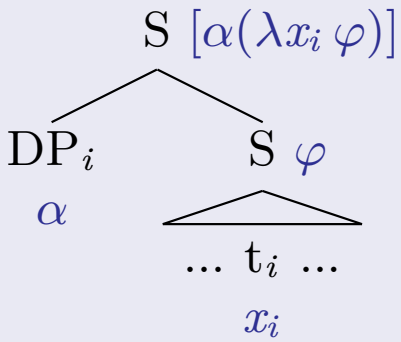
Interprétation des traces (e.g. *QR*)



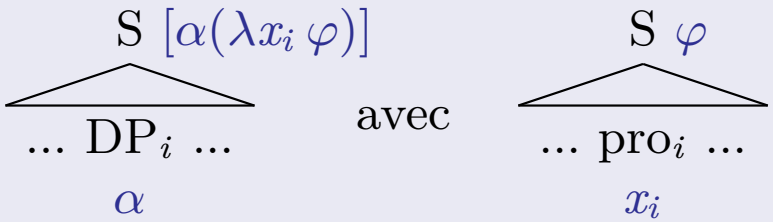
Interprétation

Les traces sont des variables λ-abstraites (au bon moment)

Interprétation des traces (e.g. *QR*)



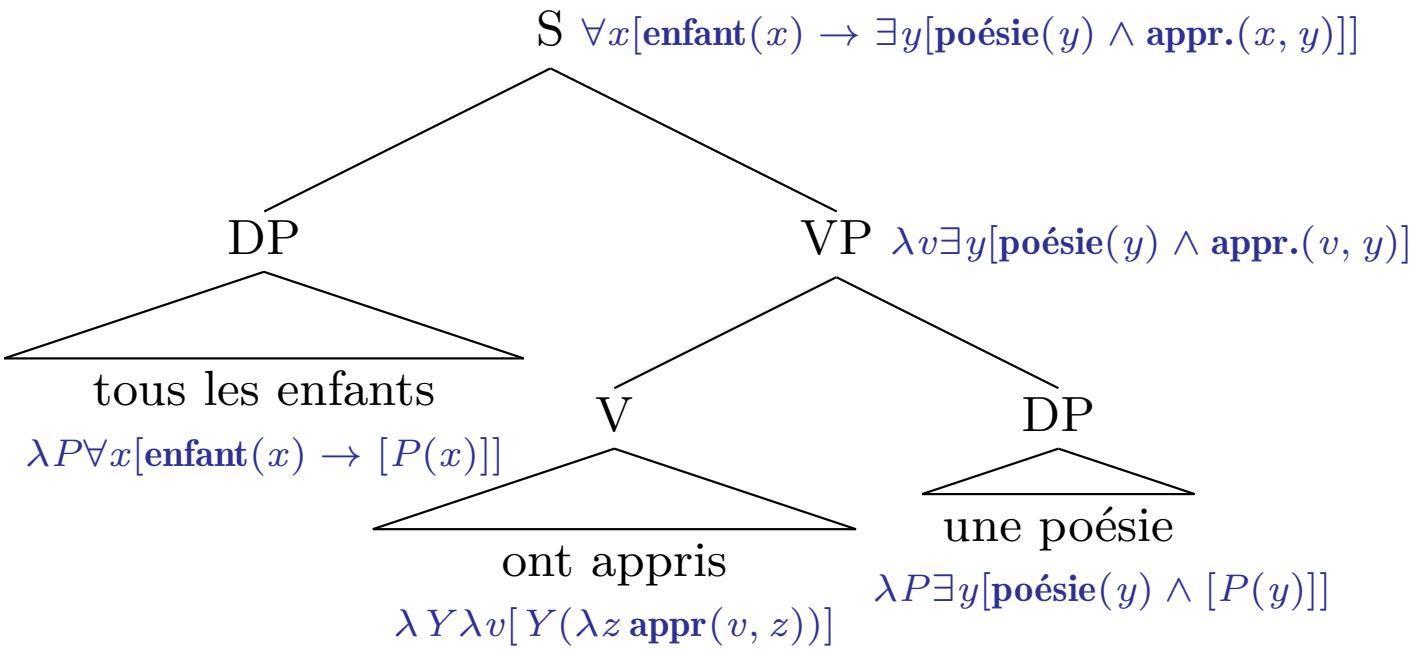
Quantifying in (Montague)



Ambiguïté de portée

Exemple

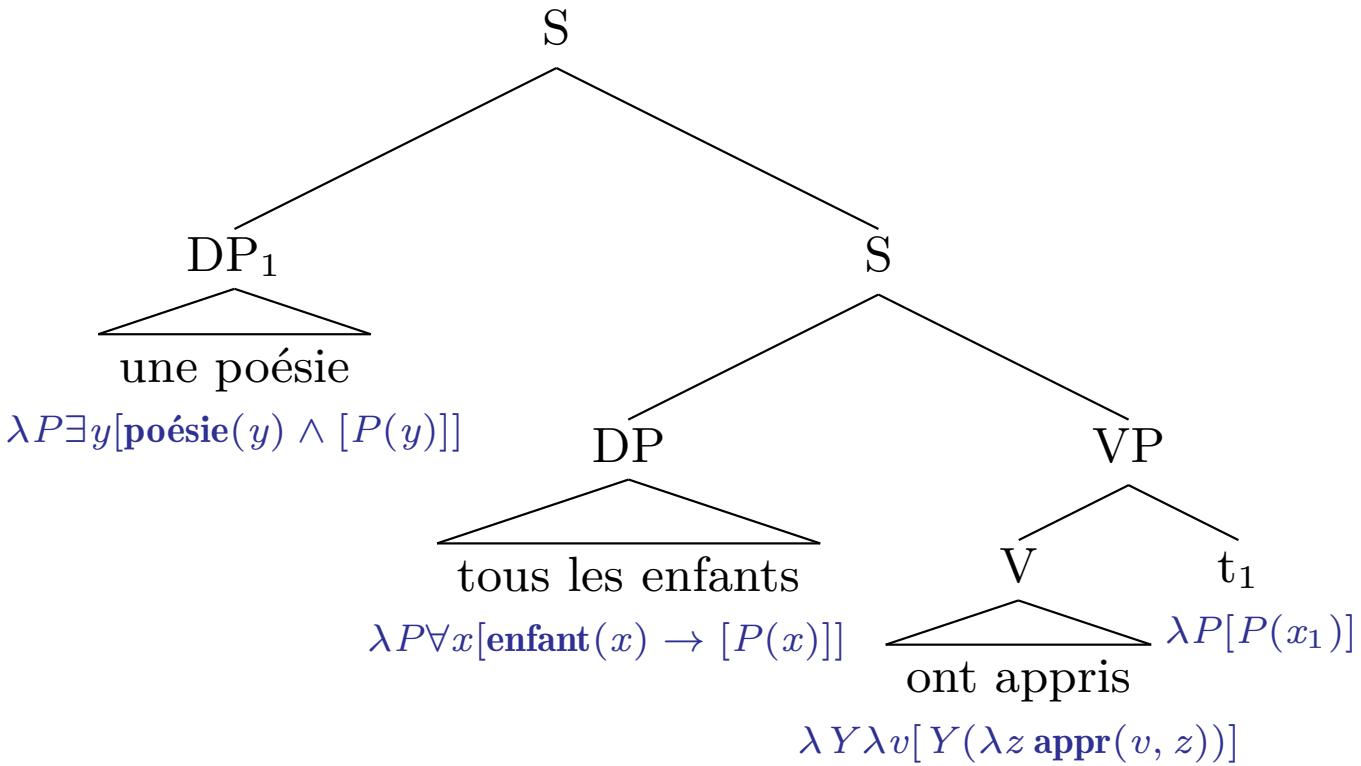
Interprétation des DP *in situ*



Ambiguïté de portée

Exemple

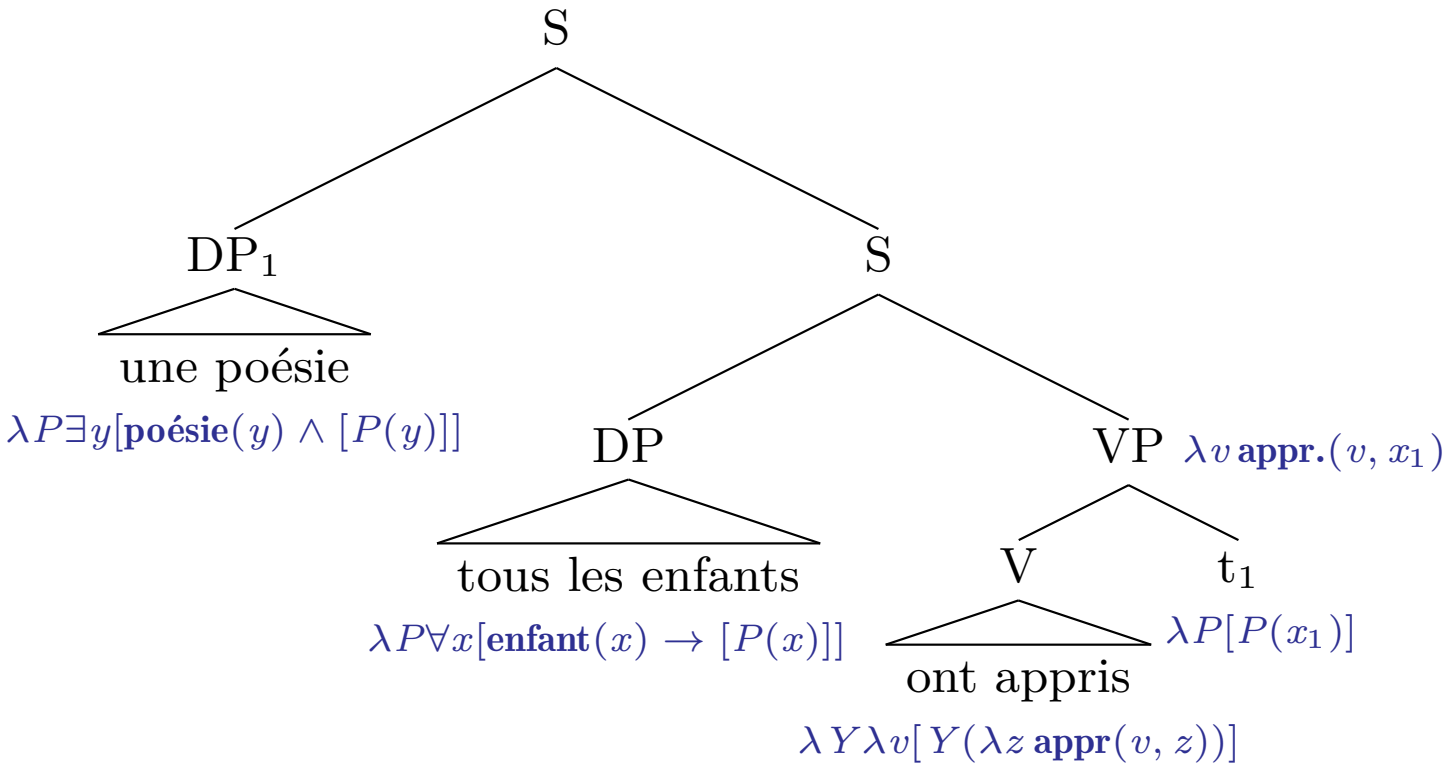
Interprétation avec montée (ou *quantifying in*) de l'objet



Ambiguïté de portée

Exemple

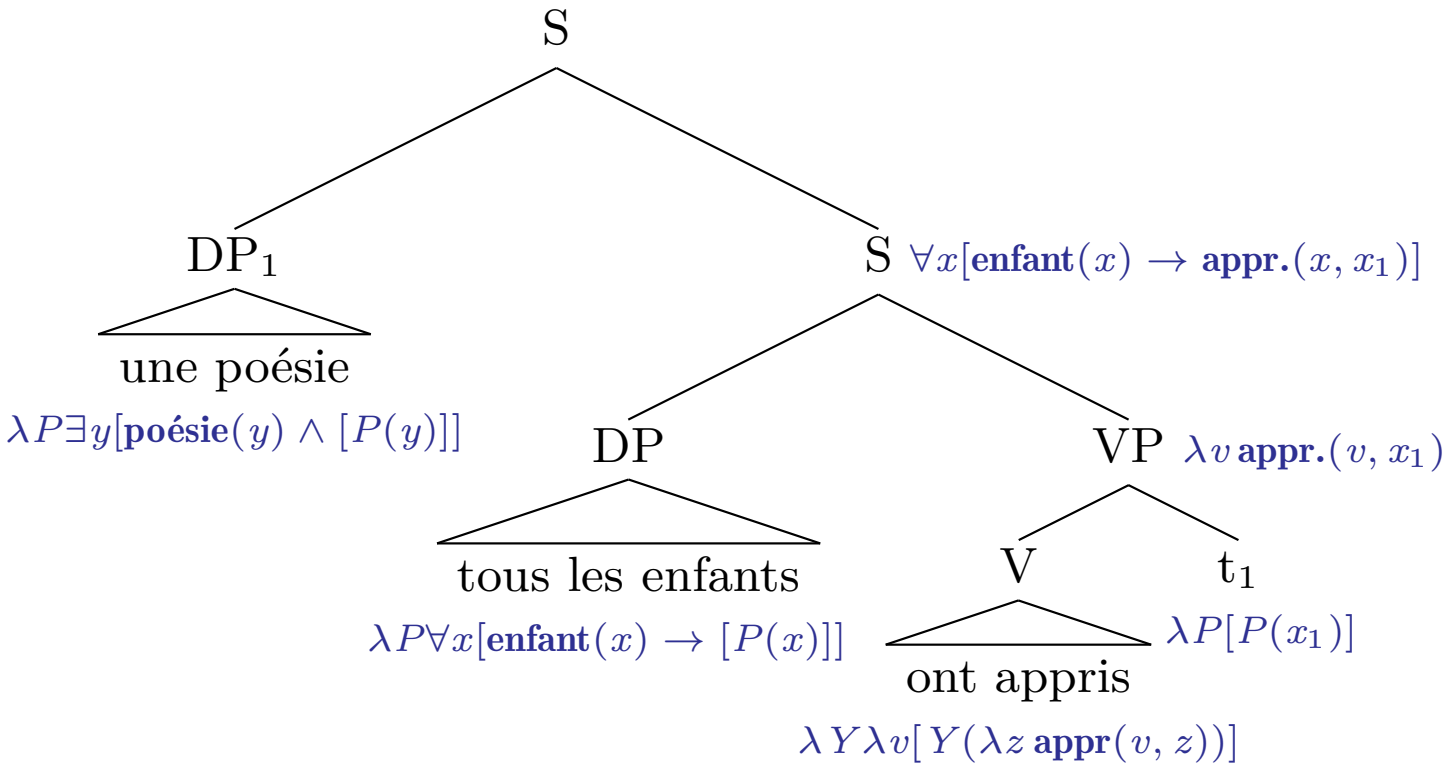
Interprétation avec montée (ou *quantifying in*) de l'objet



Ambiguïté de portée

Exemple

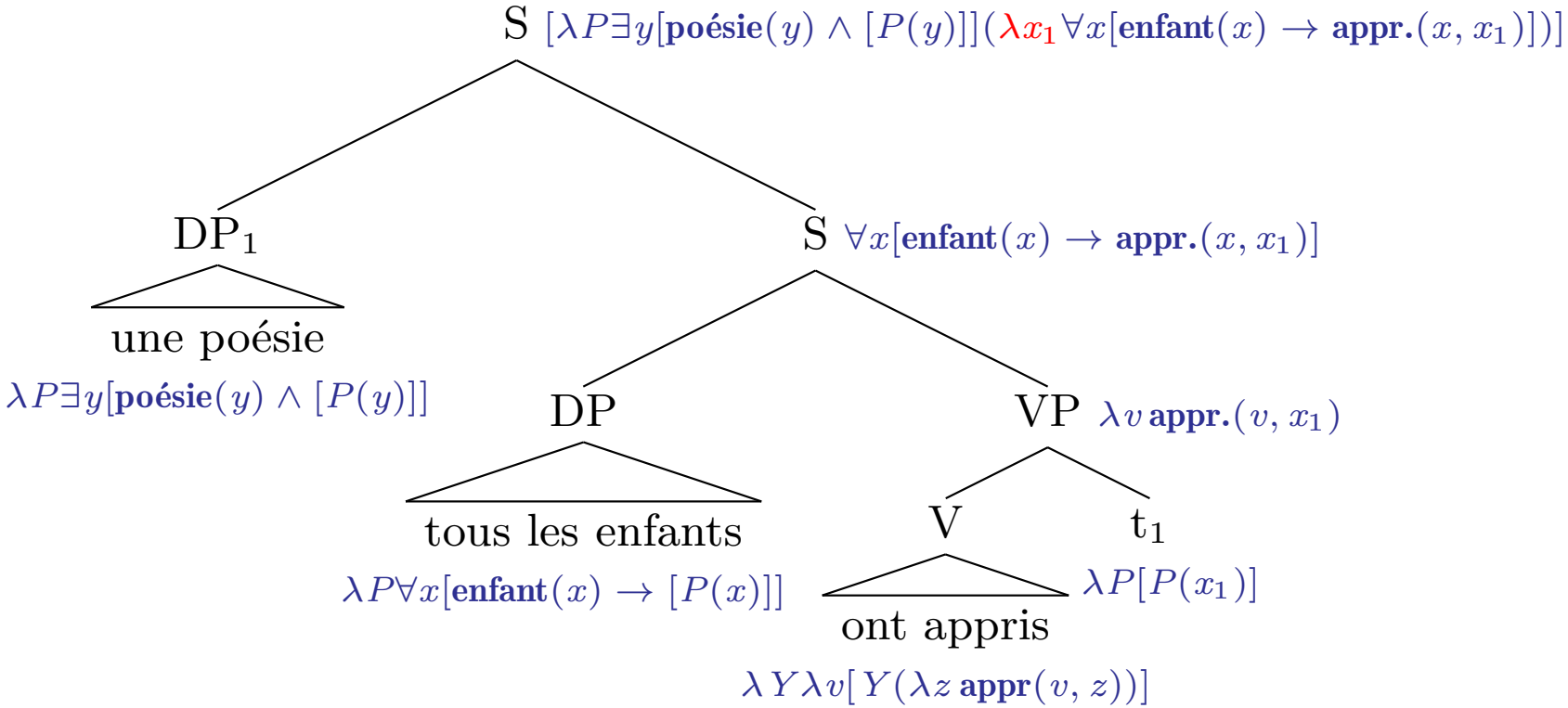
Interprétation avec montée (ou *quantifying in*) de l'objet



Ambiguïté de portée

Exemple

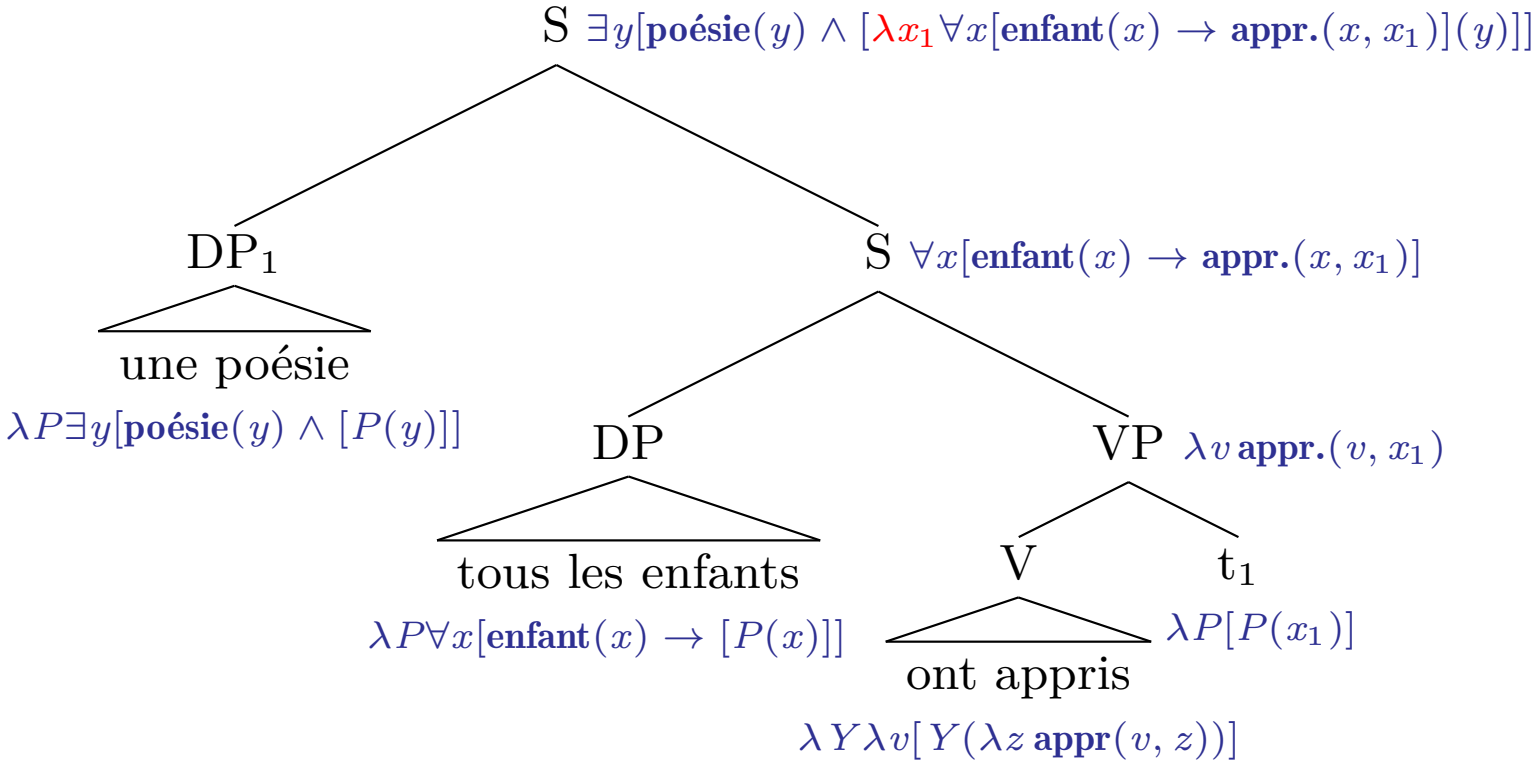
Interprétation avec montée (ou *quantifying in*) de l'objet



Ambiguïté de portée

Exemple

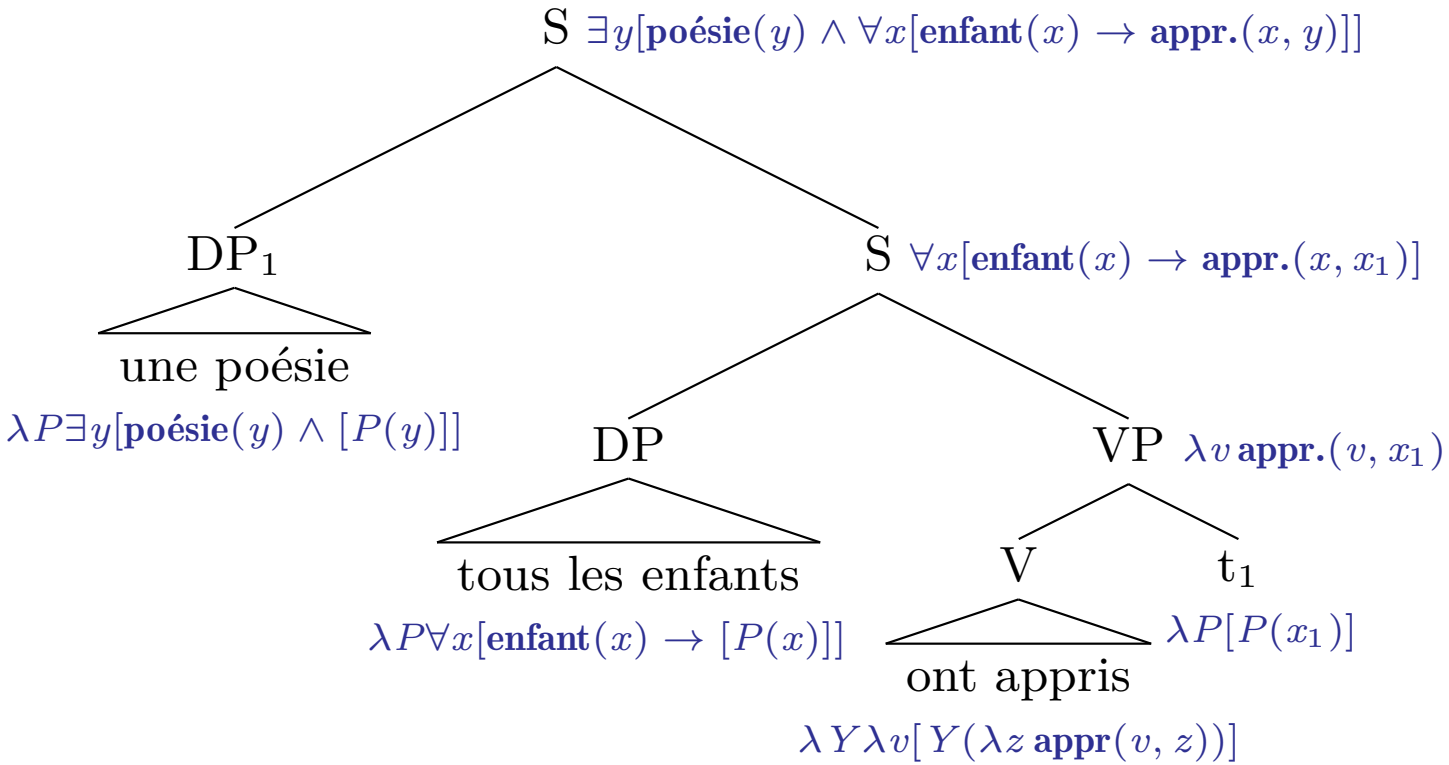
Interprétation avec montée (ou *quantifying in*) de l'objet



Ambiguïté de portée

Exemple

Interprétation avec montée (ou *quantifying in*) de l'objet



De dicto vs. de re

Verbes d'attitudes propositionnelles + subordonnées complétives

Ils dénotent une relation entre une proposition φ ($\langle s, t \rangle$) et un individu x.
Ils sont de type $\langle \langle s, t \rangle, \langle e, t \rangle \rangle$

De dicto vs. de re

Verbes d'attitudes propositionnelles + subordonnées complétives

Ils dénotent une relation entre une proposition φ ($\langle \mathrm{s}, \mathrm{t} \rangle$) et un individu x.
Ils sont de type $\langle \langle \mathrm{s}, \mathrm{t} \rangle, \langle \mathrm{e}, \mathrm{t} \rangle \rangle$

- *croire* $\rightsquigarrow \lambda\varphi\lambda x\, \mathbf{croire}(x, \varphi)$

De dicto vs. de re

Verbes d'attitudes propositionnelles + subordonnées complétives

Ils dénotent une relation entre une proposition φ ($\langle \mathrm{s}, \mathrm{t} \rangle$) et un individu x.
Ils sont de type $\langle \langle \mathrm{s}, \mathrm{t} \rangle, \langle \mathrm{e}, \mathrm{t} \rangle \rangle$

- *croire* $\rightsquigarrow \lambda\varphi\lambda x\, \mathbf{croire}(x, \varphi)$
  $[\![\mathbf{croire}(x, \varphi)]\!]^{\mathcal{M},w,g} = 1$ ssi les croyances qu'a $[\![x]\!]^{\mathcal{M},w,g}$ dans w impliquent φ ;

De dicto vs. de re

Verbes d'attitudes propositionnelles + subordonnées complétives

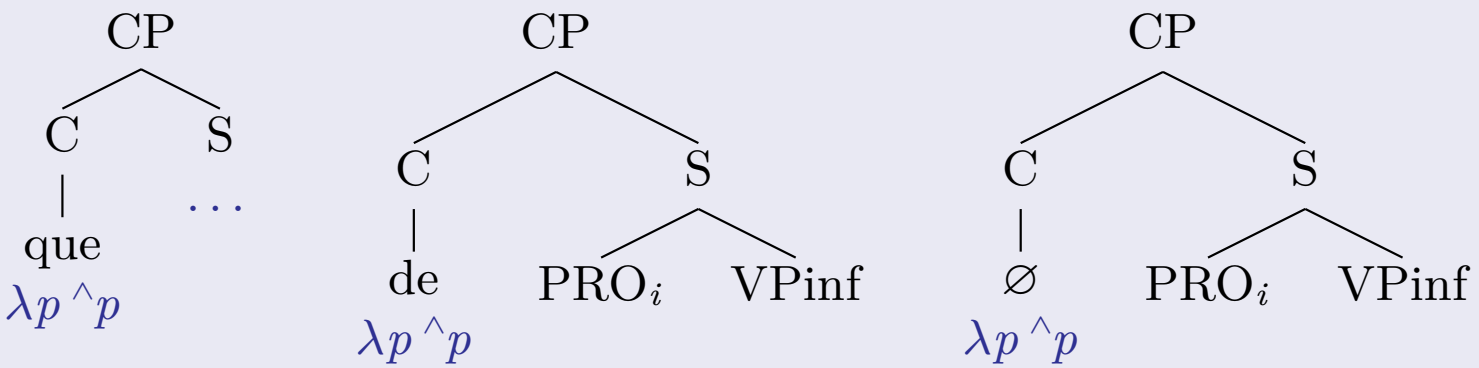
Ils dénotent une relation entre une proposition φ ($\langle s, t\rangle$) et un individu x.
Ils sont de type $\langle\langle s, t\rangle, \langle e, t\rangle\rangle$

- *croire* $\rightsquigarrow \lambda\varphi\lambda x\, \mathbf{croire}(x, \varphi)$
  $[\![\mathbf{croire}(x, \varphi)]\!]^{\mathcal{M},w,g} = 1$ ssi les croyances qu'a $[\![x]\!]^{\mathcal{M},w,g}$ dans w impliquent φ ;
- *vouloir* $\rightsquigarrow \lambda\varphi\lambda x\, \mathbf{vouloir}(x, \varphi)$
  $[\![\mathbf{vouloir}(x, \varphi)]\!]^{\mathcal{M},w,g} = 1$ ssi l'ensemble de tous les mondes w' compatibles avec les désirs de $[\![x]\!]^{\mathcal{M},w,g}$ dans w est inclus dans $[\![\varphi]\!]^{\mathcal{M},w,g}$;
  autrement dit : dans w, $[\![x]\!]^{\mathcal{M},w,g}$ souhaite que φ devienne vraie (dans (le futur de) w).

De dicto vs. de re

Complémenteurs et complétives

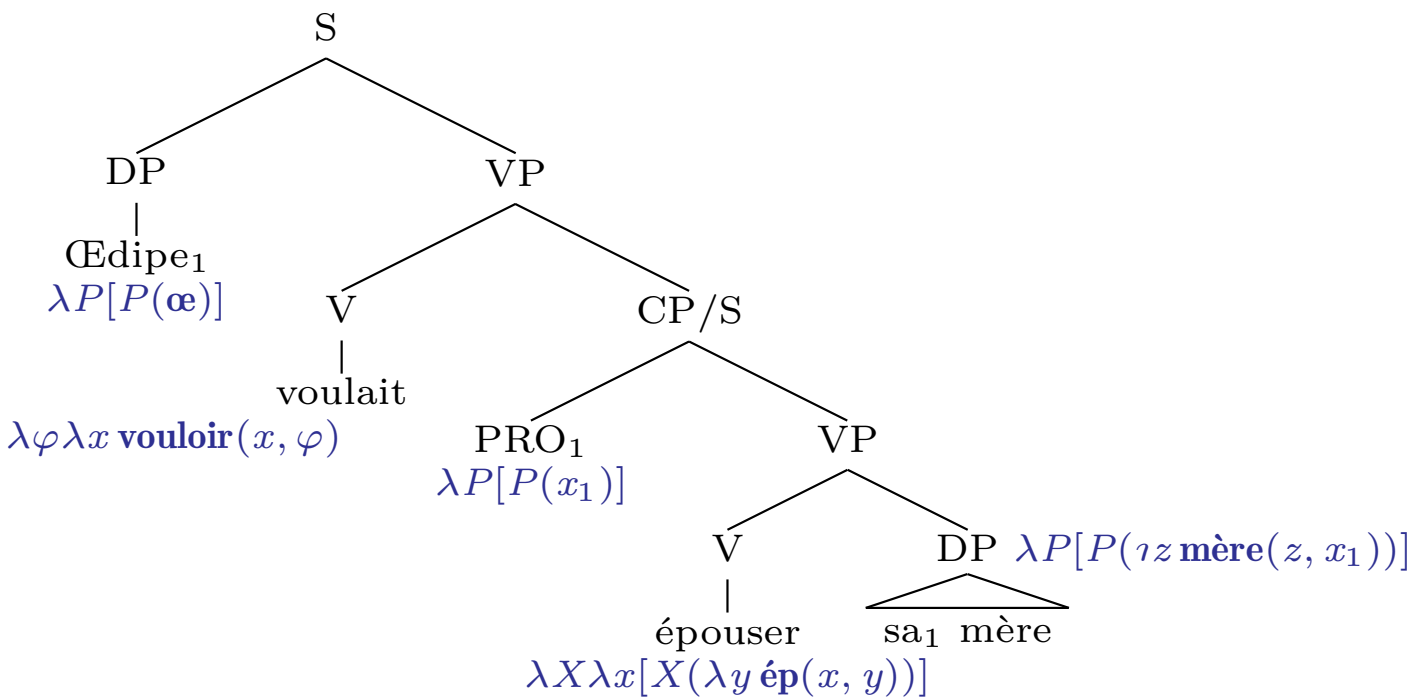
Les complémenteurs (*que*, *de*) introduisent $\hat{}$, pour former une proposition à partir d'une formule.
Ils sont donc de type $\langle t, \langle s, t\rangle\rangle$:



De dicto vs. de re

Œdipe voulait épouser sa mère (de dicto)

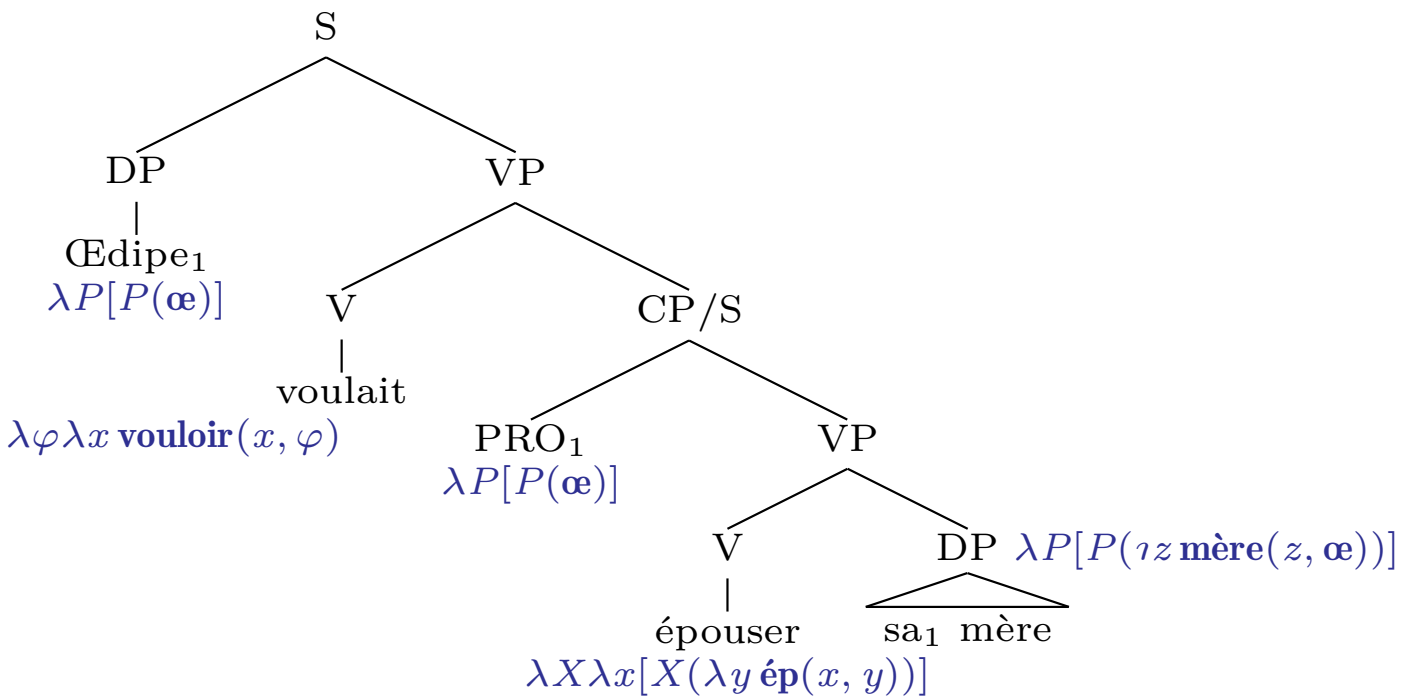
(4)



De dicto vs. de re

Œdipe voulait épouser sa mère (de dicto)

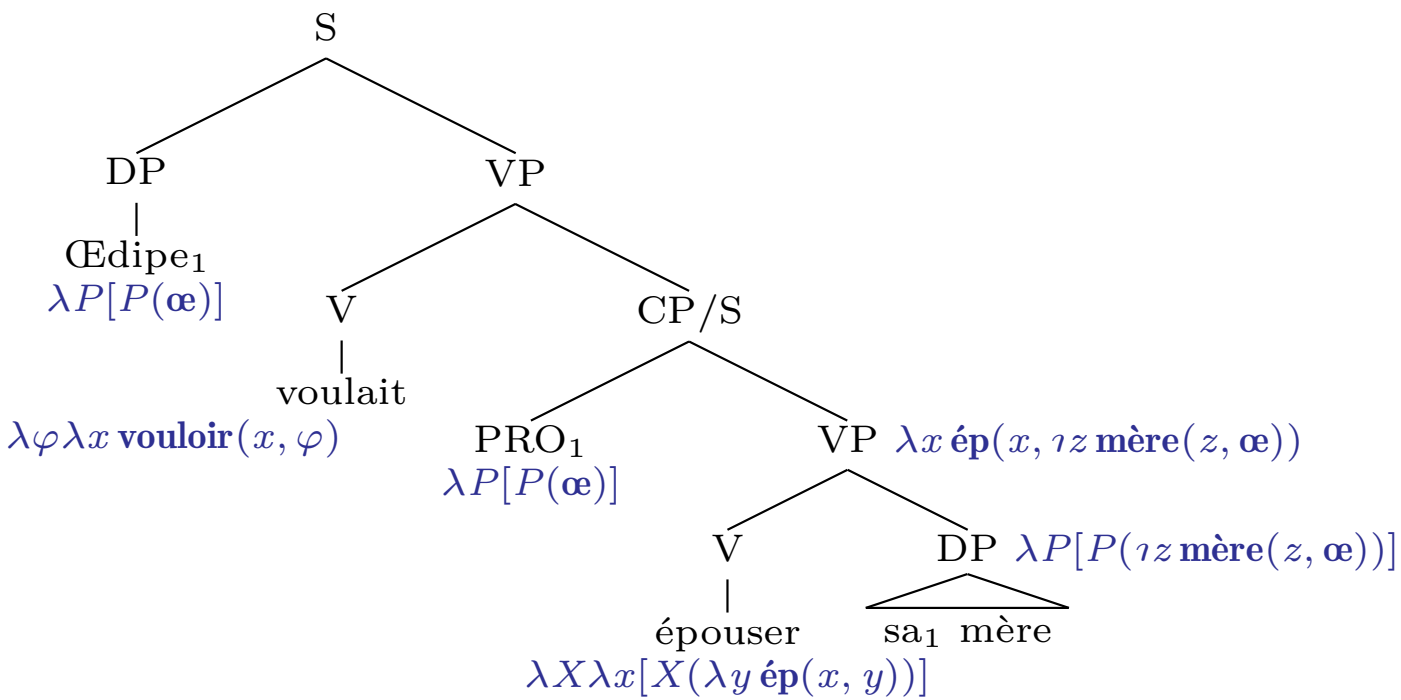
(4)



De dicto vs. de re

Œdipe voulait épouser sa mère (de dicto)

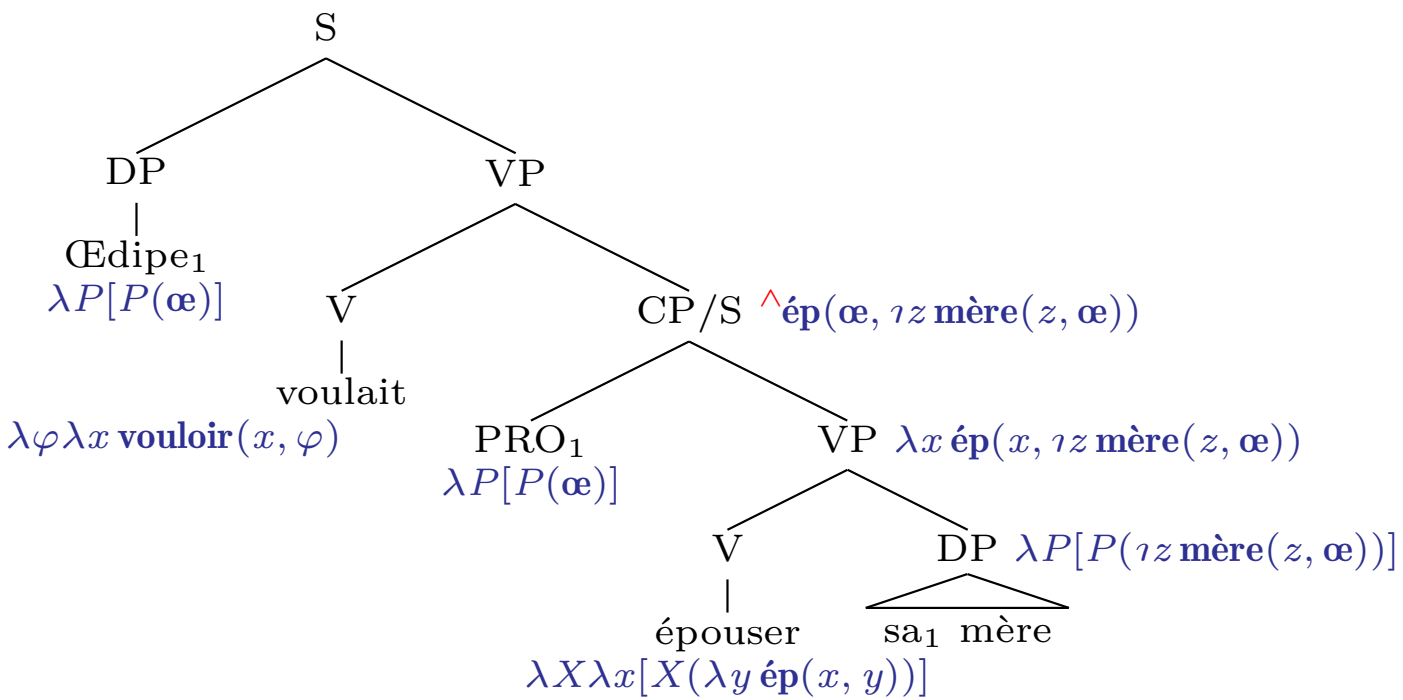
(4)



De dicto vs. de re

Œdipe voulait épouser sa mère (de dicto)

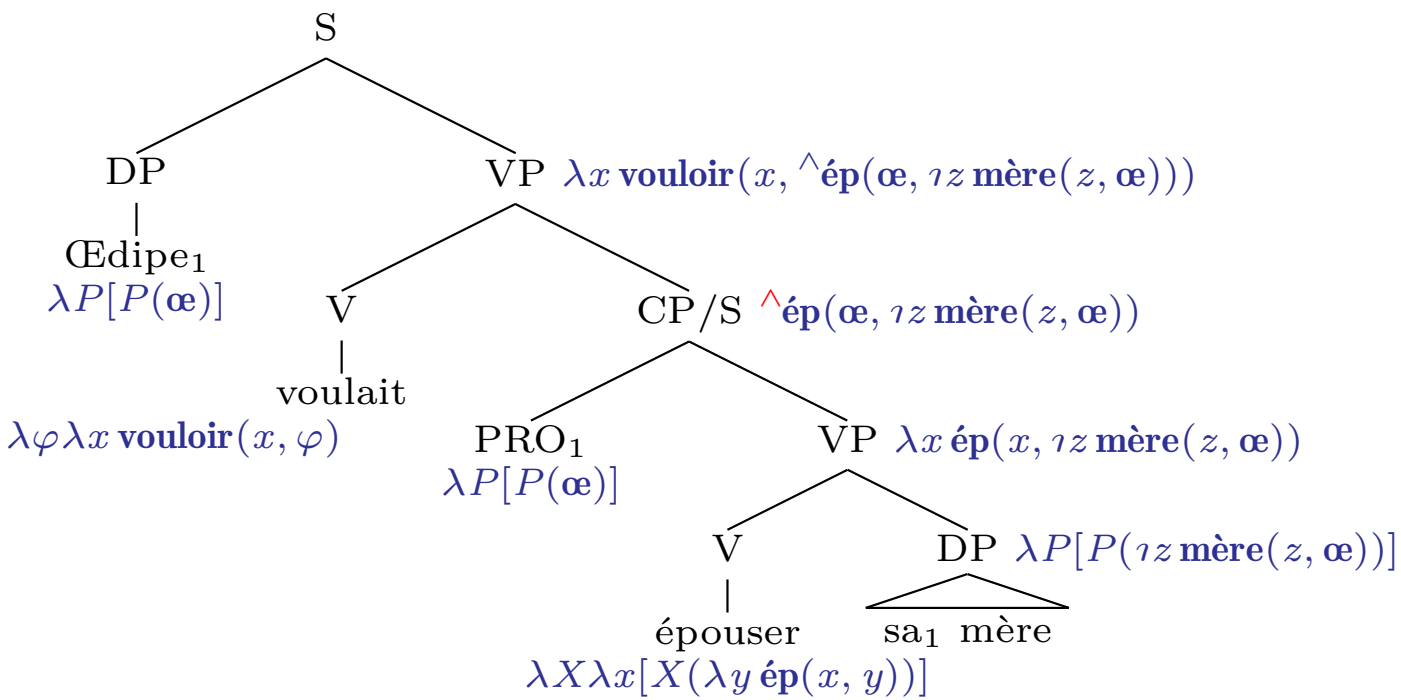
(4)



De dicto vs. de re

Œdipe voulait épouser sa mère (de dicto)

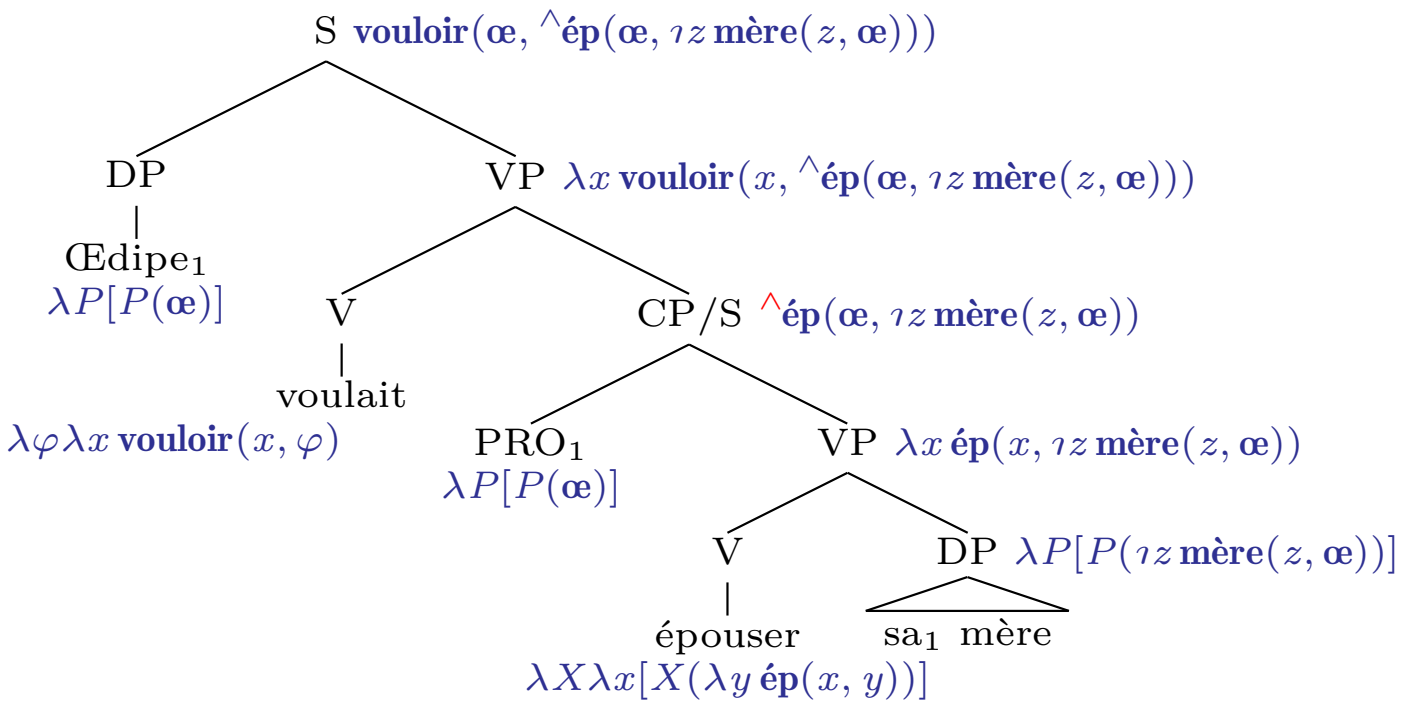
(4)



De dicto vs. de re

Œdipe voulait épouser sa mère (de dicto)

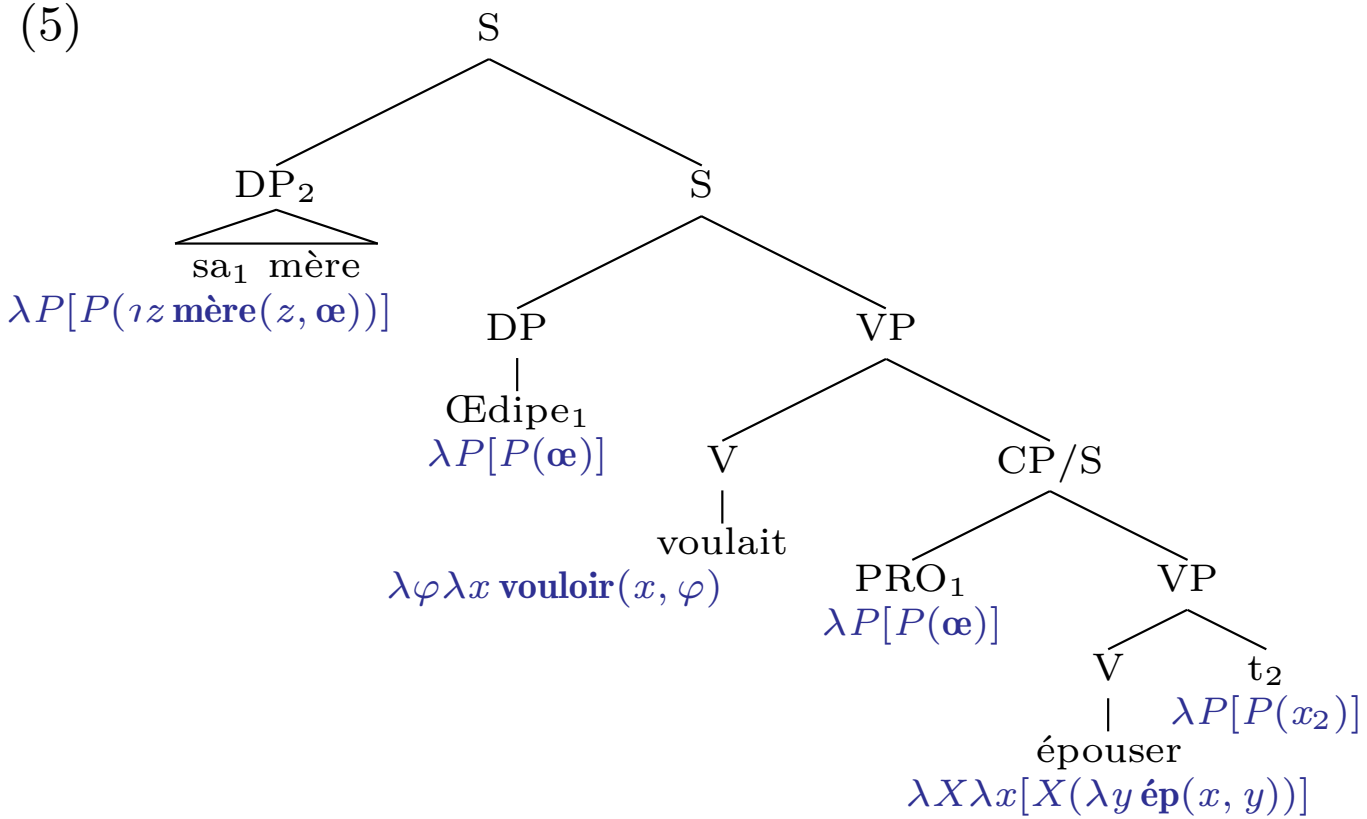
(4)



De dicto vs. de re

Œdipe voulait épouser sa mère (de re)

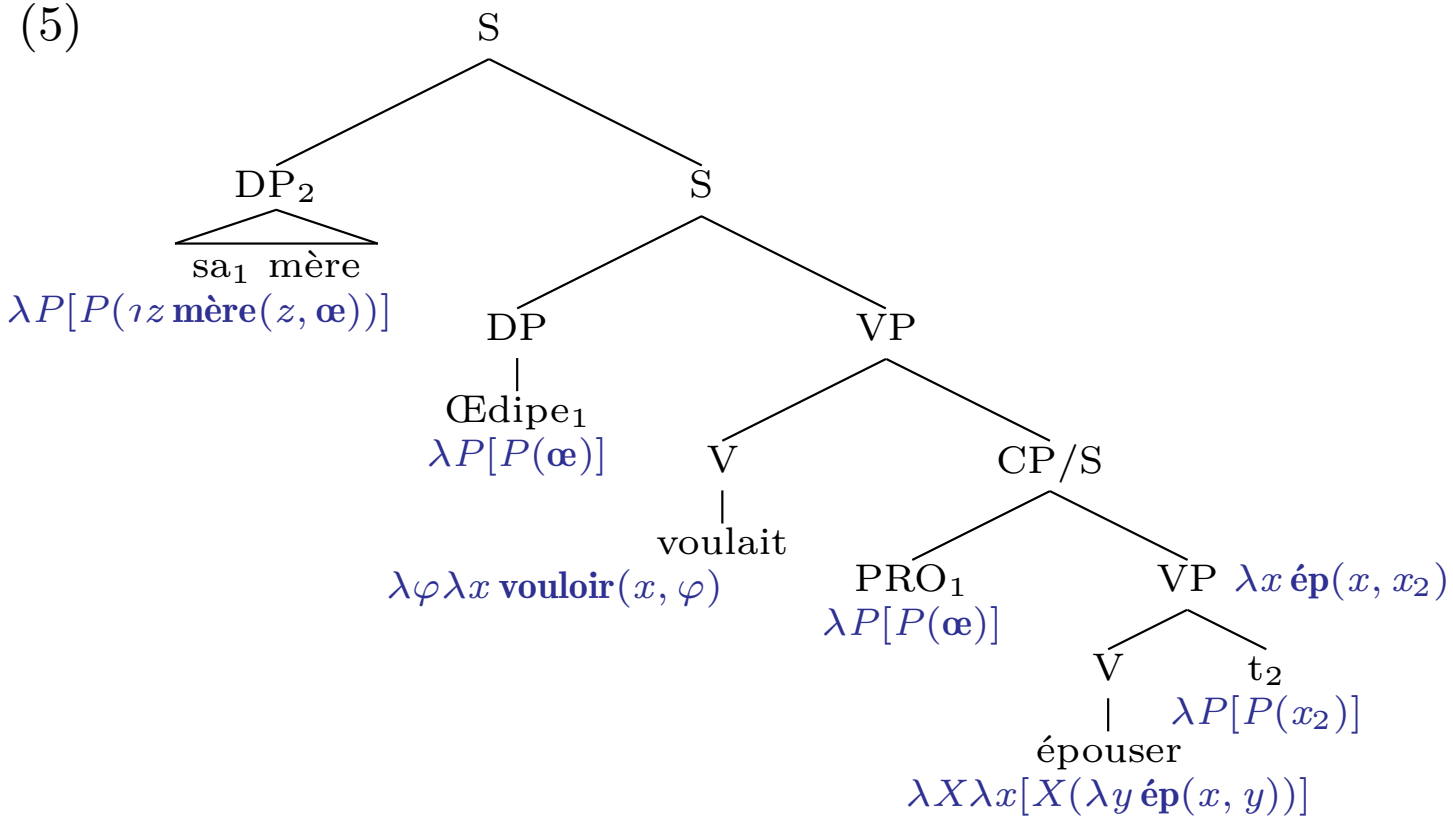
(5)



De dicto vs. de re

Œdipe voulait épouser sa mère (de re)

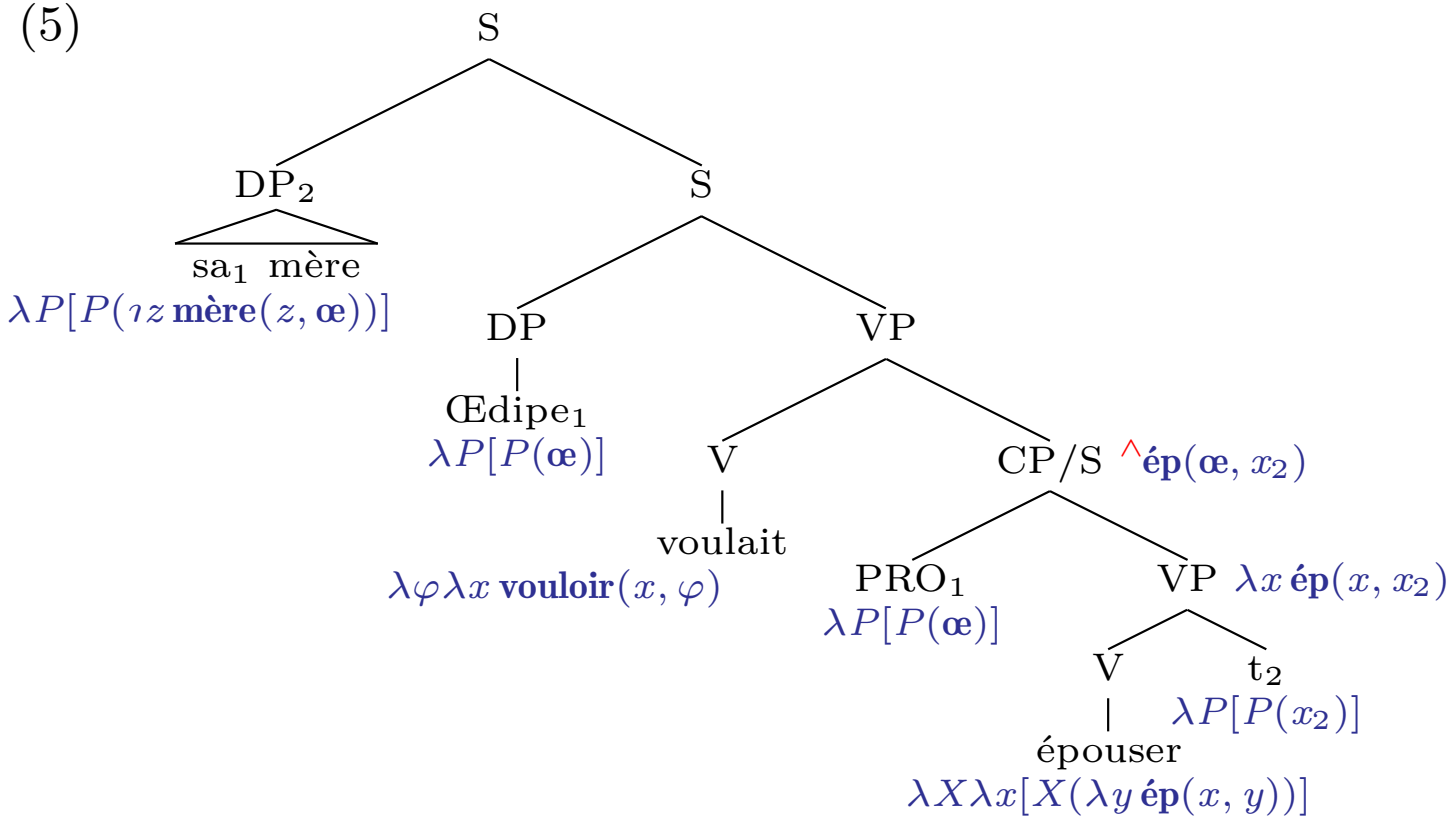
(5)



De dicto vs. de re

Œdipe voulait épouser sa mère (de re)

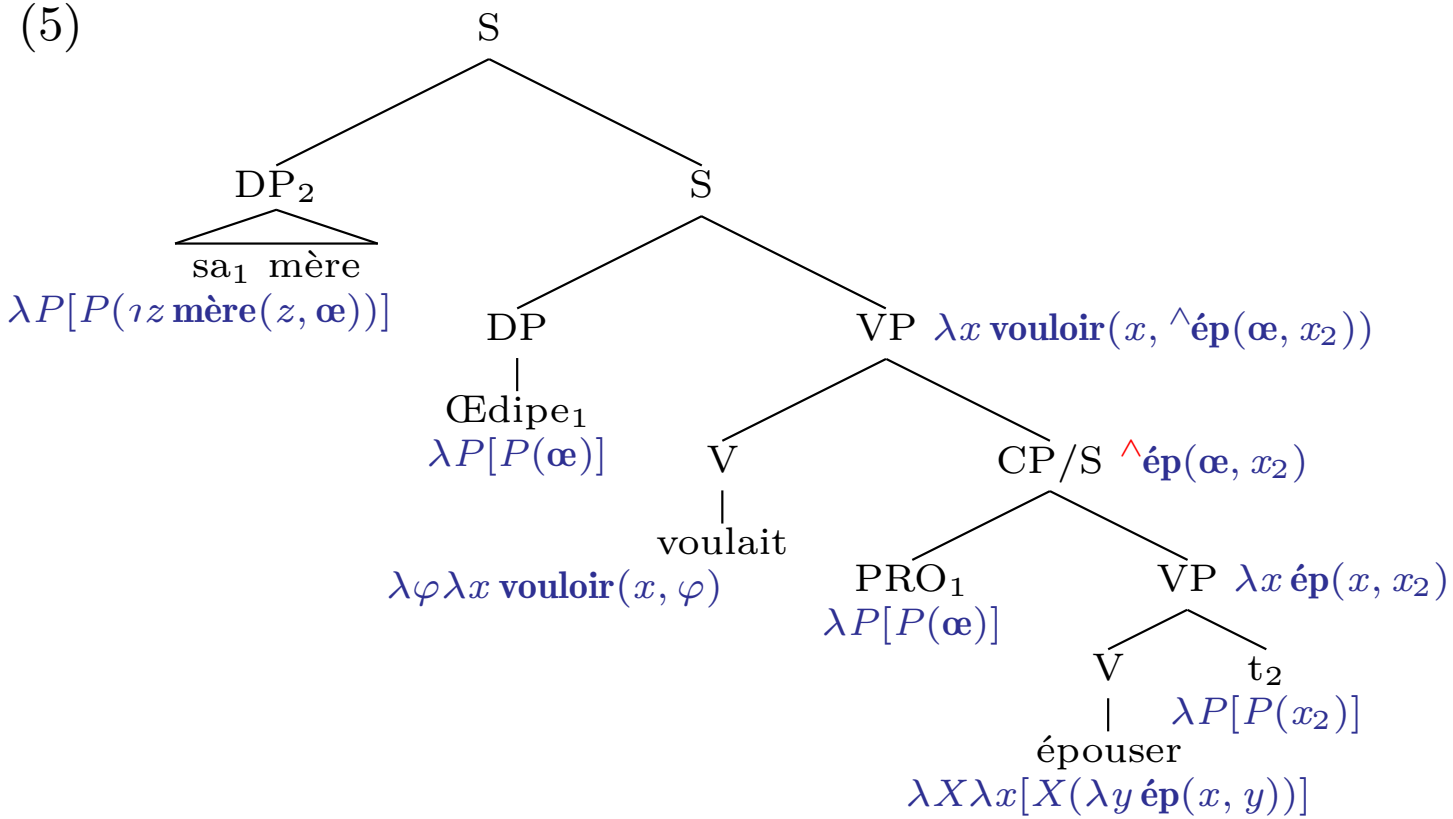
(5)



De dicto vs. de re

Œdipe voulait épouser sa mère (de re)

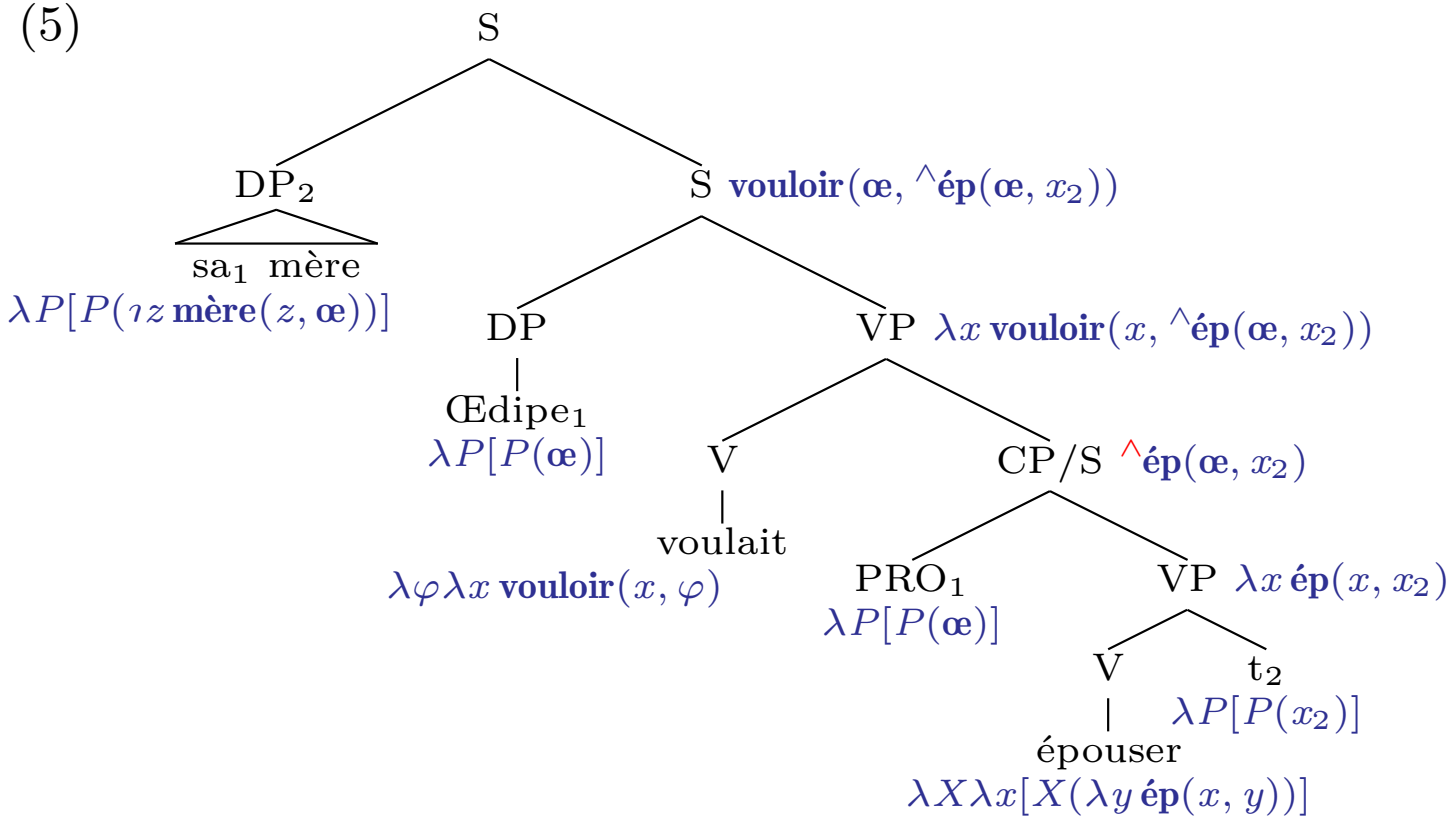
(5)



De dicto vs. de re

Œdipe voulait épouser sa mère (de re)

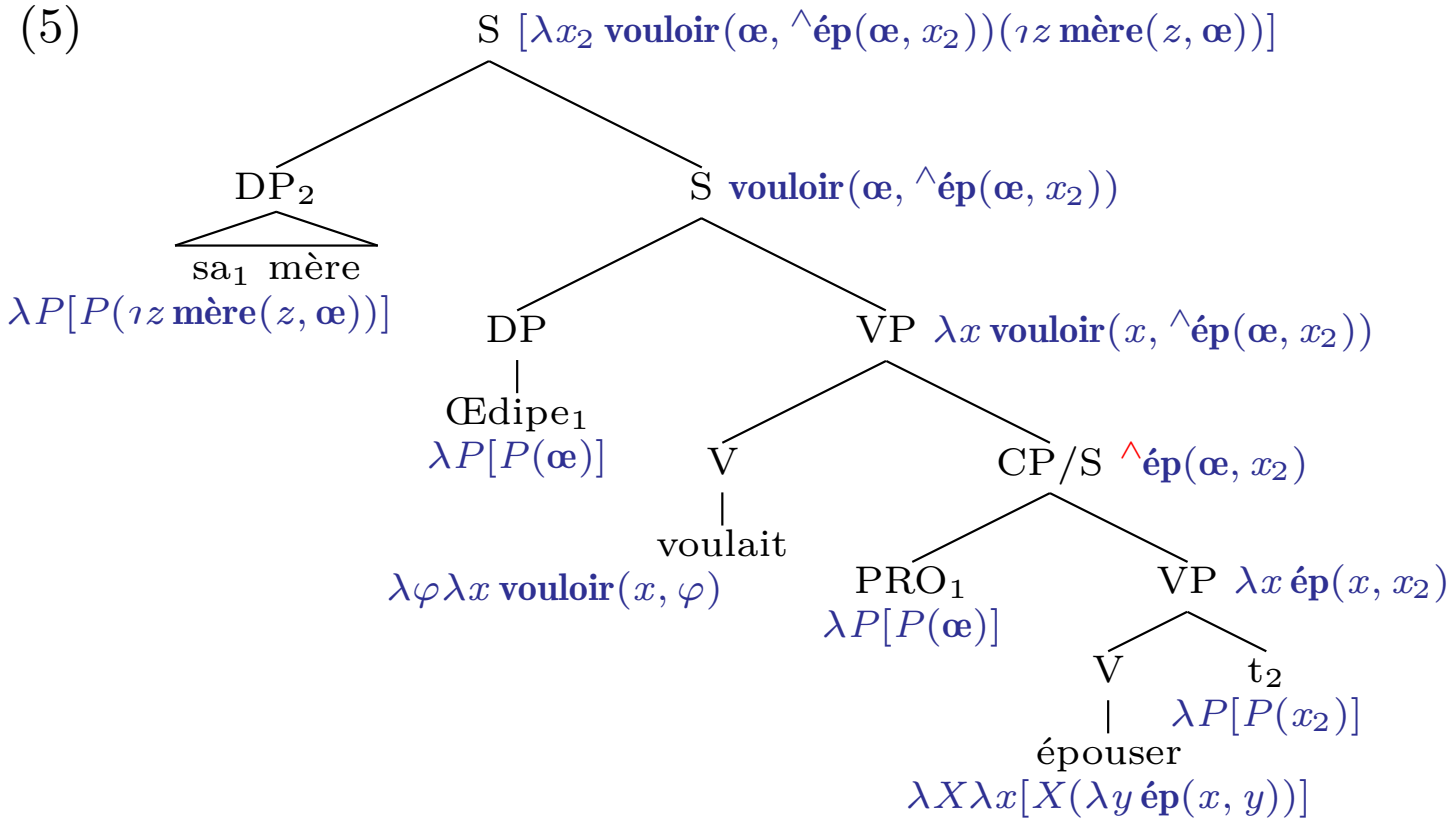
(5)



De dicto vs. de re

Œdipe voulait épouser sa mère (de re)

(5)



Lecture de dicto

Détail

- $[\![\textbf{ép}(\text{œ}, \imath z\, \textbf{mère}(z, \text{œ}))]\!]^{\mathcal{M},w',g} = 1$ ssi celle qui est la mère d'Œ. dans w' épouse Œ. dans w'.

Lecture de dicto

Détail

- $[\![\textbf{ép}(\text{œ}, \imath z\, \textbf{mère}(z, \text{œ}))]\!]^{\mathcal{M},w',g} = 1$ ssi celle qui est la mère d'Œ. dans w' épouse Œ. dans w'.
- $[\![\hat{}\textbf{ép}(\text{œ}, \imath z\, \textbf{mère}(z, \text{œ}))]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les mondes w' où celle qui est la mère d'Œ. dans w' épouse Œ.

Lecture de dicto

Détail

- $[\![\textbf{ép}(\text{œ}, \imath z\, \textbf{mère}(z, \text{œ}))]\!]^{\mathcal{M},w',g} = 1$ ssi celle qui est la mère d'Œ. dans w' épouse Œ. dans w'.
- $[\![\hat{}\textbf{ép}(\text{œ}, \imath z\, \textbf{mère}(z, \text{œ}))]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les mondes w' où celle qui est la mère d'Œ. dans w' épouse Œ.
- $[\![\textbf{vouloir}(\text{œ}, \hat{}\textbf{ép}(\text{œ}, \imath z\, \textbf{mère}(z, \text{œ})))]\!]^{\mathcal{M},w,g} = 1$ ssi dans w Œ. souhaite que w appartienne à l'ensemble de tous les mondes w' où celle qui est la mère d'Œ. dans w' épouse Œ.

Lecture de re

Détail

- $[\![\textbf{ép}(\textbf{œ}, x_2)]\!]^{\mathcal{M},w,g} = 1$ ssi Œ. épouse $g(x_2)$ dans w.

Lecture de re

Détail

- $[\![\textbf{ép}(\textbf{œ}, x_2)]\!]^{\mathcal{M},w,g} = 1$ ssi Œ. épouse $g(x_2)$ dans w.
- $[\![\hat{}\textbf{ép}(\textbf{œ}, x_2)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les mondes w' où Œ. épouse $g(x_2)$.

Lecture de re

Détail

- $[\![\textbf{ép}(\textbf{œ}, x_2)]\!]^{\mathcal{M},w,g} = 1$ ssi Œ. épouse $g(x_2)$ dans w.
- $[\![\hat{}\textbf{ép}(\textbf{œ}, x_2)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les mondes w' où Œ. épouse $g(x_2)$.
- $[\![\textbf{vouloir}(\textbf{œ}, \hat{}\textbf{ép}(\textbf{œ}, x_2))]\!]^{\mathcal{M},w,g} = 1$ ssi dans w Œ. souhaite que w appartienne à l'ensemble de tous les mondes mondes w' où Œ. épouse $g(x_2)$.

Lecture de re

Détail

- $[\![\textbf{ép}(\text{œ}, x_2)]\!]^{\mathcal{M},w,g} = 1$ ssi Œ. épouse $g(x_2)$ dans w.
- $[\![\textbf{\^{}ép}(\text{œ}, x_2)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les mondes w' où Œ. épouse $g(x_2)$.
- $[\![\textbf{vouloir}(\text{œ}, \textbf{\^{}ép}(\text{œ}, x_2))]\!]^{\mathcal{M},w,g} = 1$ ssi dans w Œ. souhaite que w appartienne à l'ensemble de tous les mondes mondes w' où Œ. épouse $g(x_2)$.
- $[\![\lambda x_2\ \textbf{vouloir}(\text{œ}, \textbf{\^{}ép}(\text{œ}, x_2))]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus qu'Œ. souhaite épouser dans w.

Lecture de re

Détail

- $[\![\textbf{ép}(\text{œ}, x_2)]\!]^{\mathcal{M},w,g} = 1$ ssi Œ. épouse $g(x_2)$ dans w.
- $[\![\hat{}\textbf{ép}(\text{œ}, x_2)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les mondes w' où Œ. épouse $g(x_2)$.
- $[\![\textbf{vouloir}(\text{œ}, \hat{}\textbf{ép}(\text{œ}, x_2))]\!]^{\mathcal{M},w,g} = 1$ ssi dans w Œ. souhaite que w appartienne à l'ensemble de tous les mondes mondes w' où Œ. épouse $g(x_2)$.
- $[\![\lambda x_2\, \textbf{vouloir}(\text{œ}, \hat{}\textbf{ép}(\text{œ}, x_2))]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus qu'Œ. souhaite épouser dans w.
- $[\![\imath z\, \textbf{mère}(z, \text{œ})]\!]^{\mathcal{M},w,g}$ = celle qui est la mère d'Œ. dans w.

Lecture de re

Détail

- $[\![\textbf{ép}(\text{œ}, x_2)]\!]^{\mathcal{M},w,g} = 1$ ssi Œ. épouse $g(x_2)$ dans w.
- $[\![\textbf{\^{}ép}(\text{œ}, x_2)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les mondes w' où Œ. épouse $g(x_2)$.
- $[\![\textbf{vouloir}(\text{œ}, \textbf{\^{}ép}(\text{œ}, x_2))]\!]^{\mathcal{M},w,g} = 1$ ssi dans w Œ. souhaite que w appartienne à l'ensemble de tous les mondes mondes w' où Œ. épouse $g(x_2)$.
- $[\![\lambda x_2\ \textbf{vouloir}(\text{œ}, \textbf{\^{}ép}(\text{œ}, x_2))]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus qu'Œ. souhaite épouser dans w.
- $[\![[\lambda x_2\ \textbf{vouloir}(\text{œ}, \textbf{\^{}ép}(\text{œ}, x_2))(\iota z\, \textbf{mère}(z, \text{œ}))]]\!]^{\mathcal{M},w,g} = 1$ ssi Œ. souhaite épouser dans w celle qui est la mère d'Œ. dans w.

Lecture de re

Détail

- $[\![\textbf{ép}(\text{œ}, x_2)]\!]^{\mathcal{M},w,g} = 1$ ssi Œ. épouse $g(x_2)$ dans w.
- $[\![\hat{}\textbf{ép}(\text{œ}, x_2)]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les mondes w' où Œ. épouse $g(x_2)$.
- $[\![\textbf{vouloir}(\text{œ}, \hat{}\textbf{ép}(\text{œ}, x_2))]\!]^{\mathcal{M},w,g} = 1$ ssi dans w Œ. souhaite que w appartienne à l'ensemble de tous les mondes mondes w' où Œ. épouse $g(x_2)$.
- $[\![\lambda x_2\, \textbf{vouloir}(\text{œ}, \hat{}\textbf{ép}(\text{œ}, x_2))]\!]^{\mathcal{M},w,g}$ = l'ensemble de tous les individus qu'Œ. souhaite épouser dans w.
- $[\![[\lambda x_2\, \textbf{vouloir}(\text{œ}, \hat{}\textbf{ép}(\text{œ}, x_2))(\imath z\, \textbf{mère}(z, \text{œ}))]]\!]^{\mathcal{M},w,g} = 1$ ssi Œ. souhaite épouser dans w celle qui est la mère d'Œ. dans w.
  Ici la β-réduction est interdite, car x_2 est dans la portée de $\hat{}$

Référence