Logique et Déduction - Cours 1 - Introduction, termes, arbres et réécriture

Pierre-Cyrille Héam

pheam [at] femto-st.fr

Licence 2 Informatique

Vidéos

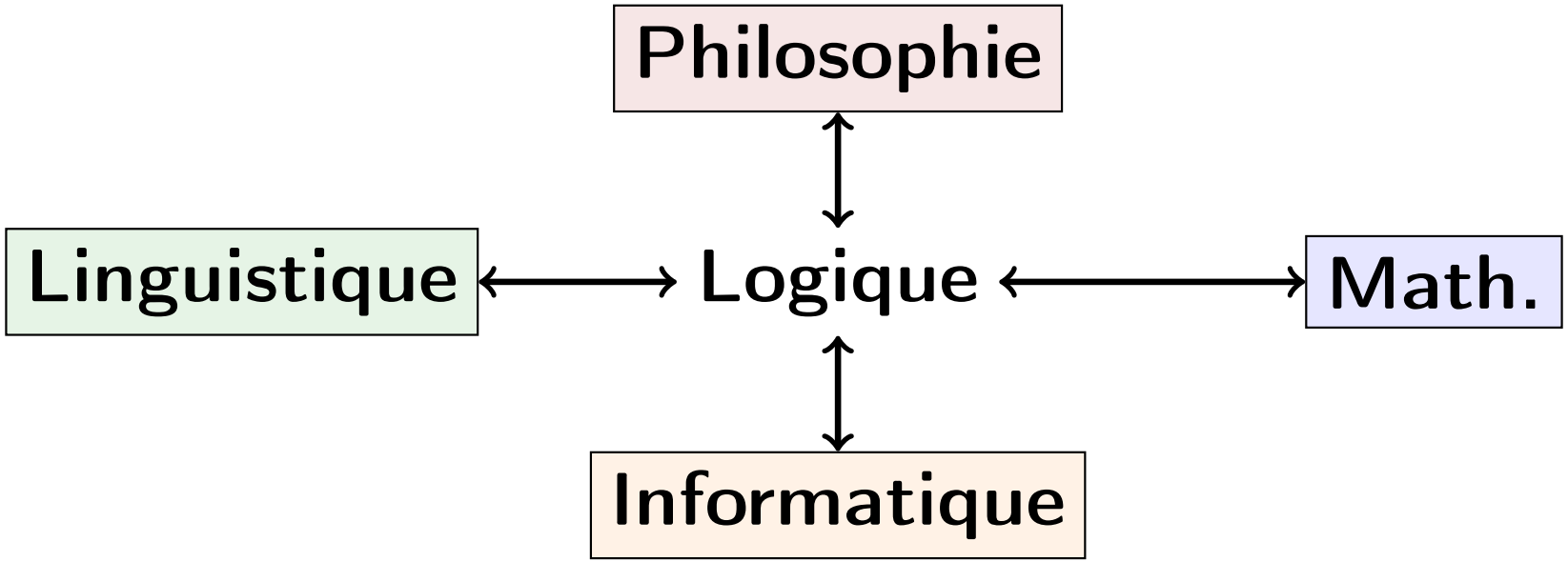
En raison de la crise sanitaire COVID des éléments de cours en vidéo sont disponibles pour certaines parties du cours.

La liste est consultable sur le site

`https://members.femto-st.fr/pierre-cyrille-heam/teaching`

dans les rubriques **Logique, réécriture** et **Python**.

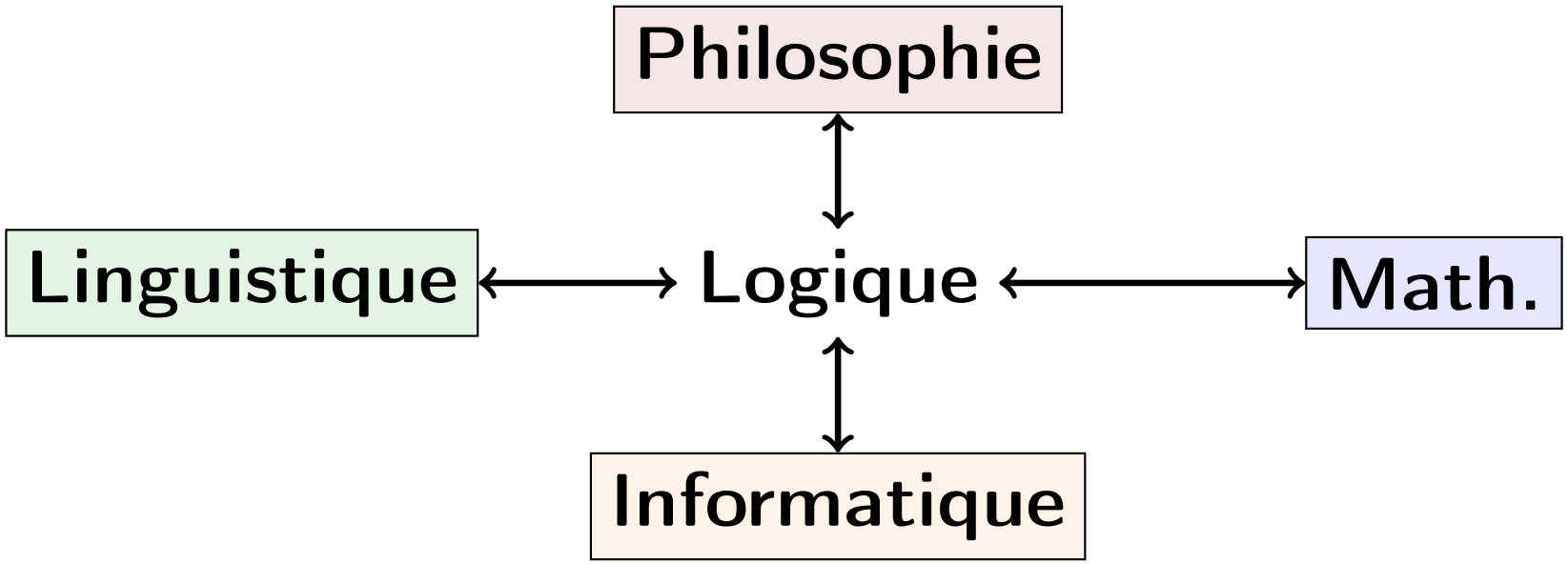
Logique et formalisation



L. Wittgenstein

"La logique n'est pas une théorie, mais une image réfléchie du monde." "La philosophie est une lutte contre la manière dont le langage ensorcelle notre intelligence."

Logique et formalisation



- Peut-on formaliser la pensée ? le monde ? le raisonnement philosophique ? Le temps ?
- Peut-on formaliser le raisonnement mathématique ? Scientifique ?
- Peut-on formaliser la syntaxe d'une langue ? sa sémantique ?
- Peut-on automatiser le raisonnement ?

Logique et automatisation

Linguistique

Sarah se dit que décidément elle ne trouverait jamais un homme qui supporte se défauts. Elle se tourna vers Will et lui dit : "Comme tant d'autres, tu vas me quitter".
Sarah savait que Will était un coureur de jupons, collectionnant les conquêtes. Elle se tourna vers lui et lui dit : "Comme tant d'autres, tu vas me quitter".

Applications : Traduction automatique, traitement automatique de documents, moteur de recherche thématique, etc.

Résolution de contraintes

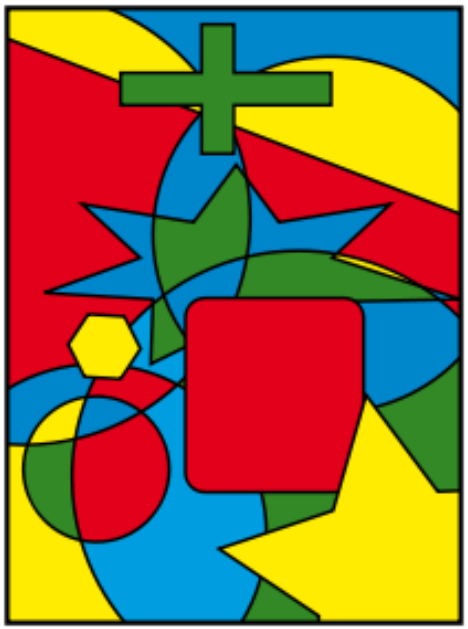
4			7			5		
3			8					1
7			9					8
	6			3			1	
	9			7			5	
	8			1		7	9	
		2			6	9		
		5					3	
					4	2		

Applications : Résolution de problèmes, emplois du temps, jeux, etc.

Logique et automatisation

Preuve automatique de théorèmes

Toute figure (carte) dessinée avec des traits peut être colorée avec 4 couleurs, sans que deux régions adjacentes ne soient de la même couleur.



source `Wikipedia`

- Application en coloriage de cartes (sans enclave, DOM-TOM,...),
- Preuves fausses en 1879 (Kempe) et 1880 (Tait),
- Preuve par ordinateur en 1976 (Appel et Haken), (environ 1500 cas),
- Preuve simplifiée plusieurs fois, puis certifiée, en 2005 (Gonthier et Werner),
- Pas de preuve *classique* connue actuellement

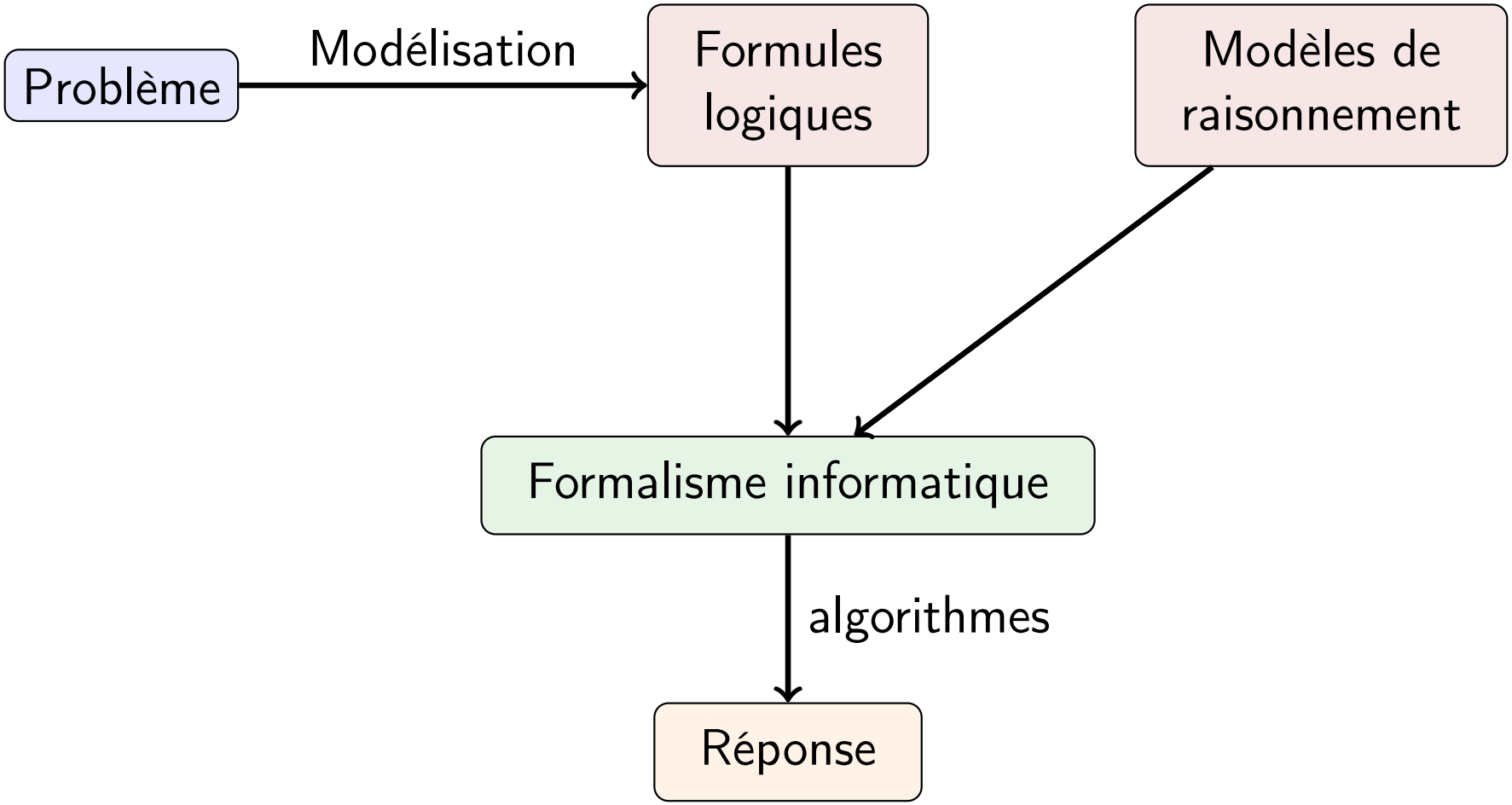
Logique et automatisation

Fiabilité logicielle, preuve de programme

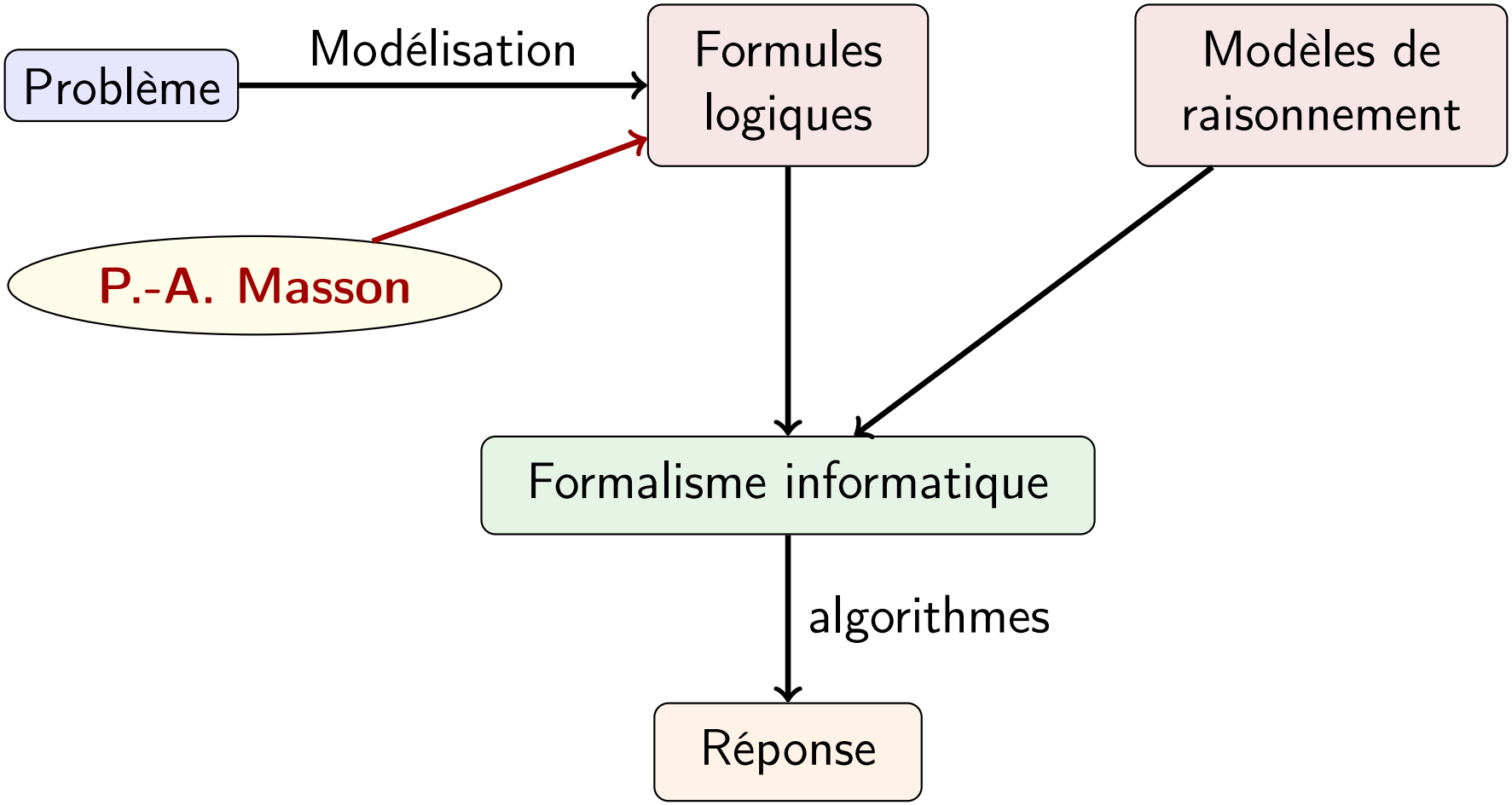
- En 1996, la fusée ariane 5 explose au décollage : en cause un dépassement mémoire donnant une mauvaise trajectoire, puis le déclenchement du système de sécurité d'autodestruction. Coût environ 200 millions d'euros.
- En 2016, dans des conditions très particulières l'ordinateur de bord de certaines voiture GM considérait être en phase de test et ne bloquait pas les ceinture sur un choc : 1 mort, 3 blessés.
- ...

La logique joue un rôle central dans la conception de logiciels fiables.
Applications : systèmes critiques, systèmes embarquées, noyaux, etc.
`https://www.rocketprojet.com/29-bugs-informatiques-catastrophe/`
`https://en.wikipedia.org/wiki/List_of_software_bugs`

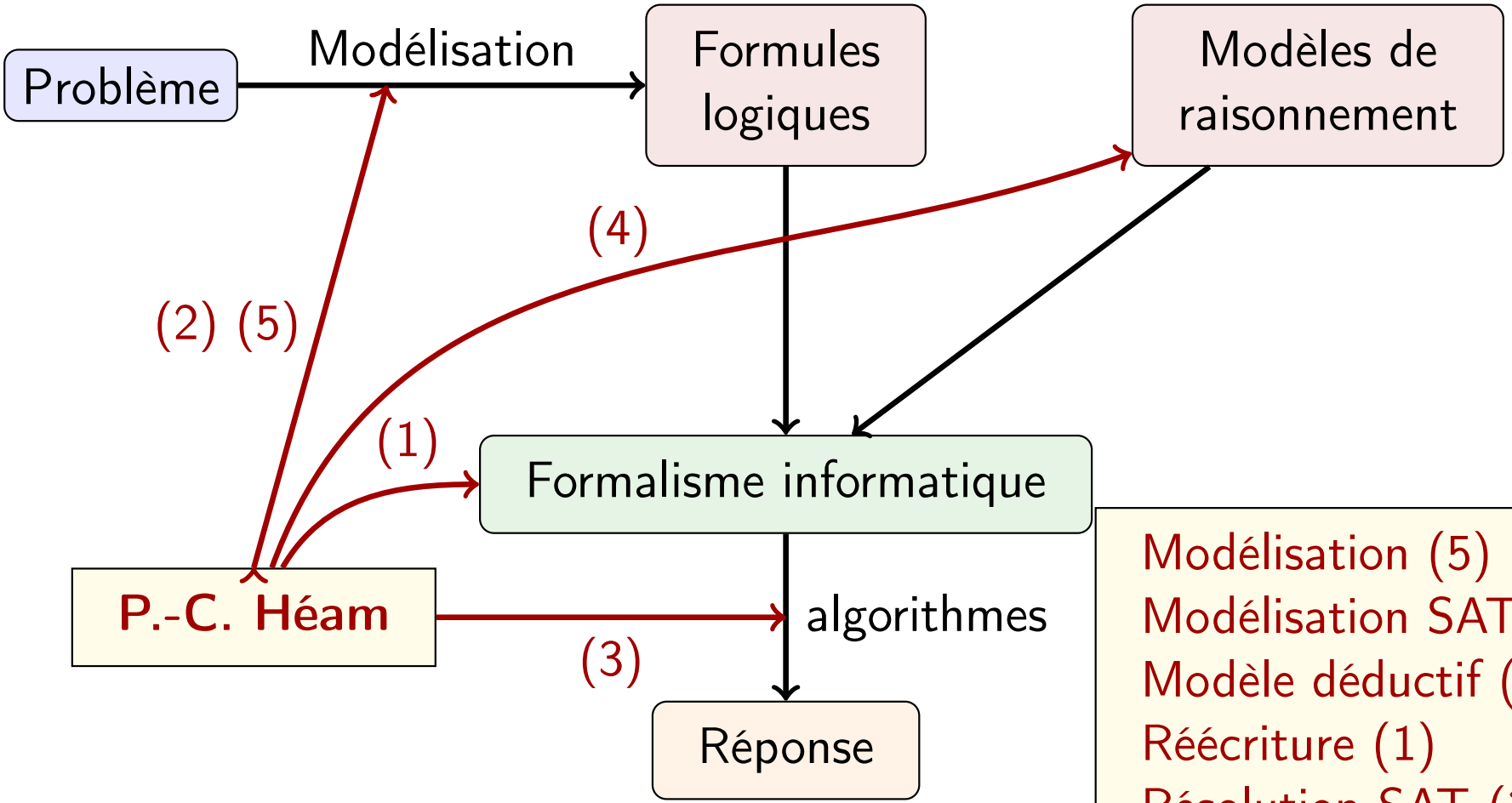
Logique et automatisation - organisation du module



Logique et automatisation - organisation du module

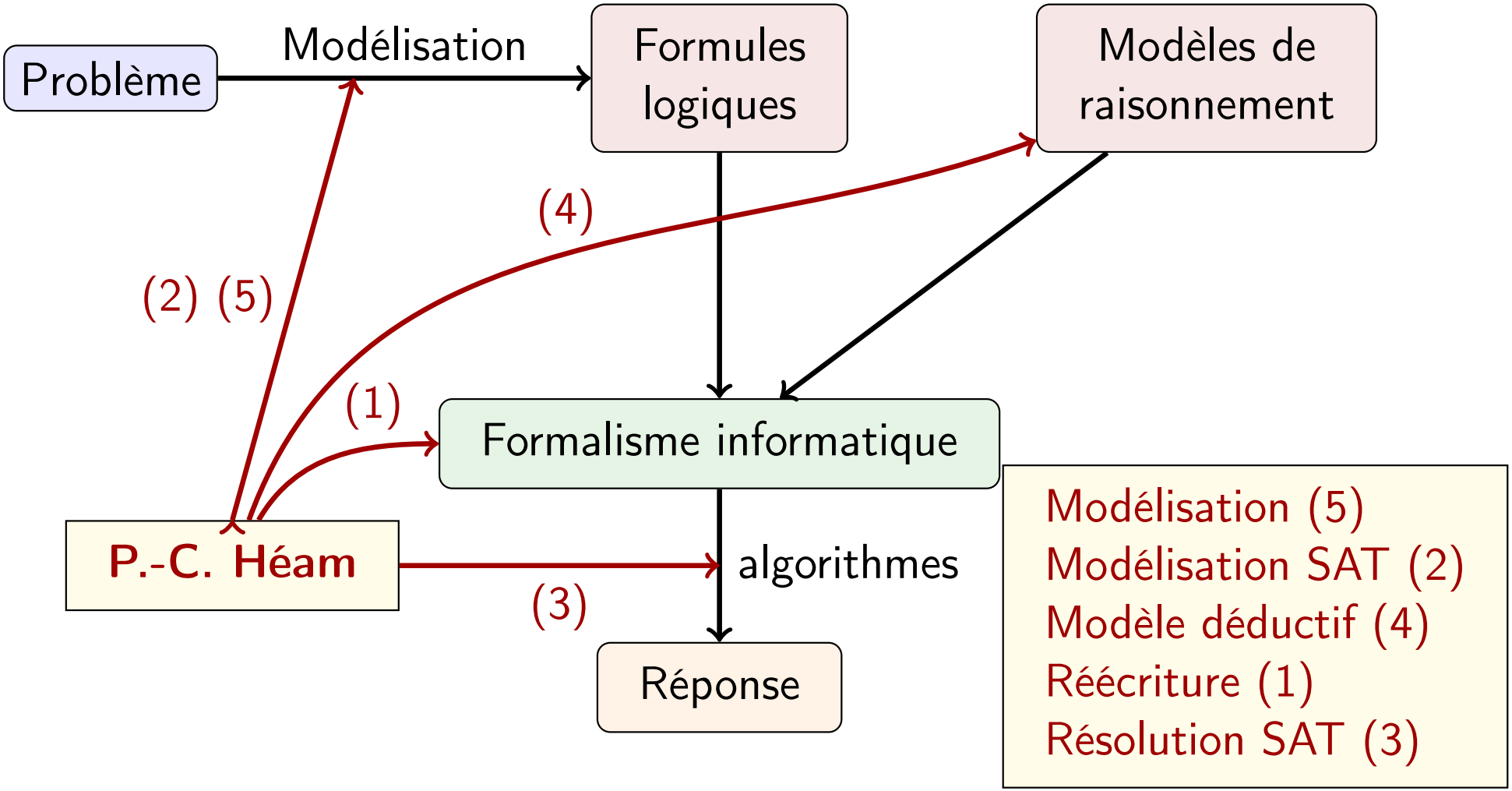


Logique et automatisation - organisation du module



Modélisation (5)
Modélisation SAT (2)
Modèle déductif (4)
Réécriture (1)
Résolution SAT (3)

Logique et automatisation - organisation du module



Autres approches : Théorie des langages, Machine Learning

Plan

Structuration de l'information

linéaire : Mots, chaînes de caractères, tableaux, listes,

arborescent : Expression, arbres, termes,

relationnel : Graphes,

fonctionnel : λ-calcul,

La réécriture est un formalisme efficace pour modéliser les transformations (programmes, raisonnement automatiques,...) sur les différentes structures.

En mémoire, physiquement c'est un adressage de blocs qui est utilisé. Dans un fichier, les éléments sont codés linéairement.

Exemples d'*expressions*

Expression (ici) : chaîne de caractères avec parenthésage (fournissant une structuration à l'expression).

Expressions d'appels de fonctions

```
def f(x,y):
    return x+y

def h(x):
    return x**x

def g(x,y):
    return x+2*y

f(2,f(h(3),g(4,h(1))))
```

Exemples d'*expressions*

Expression (ici) : chaîne de caractères avec parenthésage (fournissant une structuration à l'expression). Ici les parenthèse sont des balises.

Code HTML (simple)

```
<html>
<head><title>...</title></head>
<body>
<!-- Menu -->
<ul class=...>
  <li><a href="...">Home</a>
  <li><a href="....">Biere</a>
</ul>
<h1>Ma p page </h1>
<p>...
<p>...
<p>...
<!-- commentaire-->
<address> ici </address>
</body>
</html>
```

Exemples d'*expressions*

Expression (ici) : chaîne de caractères avec parenthésage (fournissant une structuration à l'expression).

Expressions arithmético-fonctionnelles

$$(3 + 5) * 4 - 2$$

$$\frac{3x^2 + \sin(x)}{\sqrt{x}} = \left(3x^2 + \sin(x)\right) / \sqrt{x}$$

Exemples d'*expressions*

Expression (ici) : chaîne de caractères avec parenthésage (fournissant une structuration à l'expression).

Formules propositionnelles

$$(A \vee B) \Rightarrow (\neg B \wedge C)$$

Notation des expressions

- **Infixée ou Infixe** (classique) : $(3 + 5) * (4 - 1)$ Intuitive pour l'humain mais peu adaptée à l'automatisation.

Notation des expressions

- **Infixée ou Infixe** (classique) : $(3 + 5) * (4 - 1)$ Intuitive pour l'humain mais peu adaptée à l'automatisation.
- **Préfixée ou Préfixe ou Polonaise** (comme pour les appels de fonctions) :

$$\texttt{Multiplie}(\texttt{Somme}(3, 5), \texttt{Diff}(4, 1))$$

```
Multiplie Somme 3 5 Diff 4 1
```

Notation des expressions

- **Infixée ou Infixe** (classique) : $(3 + 5) * (4 - 1)$ Intuitive pour l'humain mais peu adaptée à l'automatisation.
- **Préfixée ou Préfixe ou Polonaise** (comme pour les appels de fonctions) :

$$\texttt{Multiplie}(\texttt{Somme}(3, 5), \texttt{Diff}(4, 1))$$

$$\texttt{Multiplie Somme 3 5 Diff 4 1}$$

- **Postfixée ou Postfixe ou Polonaise inverse** :

$$((3, 5)\texttt{Somme}, (4, 1)\texttt{Diff})\texttt{Multiplie}$$

$$\texttt{3 5 Somme 4 1 Diff Multiplie}$$

RPN : parenthèse historique

La notation polonaise inversée a des avantages :

- Utilisation d'une pile (calculs plus rapides),
- Expressions plus courtes (moins de parenthèses).

source Wikipedia

Les calculatrices HP jusqu'aux années 2010 utilisaient la notation polonaise inverse.

Utilisée aussi dans : `Postscript` (imprimantes), `bibtex`, langages orientés *pile*,...

Exercices

On considère l'expression arithmétique

$$(((3-5)+(2*4))*(7+2)).$$

1. L'écrire sous forme préfixe.

2. L'écrire sous forme postfixe (polonaise inverse).

Exercices

On considère l'expression arithmétique

$$(((3-5)+(2*4))*(7+2)).$$

1. L'écrire sous forme préfixe.

   (+(-(3,5),(2,4)),+(7,2))

2. L'écrire sous forme postfixe (polonaise inverse).

Exercices

On considère l'expression arithmétique

$$(((3-5)+(2*4))*(7+2)).$$

1. L'écrire sous forme préfixe.

   (+(-(3,5),(2,4)),+(7,2))

2. L'écrire sous forme postfixe (polonaise inverse).

   (((3,5)-,(2,4)*)+,(7,2)+)*

Plan

Termes clos : définitions

Sur le plan formel, on utilise la notion de terme, pour designer une expression parenthésée en informatique.

Termes clos : définitions

Sur le plan formel, on utilise la notion de terme, pour designer une expression parenthésée en informatique.

- Soit Σ un ensemble fini, appelé alphabet.
  $\Sigma = \{f, g, h, A, B\}$

Termes clos : définitions

Sur le plan formel, on utilise la notion de terme, pour designer une expression parenthésée en informatique.

- Soit Σ un ensemble fini, appelé alphabet.
  $\Sigma = \{f, g, h, A, B\}$
- Soit une fonction arity de Σ dans les entiers positifs.
  $\mathrm{arity}(f) = 2, \mathrm{arity}(g) = 3, \mathrm{arity}(h) = 1, \mathrm{arity}(A) = \mathrm{arity}(B) = 0$
  Les éléments d'arité 0 sont appelés constantes.

Termes clos : définitions

Sur le plan formel, on utilise la notion de terme, pour designer une expression parenthésée en informatique.

- Soit Σ un ensemble fini, appelé alphabet.
  $\Sigma = \{f, g, h, A, B\}$
- Soit une fonction arity de Σ dans les entiers positifs.
  $\mathrm{arity}(f) = 2, \mathrm{arity}(g) = 3, \mathrm{arity}(h) = 1, \mathrm{arity}(A) = \mathrm{arity}(B) = 0$
  Les éléments d'arité 0 sont appelés constantes.
- (définition informelle) Un terme clos (sur Σ muni de arity) est une expression bien parenthésée finie respectant l'arité, en notation préfixée.
  $g(f(A, B), h(h(A), h(f(A, B))))$

Exemple des formules propositionnelles atomiques

On considère l'ensemble des variables propositionnelles sans variables
Une formule propositionnelle est un terme clos sur

- $\Sigma = \{\wedge, \vee, \Rightarrow, \Leftrightarrow, \neg, \mathrm{True}, \mathrm{False}\}$

Exemple des formules propositionnelles atomiques

On considère l'ensemble des variables propositionnelles sans variables
Une formule propositionnelle est un terme clos sur

- $\Sigma = \{\wedge, \vee, \Rightarrow, \Leftrightarrow, \neg, \text{True}, \text{False}\}$
- $\text{arity}(\wedge) = \text{arity}(\vee) = \text{arity}(\Rightarrow) = \text{arity}(\Leftrightarrow) = 2$
- $\text{arity}(\neg) = 1$
- $\text{arity}(\text{True}) = \text{arity}(\text{False}) = 0$

Exemple des formules propositionnelles atomiques

On considère l'ensemble des variables propositionnelles sans variables
Une formule propositionnelle est un terme clos sur

- $\Sigma = \{\wedge, \vee, \Rightarrow, \Leftrightarrow, \neg, \mathrm{True}, \mathrm{False}\}$
- $\mathrm{arity}(\wedge) = \mathrm{arity}(\vee) = \mathrm{arity}(\Rightarrow) = \mathrm{arity}(\Leftrightarrow) = 2$
- $\mathrm{arity}(\neg) = 1$
- $\mathrm{arity}(\mathrm{True}) = \mathrm{arity}(\mathrm{False}) = 0$

Exemple : $\vee(\mathit{True}, \neg \mathit{False})$ qui est l'écriture préfixe de la formule $\mathit{True} \vee \neg \mathit{False}$.

Termes clos : définition inductive

Soit Σ un alphabet fini muni d'une fonction d'arité arity. L'ensemble des termes clos sur Σ, arity, noté $\mathcal{T}(\Sigma)$, est le plus petit ensemble vérifiant :

- Si $f \in \Sigma$ et $\mathrm{arity}(f) = 0$, alors $f \in \mathcal{T}(\Sigma)$.
- Si $f \in \Sigma$ et $\mathrm{arity}(f) = k \geq 1$ et $t_1, \ldots, t_k \in \mathcal{T}(\Sigma)$, alors

$$f(t_1, \ldots, t_k) \in TS.$$

Termes clos : définition inductive

Soit Σ un alphabet fini muni d'une fonction d'arité arity. L'ensemble des termes clos sur Σ, arity, noté $\mathcal{T}(\Sigma)$, est le plus petit ensemble vérifiant :

- Si $f \in \Sigma$ et $\mathrm{arity}(f) = 0$, alors $f \in \mathcal{T}(\Sigma)$.
- Si $f \in \Sigma$ et $\mathrm{arity}(f) = k \geq 1$ et $t_1, \ldots, t_k \in \mathcal{T}(\Sigma)$, alors

$$f(t_1, \ldots, t_k) \in TS.$$

Exemple : $\Sigma = \{g, h, A, B\}$, avec $\mathrm{arity}(g) = 2$, $\mathrm{arity}(h) = 1$ $\mathrm{arity}(A) = \mathrm{arity}(B) = 0$.

Termes clos : définition inductive

Soit Σ un alphabet fini muni d'une fonction d'arité arity. L'ensemble des termes clos sur Σ, arity, noté $\mathcal{T}(\Sigma)$, est le plus petit ensemble vérifiant :

- Si $f \in \Sigma$ et $\mathrm{arity}(f) = 0$, alors $f \in \mathcal{T}(\Sigma)$.
- Si $f \in \Sigma$ et $\mathrm{arity}(f) = k \geq 1$ et $t_1, \ldots, t_k \in \mathcal{T}(\Sigma)$, alors

$$f(t_1, \ldots, t_k) \in TS.$$

Exemple : $\Sigma = \{g, h, A, B\}$, avec $\mathrm{arity}(g) = 2$, $\mathrm{arity}(h) = 1$ $\mathrm{arity}(A) = \mathrm{arity}(B) = 0$.

- A est un terme clos, de même que B.

Termes clos : définition inductive

Soit Σ un alphabet fini muni d'une fonction d'arité arity. L'ensemble des termes clos sur Σ, arity, noté $\mathcal{T}(\Sigma)$, est le plus petit ensemble vérifiant :

- Si $f \in \Sigma$ et $\mathrm{arity}(f) = 0$, alors $f \in \mathcal{T}(\Sigma)$.
- Si $f \in \Sigma$ et $\mathrm{arity}(f) = k \geq 1$ et $t_1, \ldots, t_k \in \mathcal{T}(\Sigma)$, alors

$$f(t_1, \ldots, t_k) \in TS.$$

Exemple : $\Sigma = \{g, h, A, B\}$, avec $\mathrm{arity}(g) = 2$, $\mathrm{arity}(h) = 1$ $\mathrm{arity}(A) = \mathrm{arity}(B) = 0$.

- A est un terme clos, de même que B.
- $h(A), h(B), g(A, A), g(A, B), g(B, A), g(B, B)$ sont des termes clos.

Termes clos : définition inductive

Soit Σ un alphabet fini muni d'une fonction d'arité arity. L'ensemble des termes clos sur Σ, arity, noté $\mathcal{T}(\Sigma)$, est le plus petit ensemble vérifiant :

- Si $f \in \Sigma$ et $\mathrm{arity}(f) = 0$, alors $f \in \mathcal{T}(\Sigma)$.
- Si $f \in \Sigma$ et $\mathrm{arity}(f) = k \geq 1$ et $t_1, \ldots, t_k \in \mathcal{T}(\Sigma)$, alors

$$f(t_1, \ldots, t_k) \in TS.$$

Exemple : $\Sigma = \{g, h, A, B\}$, avec $\mathrm{arity}(g) = 2$, $\mathrm{arity}(h) = 1$ $\mathrm{arity}(A) = \mathrm{arity}(B) = 0$.

- A est un terme clos, de même que B.
- $h(A), h(B), g(A, A), g(A, B), g(B, A), g(B, B)$ sont des termes clos.
- $h(h(A)), h(h(B)), h(g(A, A)), \ldots, h(g(B, B))$, $f(h(A), h(A)), f(h(A), h(B)), \ldots$ sont des termes clos.
- $\ldots$

Termes et Arbres

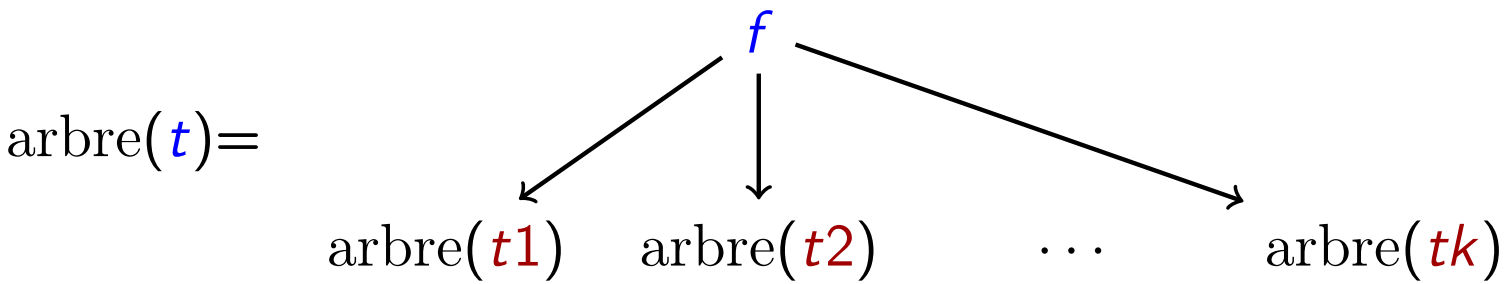
On représente souvent les termes sous forme d'arbre [1], à l'aide de la procédure suivante :

- Si t est un terme ne contenant qu'une constante, alors

$$\text{arbre}(t) = t$$

,

- Si $t = f(t_1, \ldots, t_k)$, alors



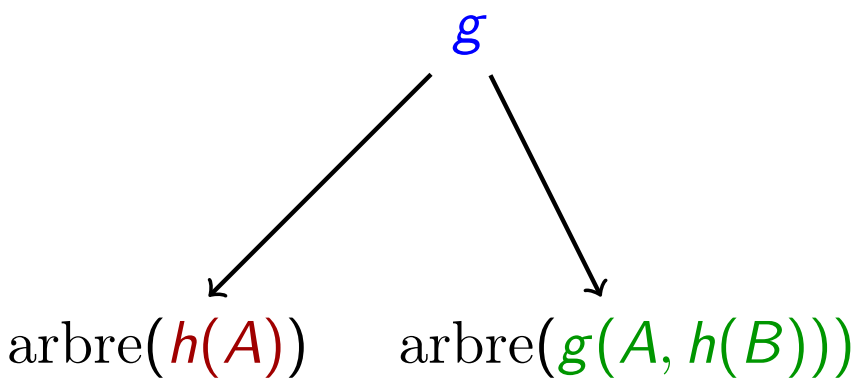
1. On ne donnera pas ici de définition formelle d'arbre

Exemple (arbres)

$\text{arbre}(g(h(A), g(A, h(B)))) =$

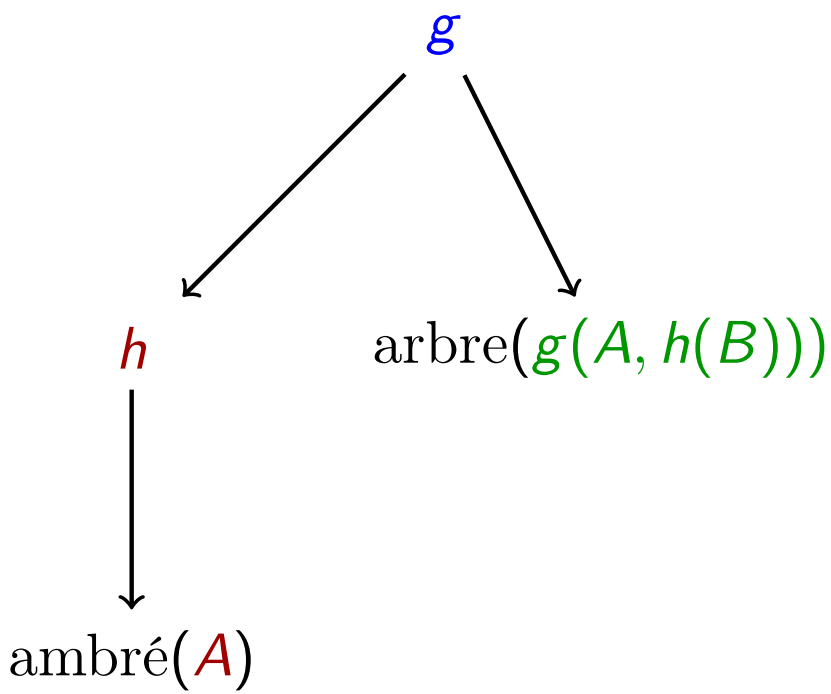
Exemple (arbres)

$\text{arbre}(g(h(A), g(A, h(B)))) =$



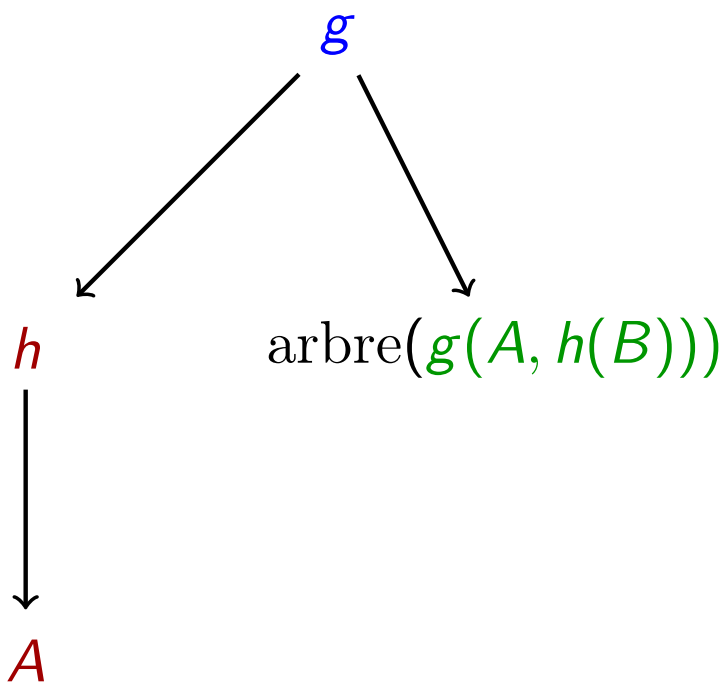
Exemple (arbres)

$\text{arbre}(g(h(A), g(A, h(B)))) =$



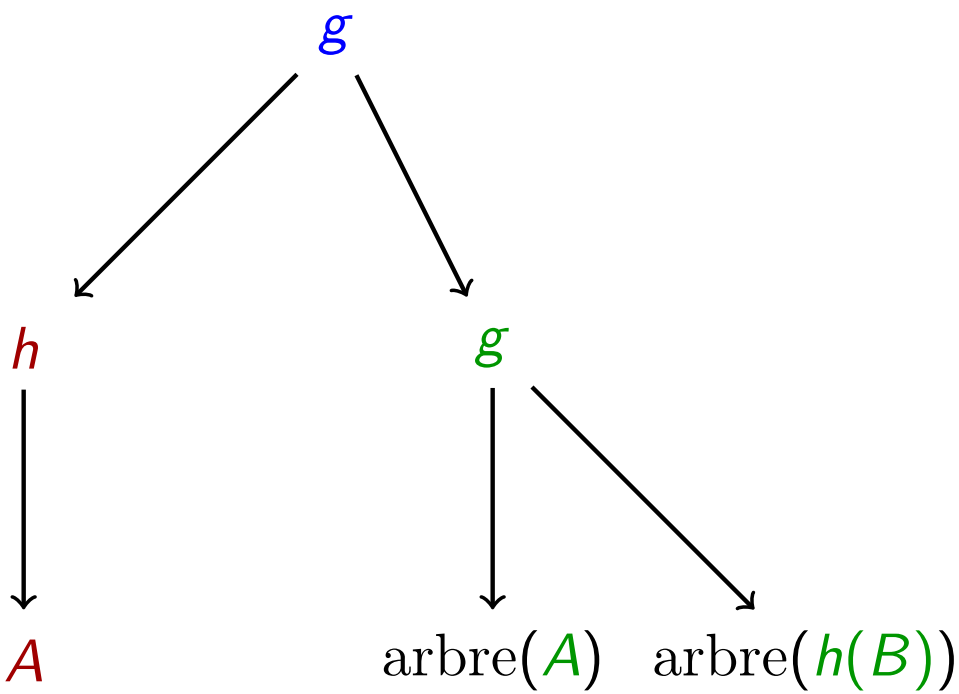
Exemple (arbres)

$\text{arbre}(g(h(A), g(A, h(B)))) =$



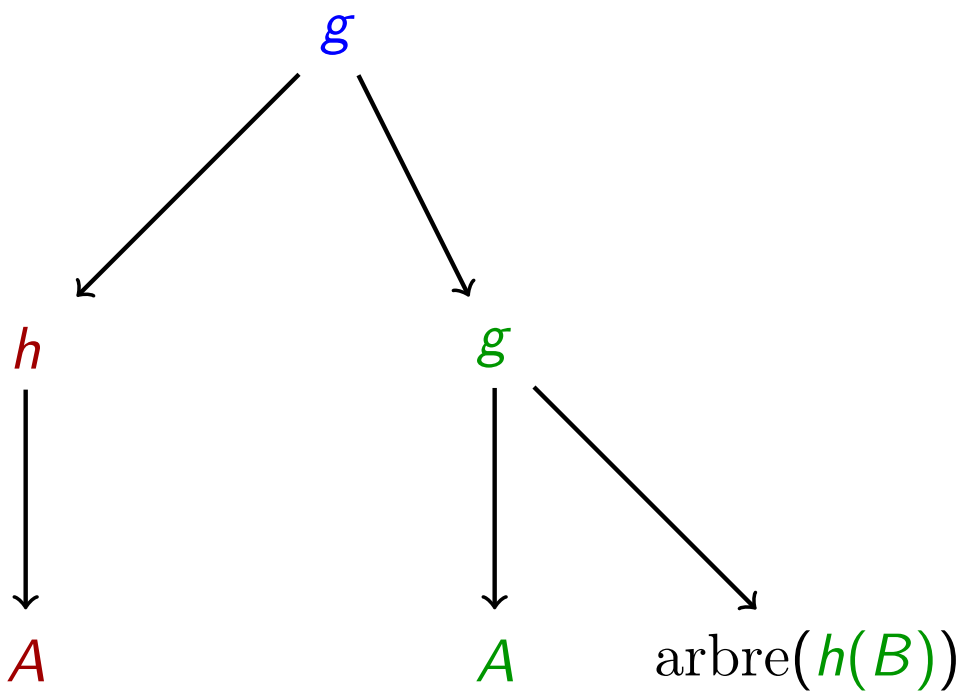
Exemple (arbres)

$\text{arbre}(g(h(A), g(A, h(B)))) =$



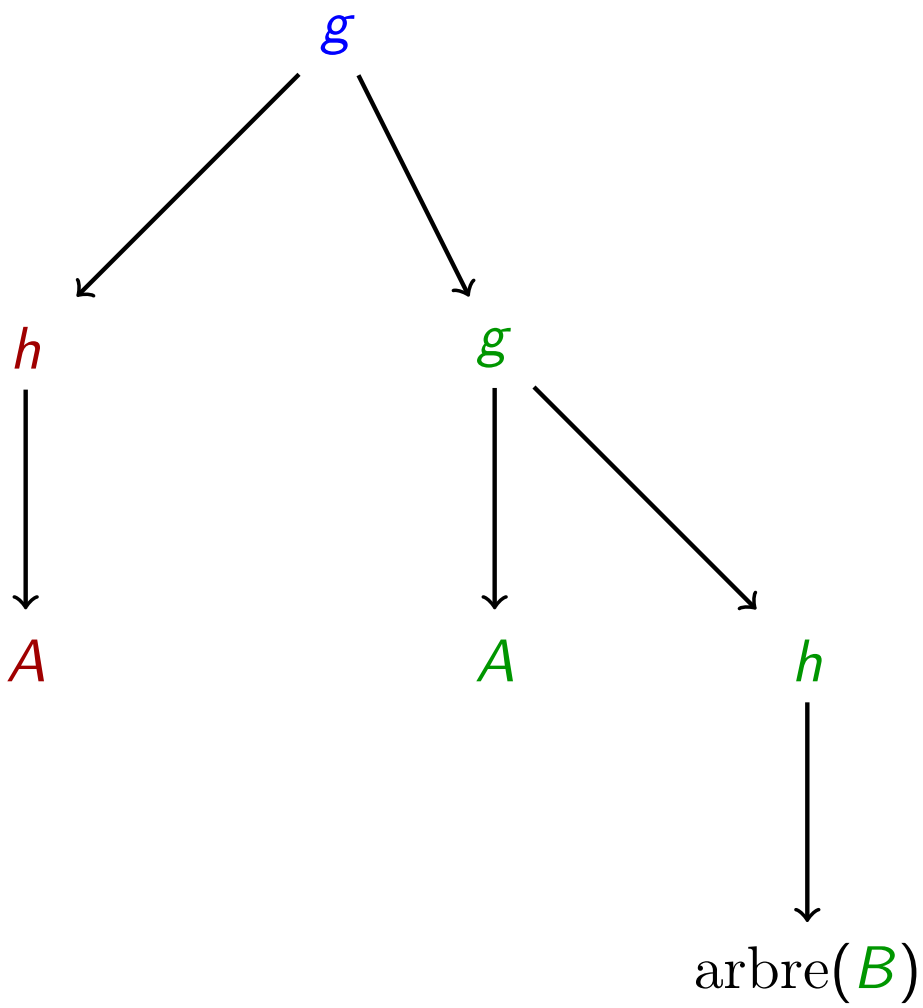
Exemple (arbres)

$\text{arbre}(g(h(A), g(A, h(B)))) =$



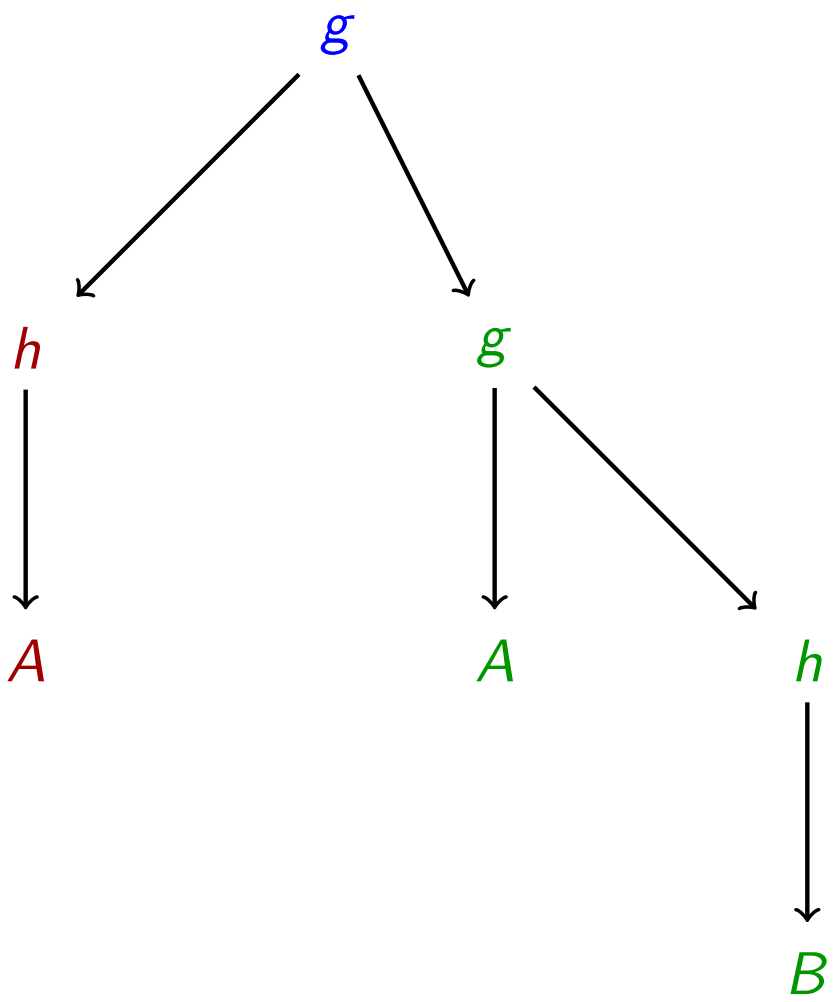
Exemple (arbres)

$\text{arbre}(g(h(A), g(A, h(B)))) =$



Exemple (arbres)

$\text{arbre}(g(h(A), g(A, h(B)))) =$



Termes (non clos)

Soit Σ un alphabet fini muni d'une fonction d'arité et $\mathcal{X}$ un ensemble infini dénombrable tel que $\Sigma \cap \mathcal{X} = \emptyset$.

Termes (non clos)

Soit Σ un alphabet fini muni d'une fonction d'arité et $\mathcal{X}$ un ensemble infini dénombrable tel que $\Sigma \cap \mathcal{X} = \emptyset$.

L'ensemble des termes sur $\Sigma, \mathcal{X}$, noté $\mathcal{T}(\Sigma, \mathcal{X})$, est le plus petit ensemble vérifiant :

- Si $f \in \Sigma$ et $\text{arity}(f) = 0$, alors $f \in \mathcal{T}(\Sigma, \mathcal{X})$.
- Si $X \in \mathcal{X}$ et $\text{arity}(f) = 0$, alors $f \in \mathcal{T}(\Sigma, \mathcal{X})$.
- Si $f \in \Sigma$ et $\text{arity}(f) = k \geq 1$ et $t_1, \ldots, t_k \in \mathcal{T}(\Sigma, \mathcal{X})$, alors

$$f(t_1, \ldots, t_k) \in \mathcal{T}(\Sigma, \mathcal{X}).$$

Comme les termes clos, les éléments de $\mathcal{X}$ sont considérés comme des constantes dans la construction. Les éléments de $\mathcal{X}$ sont appelés variables.
Exemple : $g(h(x), g(A, g(x, y)))$
On construit les arbres de la même façon.

Exercices

On considère l'alphabet $\Sigma = \{A, B, C, h, f, g\}$ où A, B, C sont des constantes, h est d'arité 1, f et g sont d'arité 2.

1. Les expressions suivantes sont-elles des termes clos ?

$$f(h(A, B), C) \quad \text{et} \quad h(h(f(A, A)))$$

2. Quels sont tous les termes que l'on ne peut écrire en n'utilisant exactement qu'une paire de parenthèses ?

3. Donner l'arbre du terme $h(h(f(A, g(B, h(C)))))$.

Exercices

On considère l'alphabet $\Sigma = \{A, B, C, h, f, g\}$ où A, B, C sont des constantes, h est d'arité 1, f et g sont d'arité 2.

1. Les expressions suivantes sont-elles des termes clos ?

$$f(h(A, B), C) \quad \text{et} \quad h(h(f(A, A)))$$

Pas la première car l'arité de h n'est pas bonne.

2. Quels sont tous les termes que l'on ne peut écrire en n'utilisant exactement qu'une paire de parenthèses ?

3. Donner l'arbre du terme $h(h(f(A, g(B, h(C)))))$.

Exercices

On considère l'alphabet $\Sigma = \{A, B, C, h, f, g\}$ où A, B, C sont des constantes, h est d'arité 1, f et g sont d'arité 2.

1. Les expressions suivantes sont-elles des termes clos ?

$$f(h(A, B), C) \quad \text{et} \quad h(h(f(A, A)))$$

Pas la première car l'arité de h n'est pas bonne.

2. Quels sont tous les termes que l'on ne peut écrire en n'utilisant exactement qu'une paire de parenthèses ?
$h(A)$, $h(B)$, $h(C)$, $f(A, A)$, $f(A, B)$, $\ldots f(C, C)$, $g(A, A)$, $g(A, B)$, $\ldots g(C, C)$.
3. Donner l'arbre du terme $h(h(f(A, g(B, h(C)))))$.

Exercices

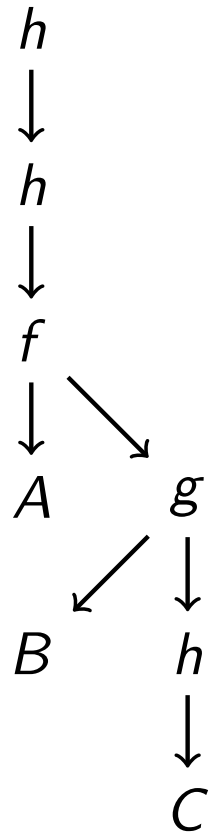
On considère l'alphabet $\Sigma = \{A, B, C, h, f, g\}$ où A, B, C sont des constantes, h est d'arité 1, f et g sont d'arité 2.

1. Les expressions suivantes sont-elles des termes clos ?

$$f(h(A, B), C) \quad \text{et} \quad h(h(f(A, A)))$$

Pas la première car l'arité de h n'est pas bonne.

2. Quels sont tous les termes que l'on ne peut écrire en n'utilisant exactement qu'une paire de parenthèses ?
$h(A)$, $h(B)$, $h(C)$, $f(A, A)$, $f(A, B)$, $\ldots f(C, C)$, $g(A, A)$, $g(A, B)$, $\ldots g(C, C)$.
3. Donner l'arbre du terme $h(h(f(A, g(B, h(C)))))$.



Termes et positions

Une position est une suite finie d'entiers strictement positifs, notée [2] en séparant les éléments par un $\cdot$.
La liste vide est notée ε.
Exemple : $1 \cdot 3 \cdot 1 \cdot 1$

2. S'il n'y a pas d'ambiguïté possible, on omet en général le séparateur.

Termes et positions

Une position est une suite finie d'entiers strictement positifs, notée [2] en séparant les éléments par un $\cdot$.
La liste vide est notée ε.
Exemple : $1 \cdot 3 \cdot 1 \cdot 1$
L'ensemble des positions, noté $\mathrm{pos}(t)$ d'un terme t est le plus petit ensemble vérifiant :

- Si $t \in \Sigma$, alors $\mathrm{pos}(t) = \{\varepsilon\}$;
- Si $t = f(t_1, \ldots, t_k)$, alors

$$\mathrm{pos}(t) = \{i \cdot p \mid 1 \leq i \leq k,\ p \in \mathrm{pos}(t_i)\} \cup \{\varepsilon\}$$

2. S'il n'y a pas d'ambiguïté possible, on omet en général le séparateur.

Termes et positions : exemple

Considérons le terme

$$t = g(A, g(B, h(A, B)))$$

$\mathrm{pos}(t)$ est constitué de

- ε, et
- $1 \cdot \mathrm{pos}(A)$, et
- $2 \cdot \mathrm{pos}(g(B, h(A, B))$

Termes et positions : exemple

Considérons le terme

$$t = g(A, g(B, h(A, B)))$$

$\mathrm{pos}(t)$ est constitué de

- ε, et
- $1 \cdot \mathrm{pos}(A)$, et
  $1 \cdot \mathrm{pos}(A) = 1 \cdot \{\varepsilon\} = \{1\}$
- $2 \cdot \mathrm{pos}(g(B, h(A, B))$

Termes et positions : exemple

Considérons le terme

$$t = g(A, g(B, h(A, B)))$$

$\mathrm{pos}(t)$ est constitué de

- ε, et
- $1 \cdot \mathrm{pos}(A)$, et
  $1 \cdot \mathrm{pos}(A) = 1 \cdot \{\varepsilon\} = \{1\}$
- $2 \cdot \mathrm{pos}(g(B, h(A, B))$ qui est l'union de
  - $2 \cdot \{\varepsilon\} = \{2\}$ et
  - $2 \cdot 1 \cdot \mathrm{pos}(B) = \{2 \cdot 1\}$ et
  - $2 \cdot 2 \cdot \mathrm{pos}(h(A, B))$

Termes et positions : exemple

Considérons le terme

$$t = g(A, g(B, h(A, B)))$$

$\mathrm{pos}(t)$ est constitué de

- ε, et
- $1 \cdot \mathrm{pos}(A)$, et
  $1 \cdot \mathrm{pos}(A) = 1 \cdot \{\varepsilon\} = \{1\}$
- $2 \cdot \mathrm{pos}(g(B, h(A, B))$ qui est l'union de
  - $2 \cdot \{\varepsilon\} = \{2\}$ et
  - $2 \cdot 1 \cdot \mathrm{pos}(B) = \{2 \cdot 1\}$ et
  - $2 \cdot 2 \cdot \mathrm{pos}(h(A, B))$ qui vaut $\{2 \cdot 2, 2 \cdot 2 \cdot 1, 2 \cdot 2 \cdot 2\}$

Termes et positions : exemple

Considérons le terme

$$t = g(A, g(B, h(A, B)))$$

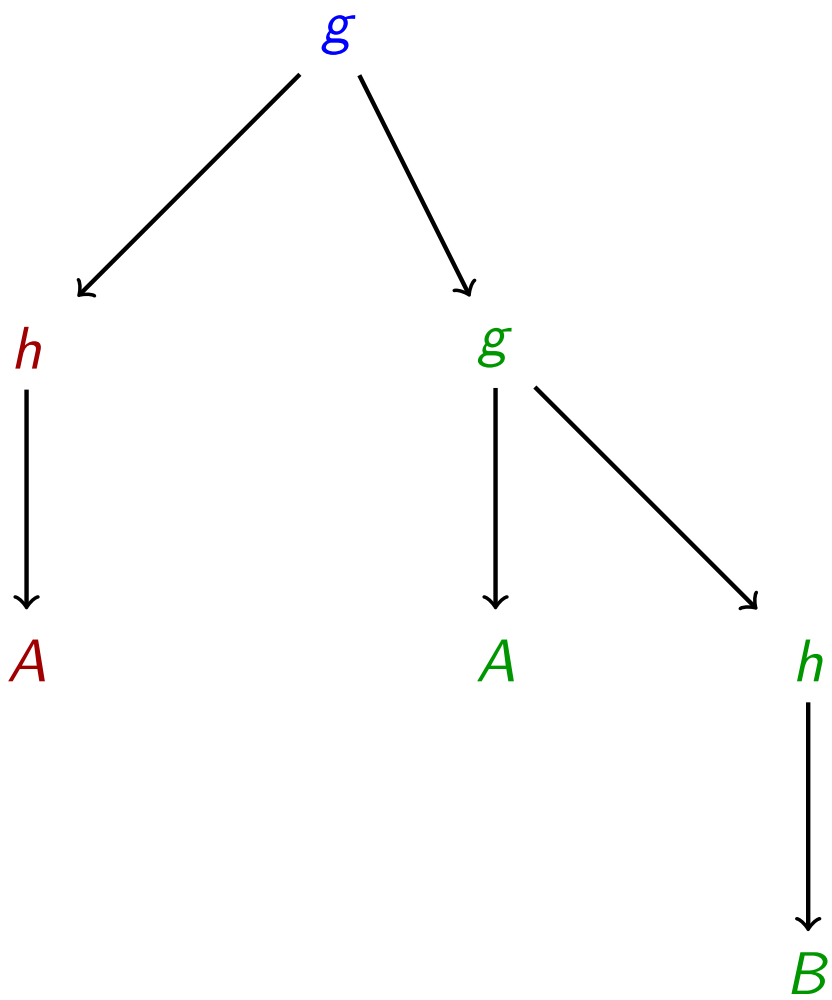
$\mathrm{pos}(t)$ est constitué de

- ε, et
- $1 \cdot \mathrm{pos}(A)$, et
  $1 \cdot \mathrm{pos}(A) = 1 \cdot \{\varepsilon\} = \{1\}$
- $2 \cdot \mathrm{pos}(g(B, h(A, B))$ qui est l'union de
  - $2 \cdot \{\varepsilon\} = \{2\}$ et
  - $2 \cdot 1 \cdot \mathrm{pos}(B) = \{2 \cdot 1\}$ et
  - $2 \cdot 2 \cdot \mathrm{pos}(h(A, B))$ qui vaut $\{2 \cdot 2, 2 \cdot 2 \cdot 1, 2 \cdot 2 \cdot 2\}$

$\mathrm{pos}(t) = \{\varepsilon, 1, 2, 2 \cdot 1, 2 \cdot 2, 2 \cdot 2 \cdot 1, 2 \cdot 2 \cdot 2\}$.

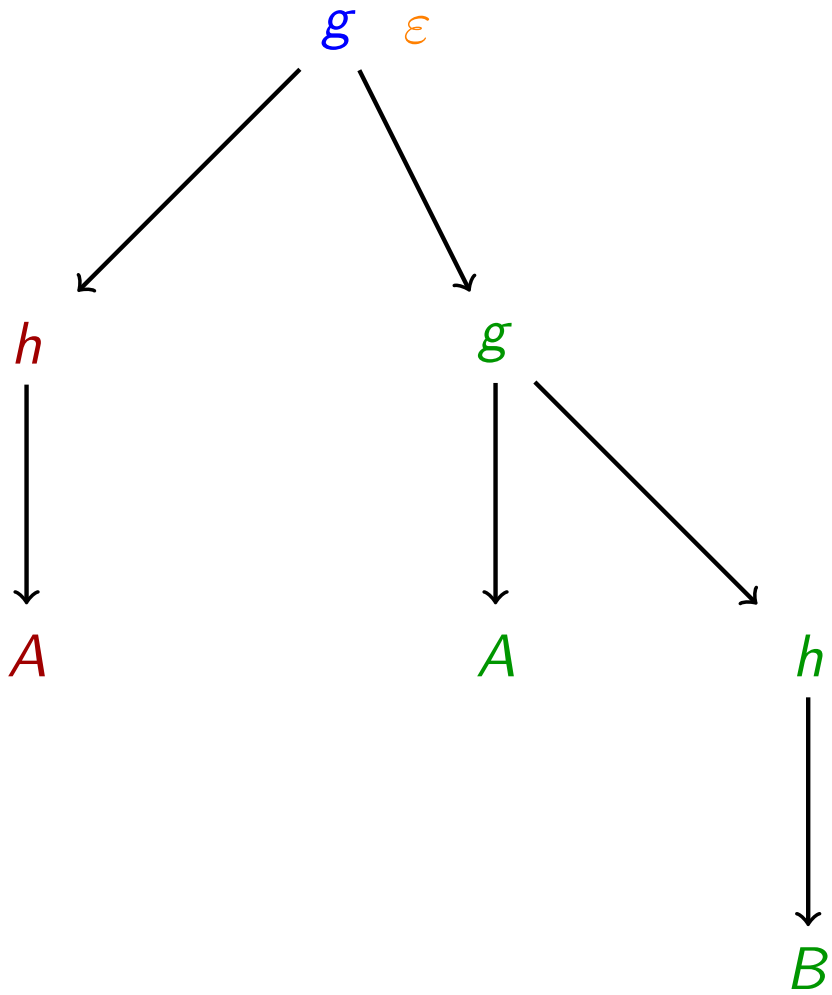
Positions et arbres

$t = g(h(A), g(A, h(B))) =$



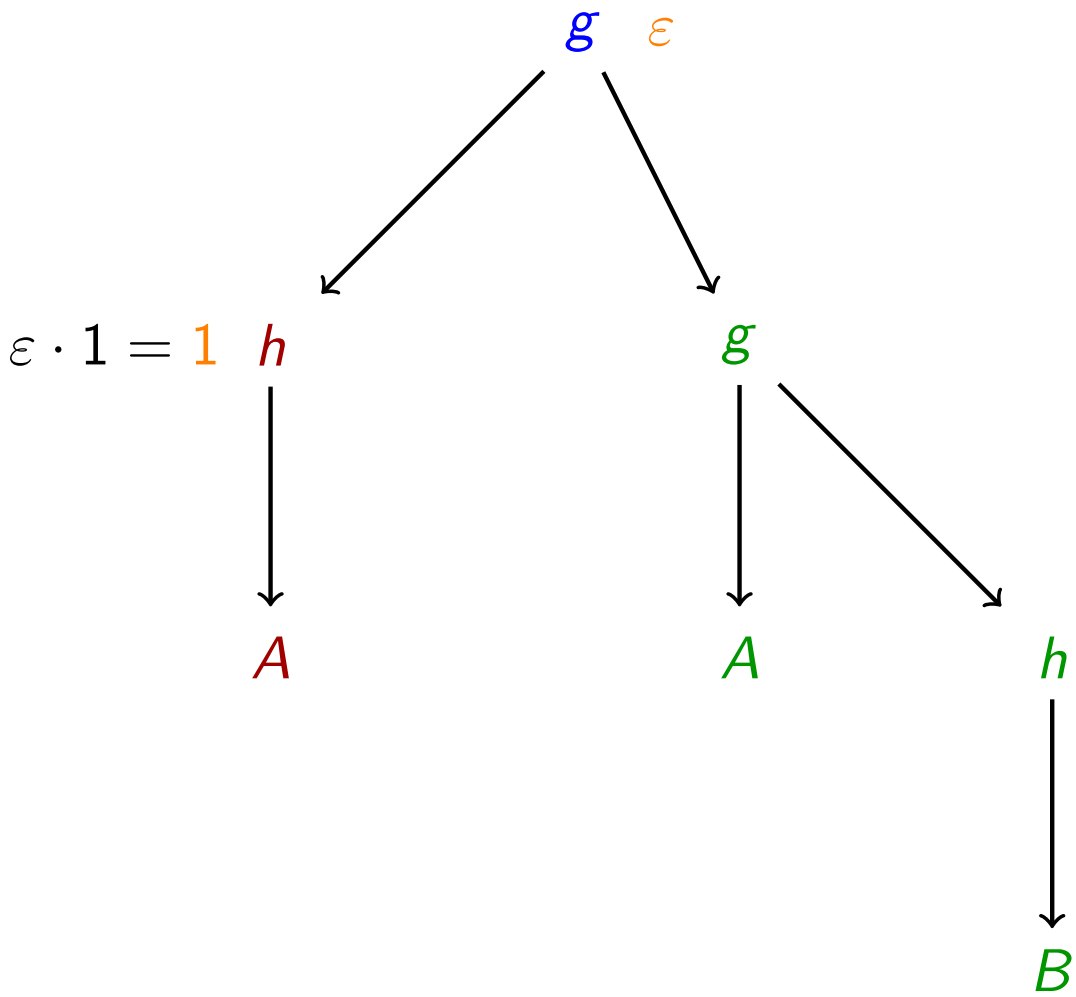
Positions et arbres

$t = g(h(A), g(A, h(B))) =$



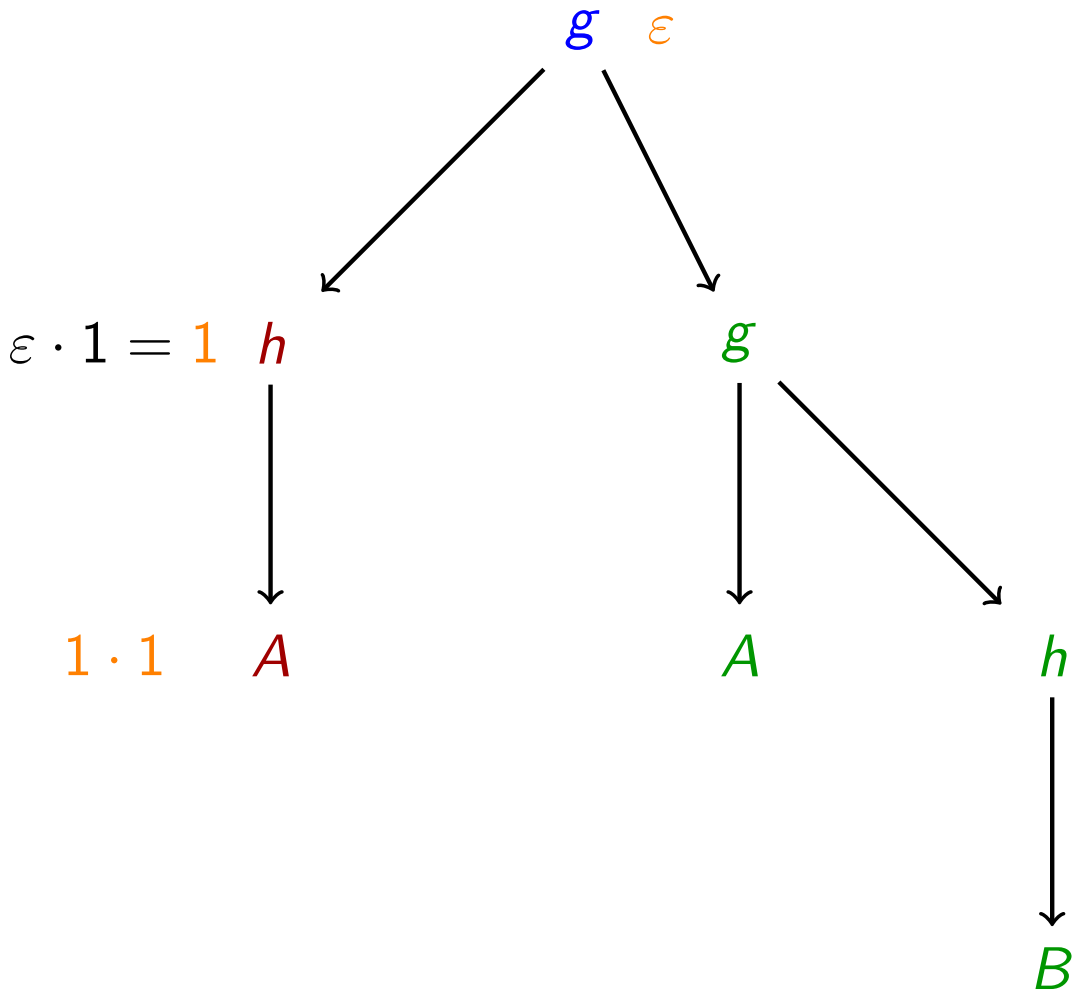
Positions et arbres

$t = g(h(A), g(A, h(B))) =$



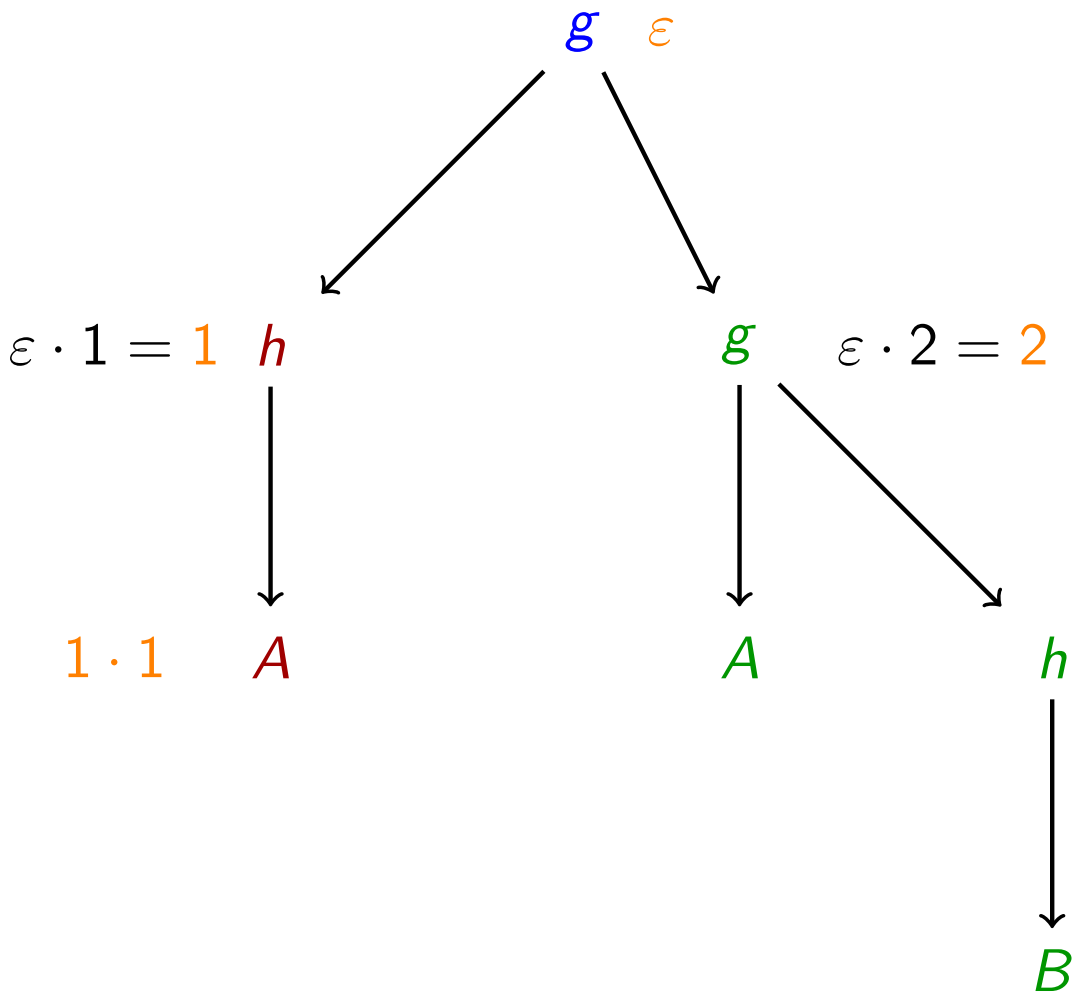
Positions et arbres

$t = g(h(A), g(A, h(B))) =$



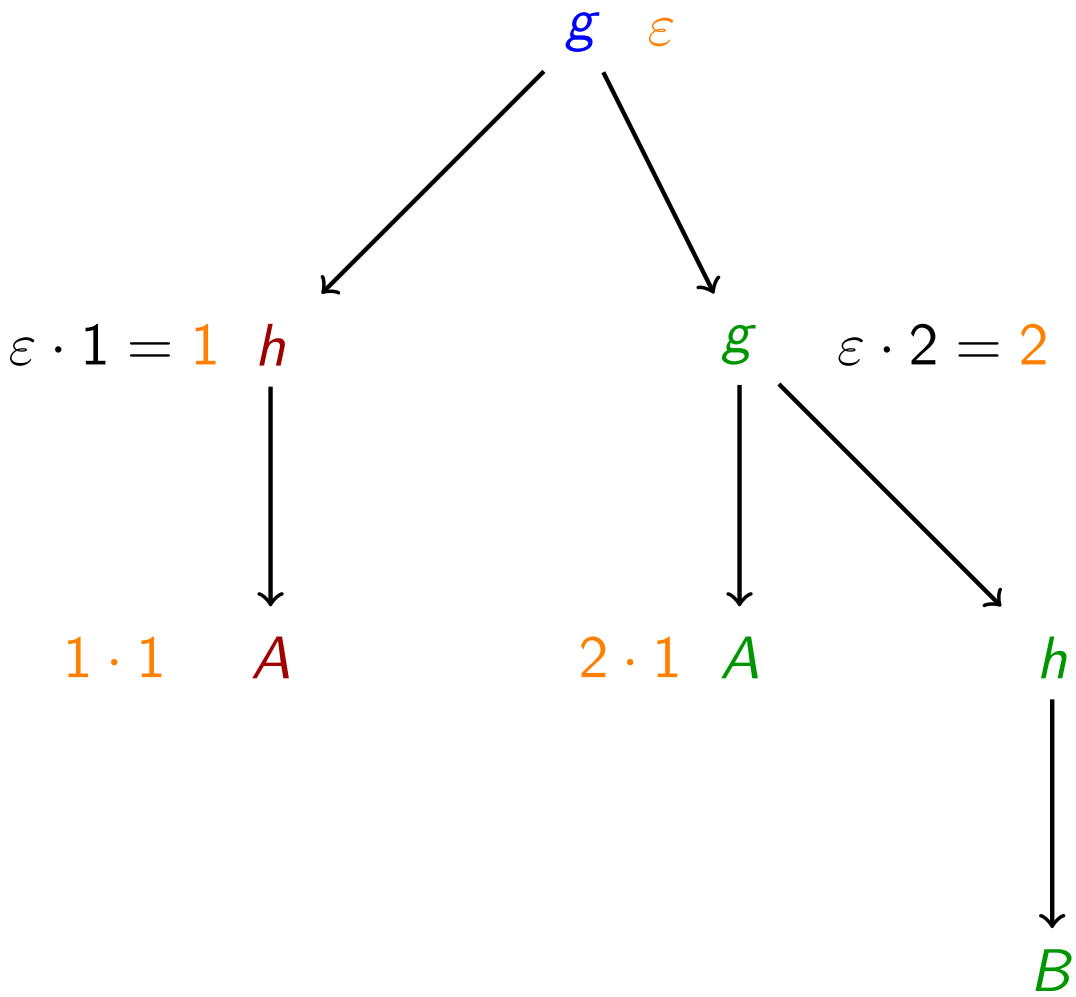
Positions et arbres

$t = g(h(A), g(A, h(B))) =$



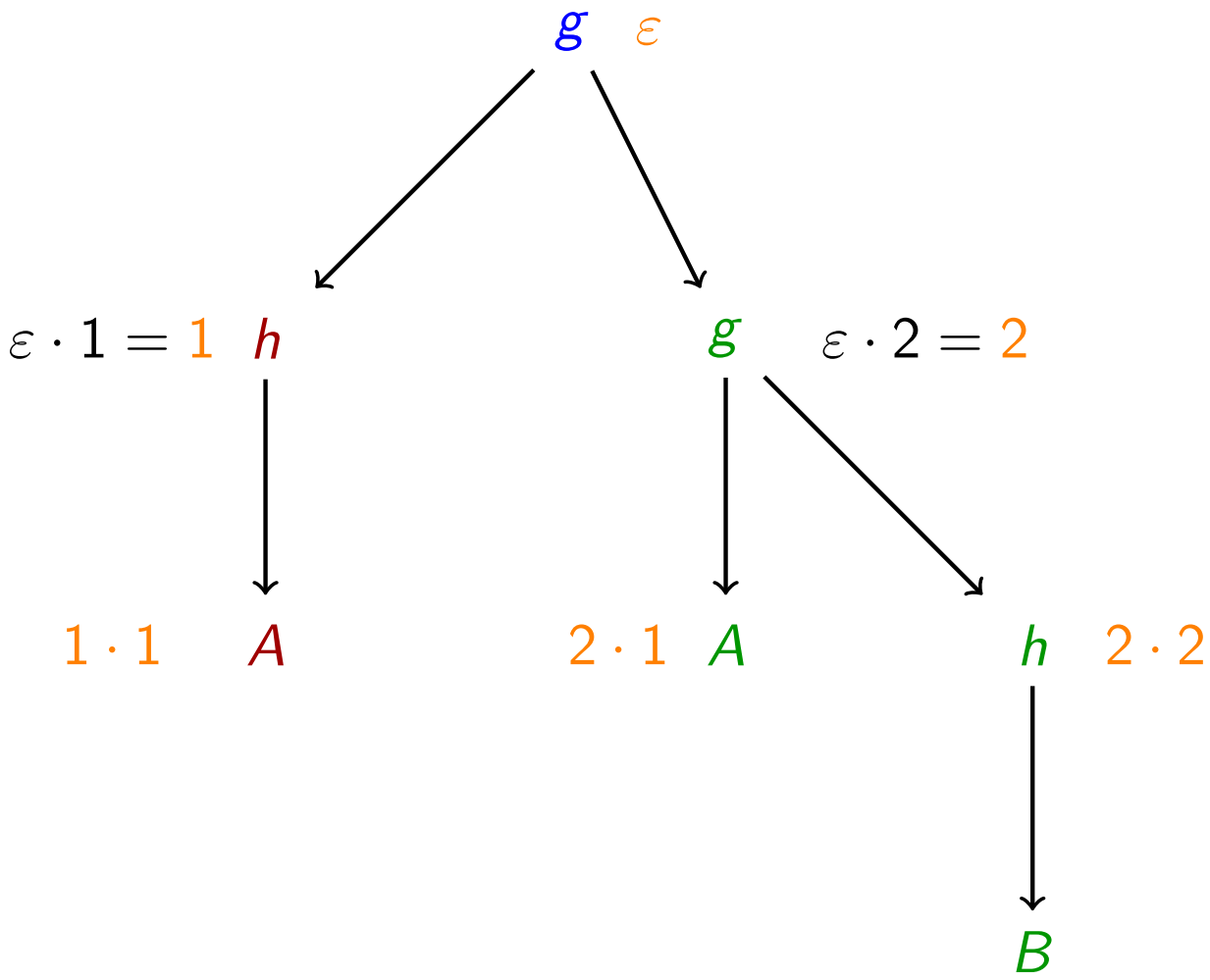
Positions et arbres

$t = g(h(A), g(A, h(B))) =$



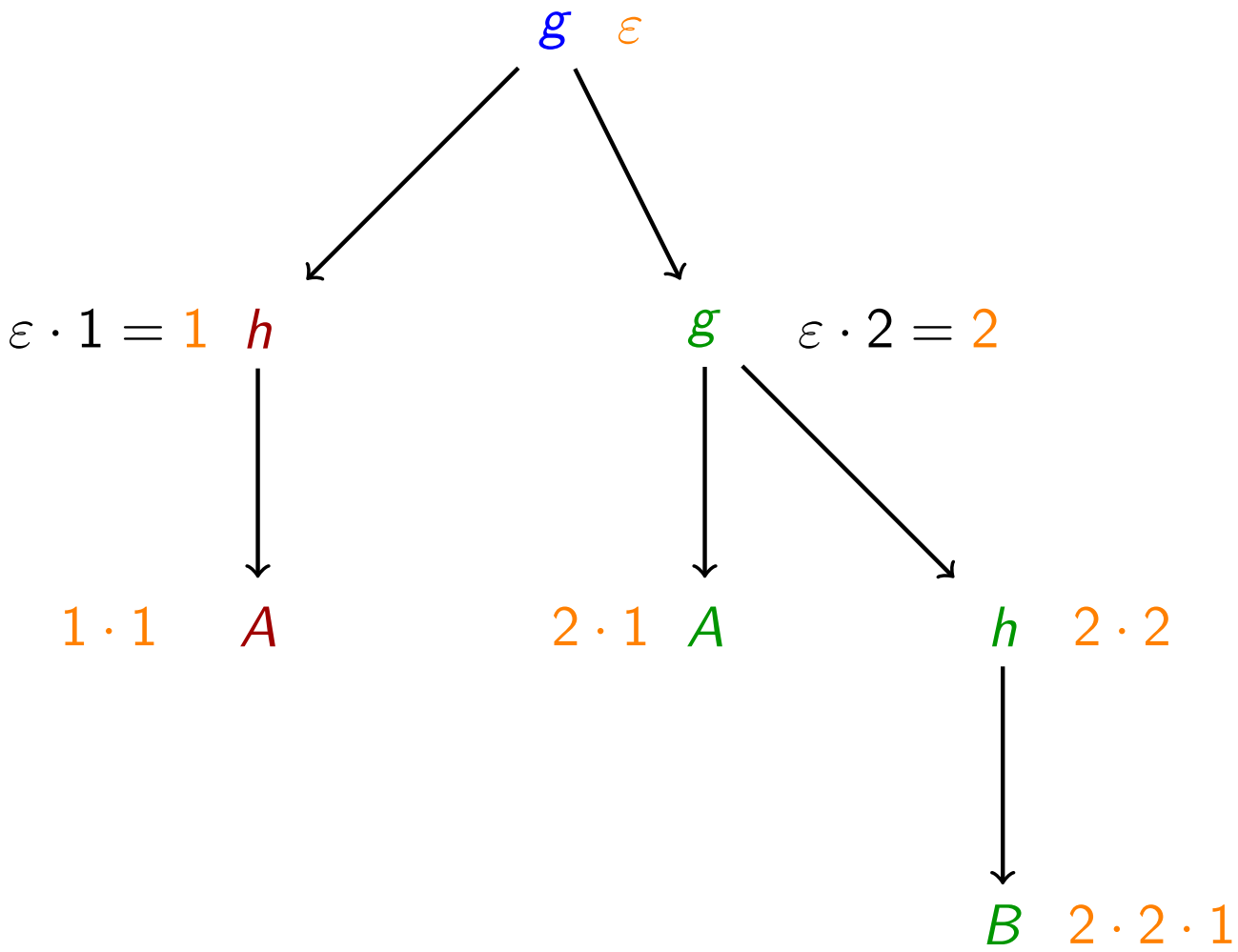
Positions et arbres

$t = g(h(A), g(A, h(B))) =$



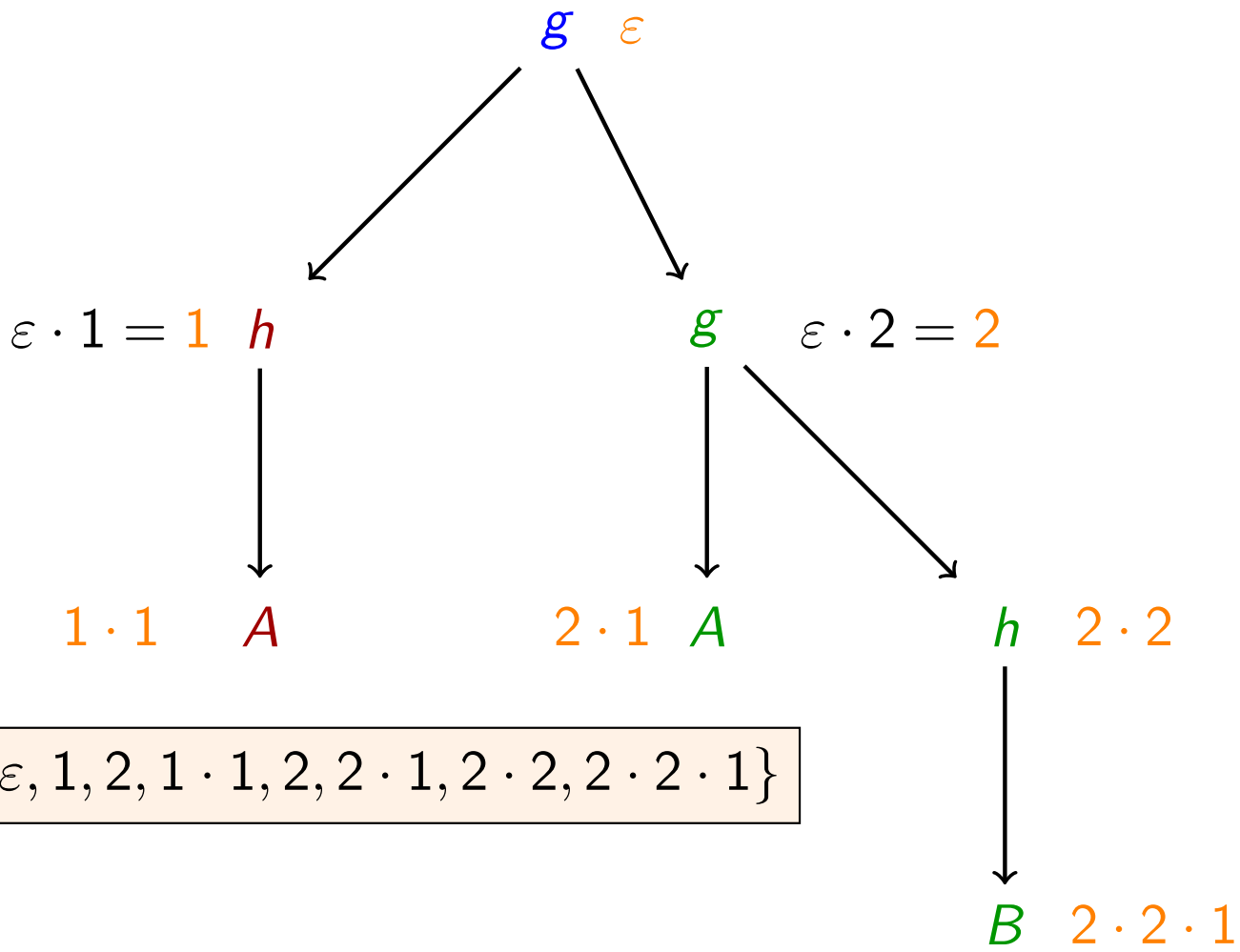
Positions et arbres

$t = g(h(A), g(A, h(B))) =$



Positions et arbres

$t = g(h(A), g(A, h(B))) =$



$$\text{pos}(t) = \{\varepsilon, 1, 2, 1 \cdot 1, 2, 2 \cdot 1, 2 \cdot 2, 2 \cdot 2 \cdot 1\}$$

Hauteur d'un terme

Informellement, la hauteur d'un terme est le nombre d'étage de l'arbre associée.

Définition

La hauteur d'un terme t est définie par :

- Si $t = A$ est une constante ou une variable, alors sa hauteur est 1.
- Si $t = f(t_1, \ldots, t_k)$, alors la hauteur de t est un plus le **maximum** des hauteurs des t_i.

La hauteur d'un terme est aussi un plus la longueur maximale d'une position de ce terme.

Variables d'un terme

L'ensemble des variables d'un termes est l'ensemble des variables qui y apparaissent. Plus formellement :

- Si $t = A$ est une constante, alors $\mathrm{var}(t) = \emptyset$,
- Si $t = x$ est une variable, alors $\mathrm{var}(t) = \{x\}$,
- Si $t = f(t_1, \ldots, t_k)$, alors

$$\mathrm{var}(t) = \mathrm{var}(t_1) \cup \ldots \cup \mathrm{var}(t_k).$$

Variables d'un terme

L'ensemble des variables d'un termes est l'ensemble des variables qui y apparaissent. Plus formellement :

- Si $t = A$ est une constante, alors $\mathrm{var}(t) = \emptyset$,
- Si $t = x$ est une variable, alors $\mathrm{var}(t) = \{x\}$,
- Si $t = f(t_1, \ldots, t_k)$, alors

$$\mathrm{var}(t) = \mathrm{var}(t_1) \cup \ldots \cup \mathrm{var}(t_k).$$

Exemple :

$$\begin{aligned}
\mathrm{var}(f(f(x, A), f(y, x))) &= \mathrm{var}(f(x, A)) \cup \mathrm{var}(f(y, x)) \\
&= \mathrm{var}(x)) \cup \mathrm{var}(A)\mathrm{var}(y) \cup \mathrm{var}(x) \\
&= \{x\} \cup \emptyset \cup \{y\} \cup \{x\} \\
&= \{x, y\}
\end{aligned}$$

Exercices

On considère le terme $f(f(A, h(f(x, A))), h(f(h(y), f(A, x))))$.

- Quel est l'ensemble des positions ?
- Quelle est sa hauteur ?
- Quel est son ensemble de variables ?

Exercices

On considère le terme $f(f(A, h(f(x, A))), h(f(h(y), f(A, x))))$.

- Quel est l'ensemble des positions ?
  $\{\varepsilon,\ 1,\ 1\cdot 1,\ 1\cdot 2,\ 1\cdot 2\cdot 1,\ 1\cdot 2\cdot 1\cdot 1,\ 1\cdot 2\cdot 1\cdot 2,\ 2, 2\cdot 1,\ 2\cdot 1\cdot 1,\ 2\cdot 1\cdot 1\cdot 1,\ 2\cdot 1\cdot 2,\ 2\cdot 1\cdot 2\cdot 1,\ 2\cdot 1\cdot 1\}$
- Quelle est sa hauteur ?
  5
- Quel est son ensemble de variables ?
  $\{x, y\}$

Sous-termes

Soit t un terme et p une position de t. Le sous-terme de t en position p, noté $t_{|p}$ est défini par :

- $t_{|\varepsilon} = t$
- Si $t = f(t_1, \ldots, t_k)$, et $p = i \cdot p'$, avec $i \in \{1, \ldots, k\}$, alors

$$t_{|i \cdot p'} = t_{i|p'}$$

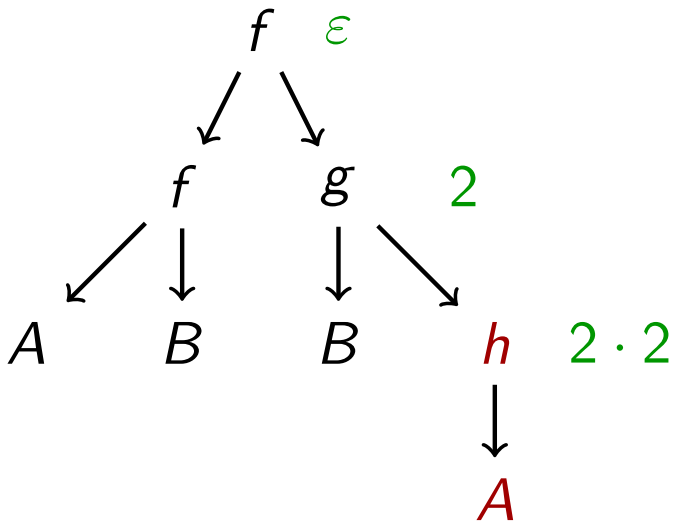
Intuition

Informellement, $t_{|p}$ est le terme que l'on obtient à partit de t en ne gardant que ce qui est sous la position p.

Sous-terme - exemple

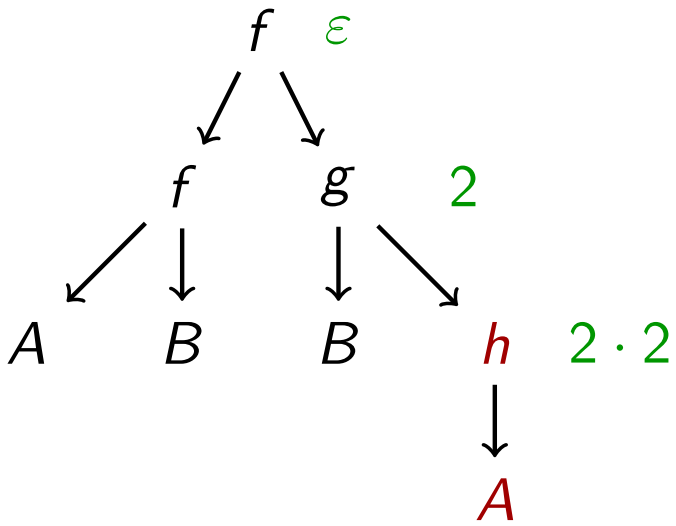
$t = f(f(A, B), g(B, h(A)))$

$t_{|2\cdot 2} = f(f(A, B), g(B, h(h(A))))_{|2\cdot 2}$



Sous-terme - exemple

$t = f(f(A, B), g(B, h(A)))$

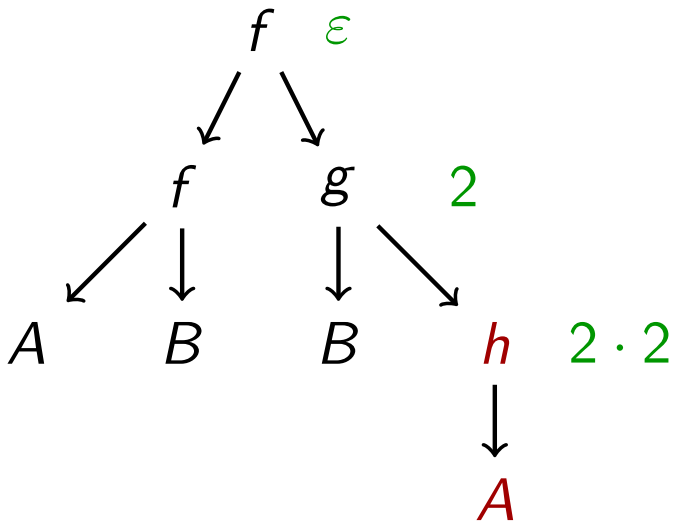


$t_{|2\cdot 2} = f(f(A, B), g(B, h(h(A))))_{|2\cdot 2}$

$= g(B, h(h(A)))_{|2}$

Sous-terme - exemple

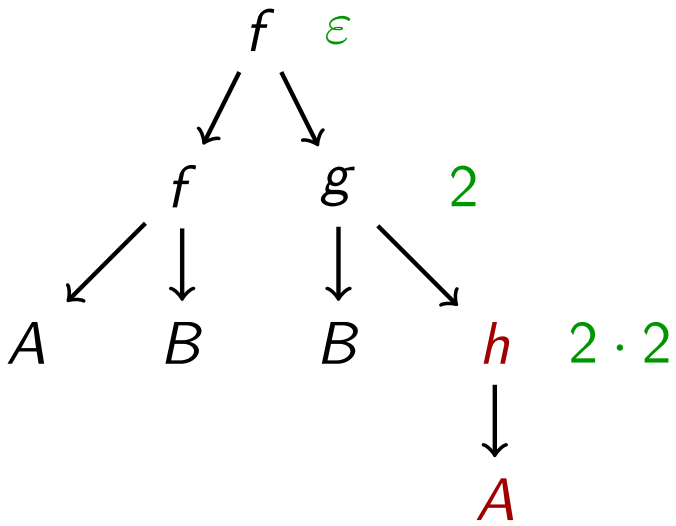
$t = f(f(A, B), g(B, h(A)))$



$$t_{|2\cdot 2} = f(f(A, B), g(B, h(h(A))))_{|2\cdot 2}$$
$$= g(B, h(h(A)))_{|2}$$
$$= h(h(A))_{|\varepsilon}$$

Sous-terme - exemple

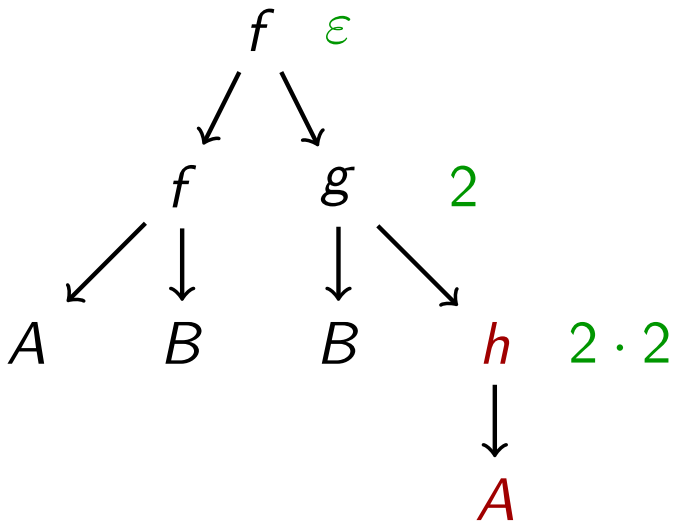
$t = f(f(A, B), g(B, h(A)))$



$$
\begin{aligned}
t_{|2\cdot 2} &= f(f(A, B), g(B, h(h(A))))_{|2\cdot 2} \\
&= g(B, h(h(A)))_{|2} \\
&= h(h(A))_{|\varepsilon} \\
&= h(h(A))
\end{aligned}
$$

Sous-terme - exemple

$t = f(f(A, B), g(B, h(A)))$

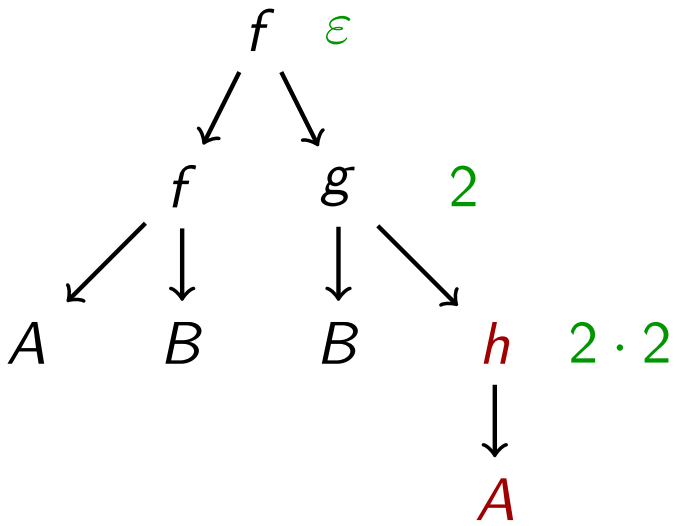


$$\begin{aligned} t_{|2\cdot 2} &= f(f(A, B), g(B, h(h(A))))_{|2\cdot 2} \\ &= g(B, h(h(A)))_{|2} \\ &= h(h(A))_{|\varepsilon} \\ &= h(h(A)) \end{aligned}$$

$t_{|1\cdot 2} = f(f(A, B), g(B, h(h(A))))_{|1\cdot 2}$

Sous-terme - exemple

$t = f(f(A, B), g(B, h(A)))$

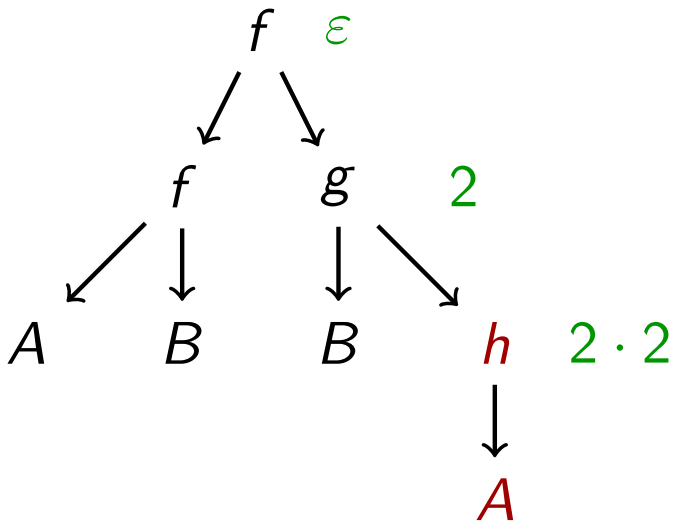


$$t_{|2\cdot 2} = f(f(A, B), g(B, h(h(A))))_{|2\cdot 2}$$
$$= g(B, h(h(A)))_{|2}$$
$$= h(h(A))_{|\varepsilon}$$
$$= h(h(A))$$

$$t_{|1\cdot 2} = f(f(A, B), g(B, h(h(A))))_{|1\cdot 2}$$
$$= f(A, B)_{|2}$$

Sous-terme - exemple

$t = f(f(A, B), g(B, h(A)))$

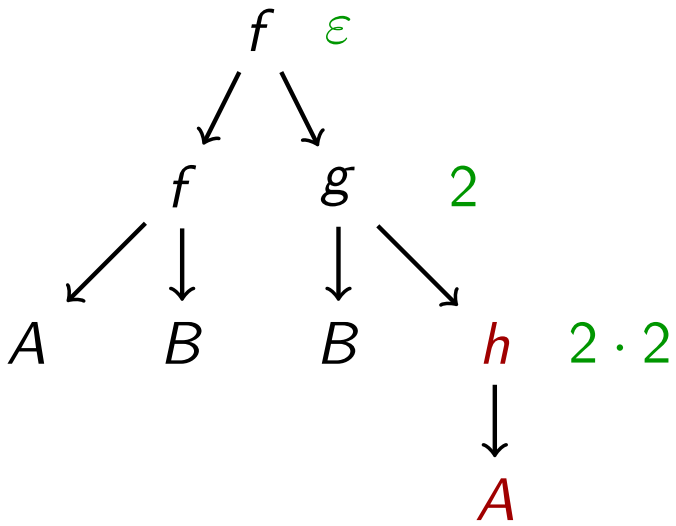


$$t_{|2\cdot 2} = f(f(A, B), g(B, h(h(A))))_{|2\cdot 2}$$
$$= g(B, h(h(A)))_{|2}$$
$$= h(h(A))_{|\varepsilon}$$
$$= h(h(A))$$

$$t_{|1\cdot 2} = f(f(A, B), g(B, h(h(A))))_{|1\cdot 2}$$
$$= f(A, B)_{|2}$$
$$= B_{|\varepsilon}$$

Sous-terme - exemple

$t = f(f(A, B), g(B, h(A)))$

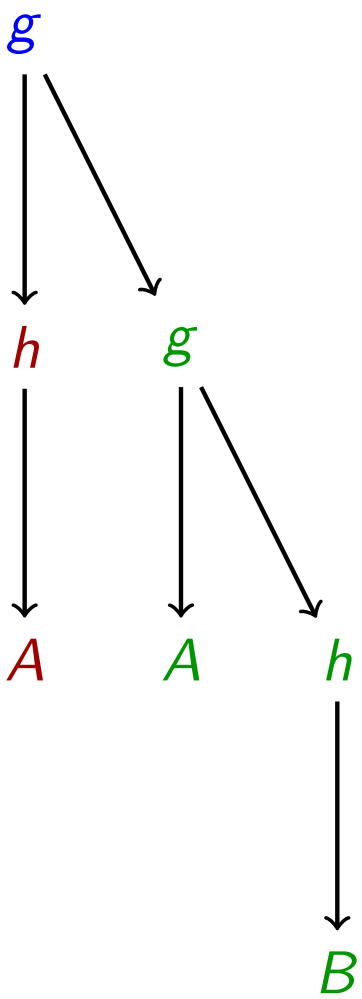


$$t_{|2\cdot 2} = f(f(A, B), g(B, h(h(A))))_{|2\cdot 2}$$
$$= g(B, h(h(A)))_{|2}$$
$$= h(h(A))_{|\varepsilon}$$
$$= h(h(A))$$

$$t_{|1\cdot 2} = f(f(A, B), g(B, h(h(A))))_{|1\cdot 2}$$
$$= f(A, B)_{|2}$$
$$= B_{|\varepsilon}$$
$$= B$$

Plan

Comment coder en machine ?



- En utilisant des chaînes de caractères ? (listes)

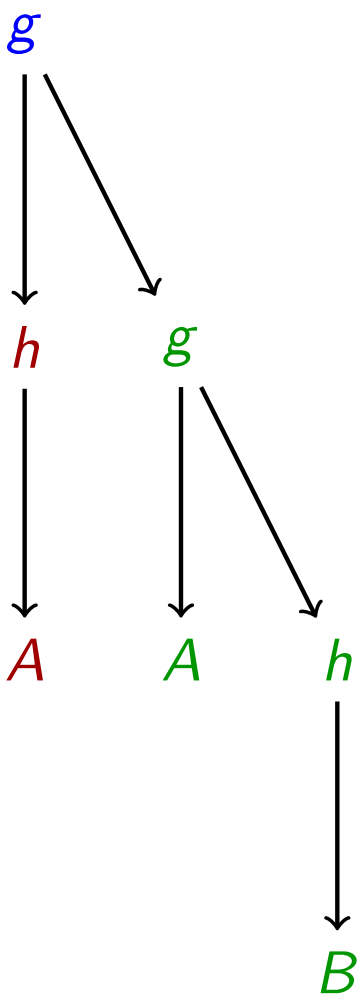
```
terme="g(h(A),g(A,h(B))))"
terme="AhABhgg" #en polonaise inverse
```

- Un tableau associatif/dictionnaire ?

```
terme={0:'g',1:'h',11:'A',2:'g',21:'A',etc.}
```

Quel est le sous arbre droit à la racine ? Quelle est la hauteur ?

Comment coder en machine ?



- En utilisant des chaînes de caractères ? (listes)

```
terme="g(h(A),g(A,h(B))))"
terme="AhABhgg" #en polonaise inverse
```

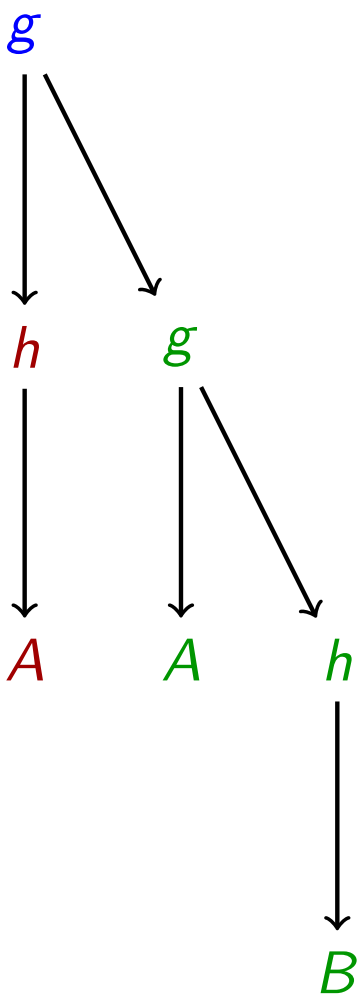
peu adapté au traitement algorithmique

- Un tableau associatif/dictionnaire ?

```
terme={0:'g',1:'h',11:'A',2:'g',21:'A',etc.}
```

Quel est le sous arbre droit à la racine ? Quelle est la hauteur ?

Comment coder en machine ?



- En utilisant des chaînes de caractères ? (listes)

```
terme="g(h(A),g(A,h(B))))"
terme="AhABhgg" #en polonaise inverse
```

peu adapté au traitement algorithmique

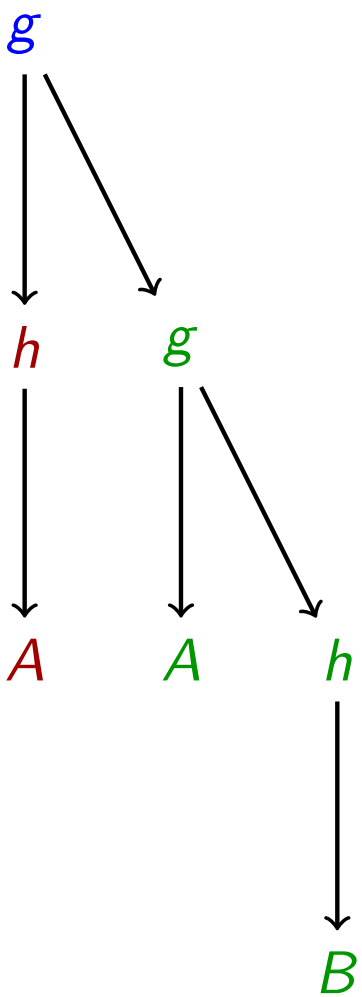
- Un tableau associatif/dictionnaire ?

```
terme={0:'g',1:'h',11:'A',2:'g',21:'A',etc.}
```

mieux, mais toujours insuffisant

Quel est le sous arbre droit à la racine ? Quelle est la hauteur ?

Comment coder en machine ?



- En utilisant des chaînes de caractères ? (listes)

```
terme="g(h(A),g(A,h(B))))"
terme="AhABhgg" #en polonaise inverse
```

peu adapté au traitement algorithmique

- Un tableau associatif/dictionnaire ?

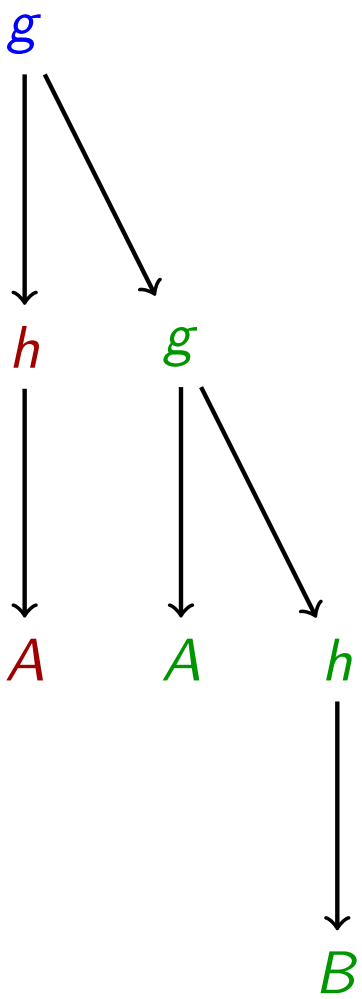
```
terme={0:'g',1:'h',11:'A',2:'g',21:'A',etc.}
```

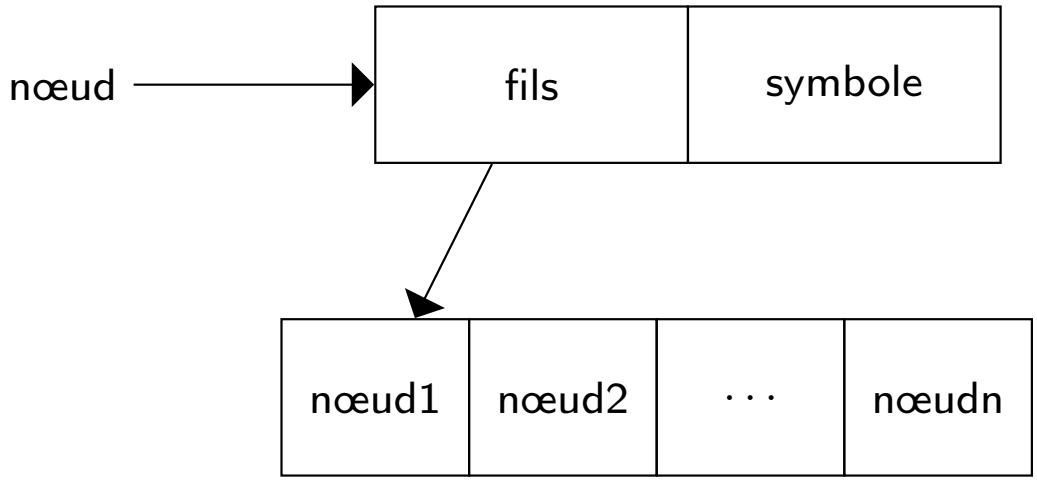
mieux, mais toujours insuffisant

Quel est le sous arbre droit à la racine ? Quelle est la hauteur ?

Utiliser un structure arborescente

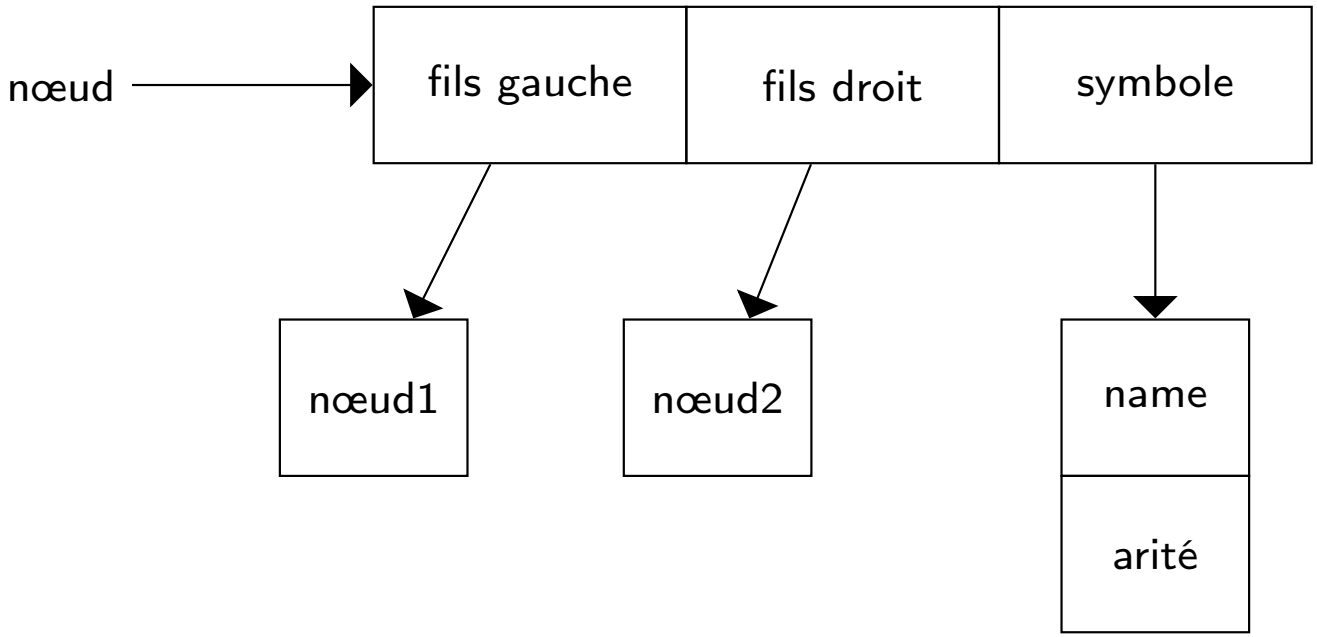
Codage général





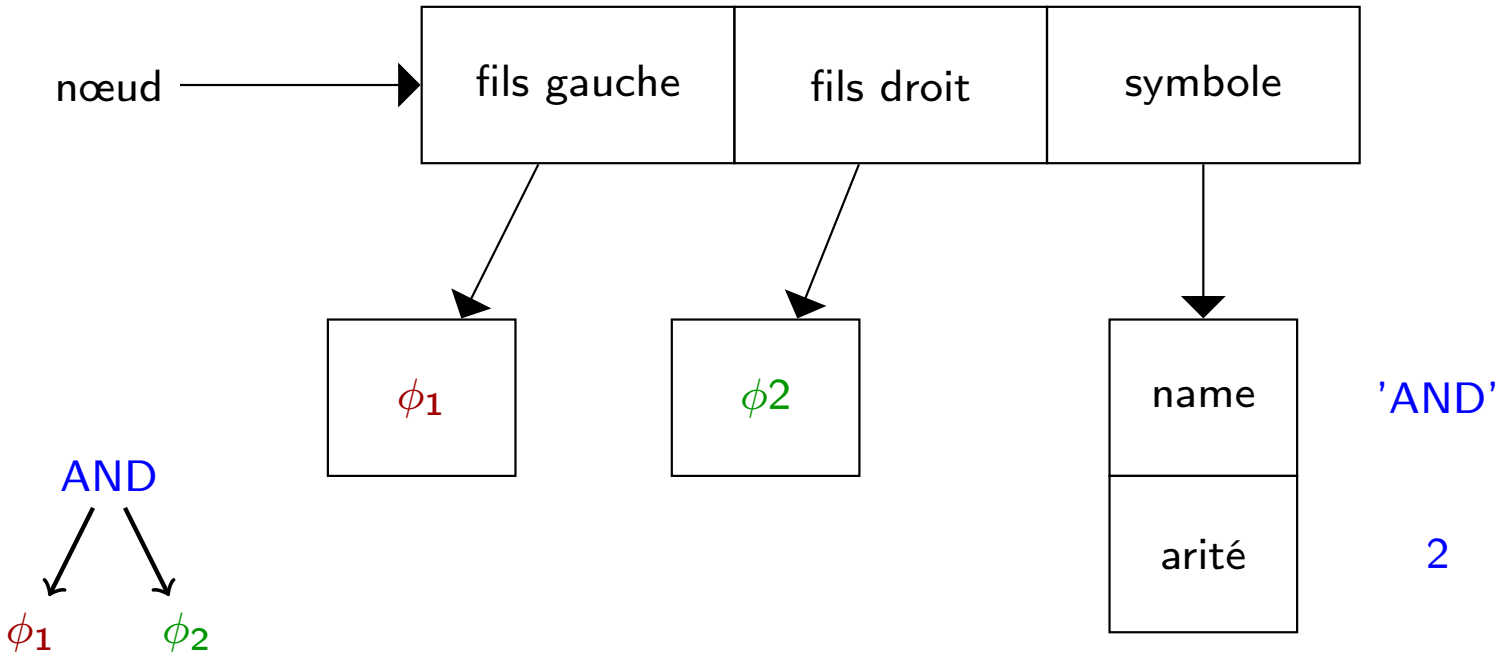
Exemples pour les formules logiques

On utilise des arbres binaires : tout nœud a au plus deux fils, dénotés *fils gauche* et *fils droit*.



Exemples pour les formules logiques

On utilise des arbres binaires : tout nœud a au plus deux fils, dénotés *fils gauche* et *fils droit*.



Les symboles

```
class Symbol:
    def __init__(self,arity,name):
        self.arity=arity
        self.name=name

    def OR():
        return Symbol(2,"OR")
    def AND():
        return Symbol(2,"AND")
    def IMP():
        return Symbol(2,"IMP")
    def EQU():
        return Symbol(2,"EQU")
    def T():
        return Symbol(0,"TRUE")
    def F():
        return Symbol(0,"FALSE")
    def NOT():
        return Symbol(1,"NOT")
    def VAR(i):
        return Symbol(0,"X"+str(i))
```

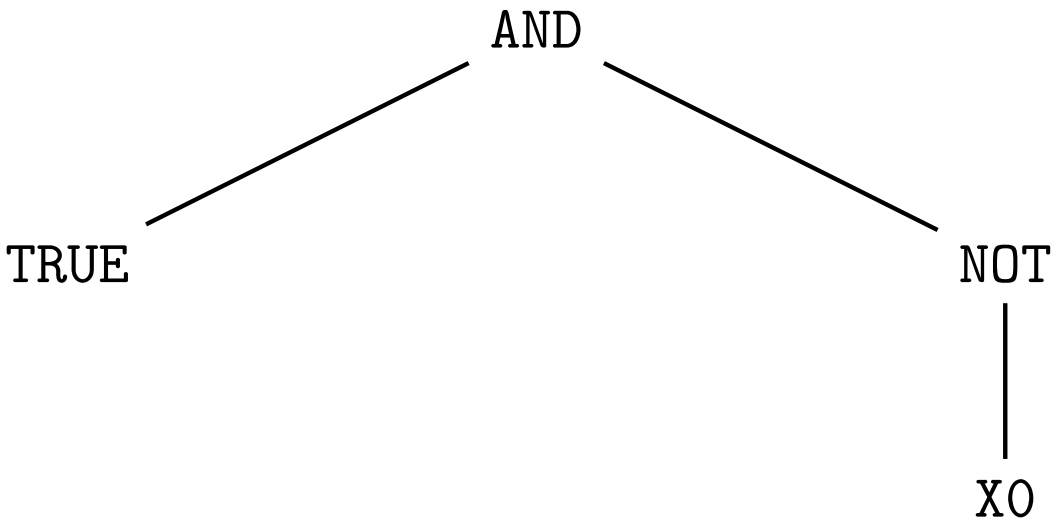
Les nœuds

```
class Node:
    def __init__(self,S):
        self.left=None #not used if TRUE/FALSE/Variable
        self.right=None #not used if unary
        self.value=S
```

```
Myformule=Node(Symbol.OR())
Myformule.left=Node(Symbol.T())
Myformule.right=Node(Symbol.F())
print(Myformule)
```

```
OR(TRUE,FALSE)
```

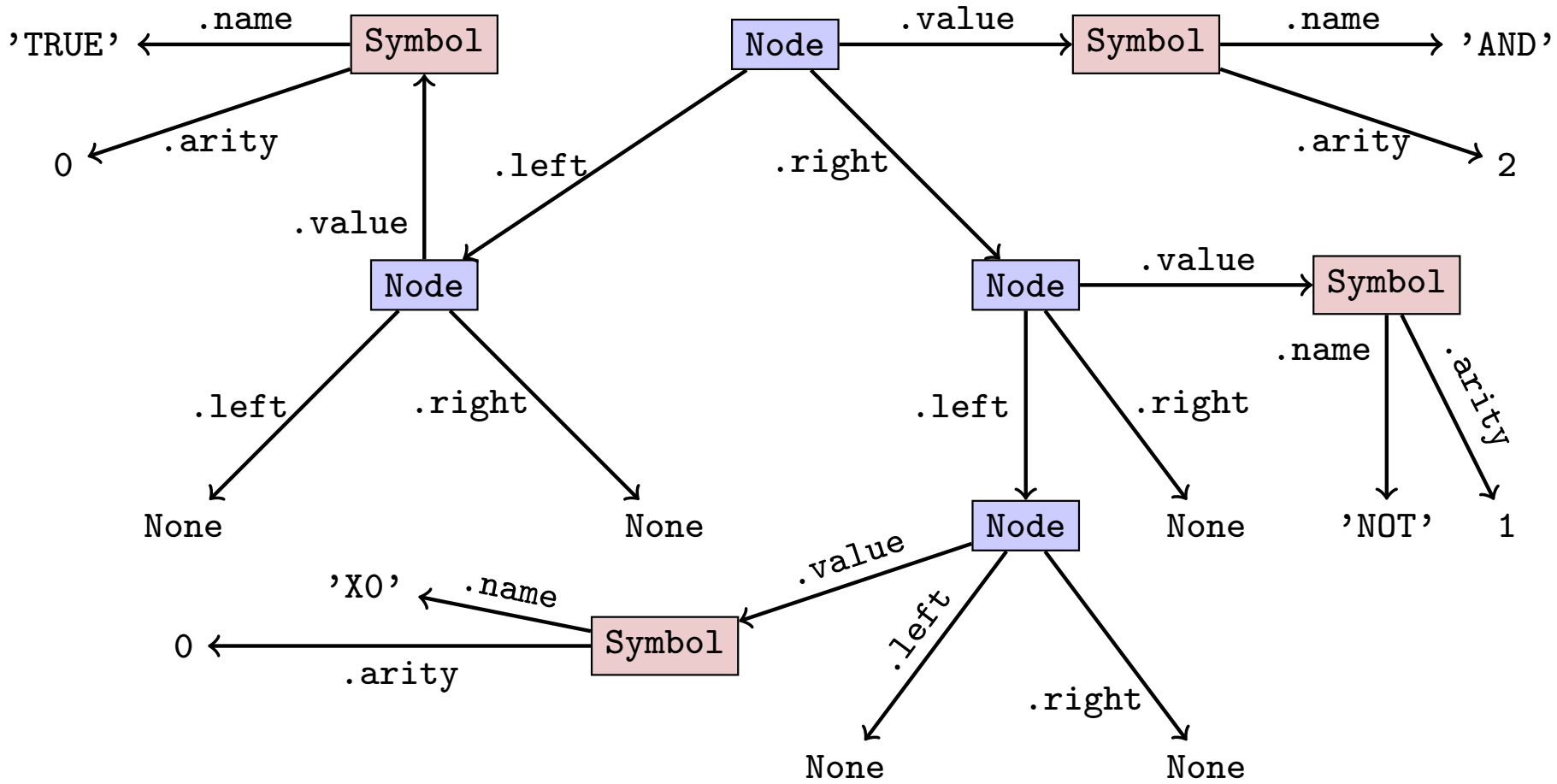
Exemple




```
exampleForm=Node(Symbol.AND()) #creation de la racine
print(exampleForm)
exampleForm.left=Node(Symbol.T()) # creation du fils gauche
exampleForm.right=Node(Symbol.NOT()) # creation du fils droit
print(exampleForm)
exampleForm.right.left=Node(Symbol.VAR(0)) # dernier noeud
print(exampleForm)
```

```
AND(None,None)
AND(TRUE,NOT(None))
AND(TRUE,NOT(X0))
```

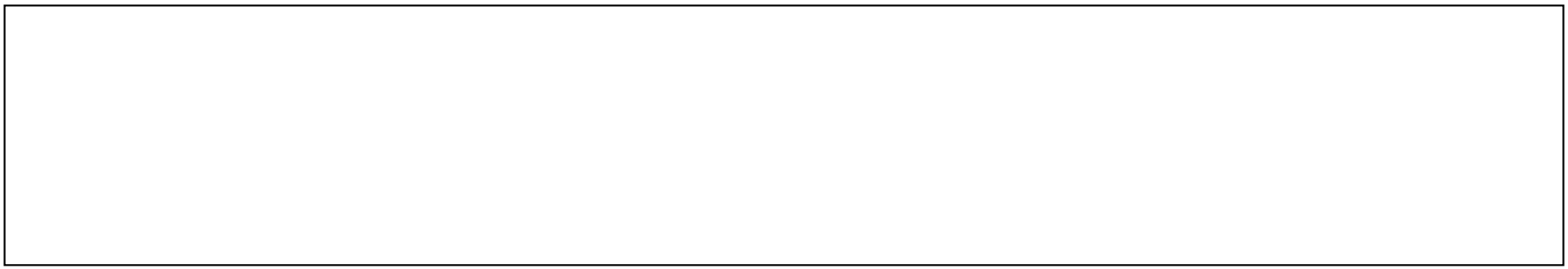
Exemple : codage de AND(TRUE,NOT(X0))



Exemple d'algorithme

Exercice

Écrire un algorithme pour tester si une formule a `'AND'` comme symbole à la racine.



Exemple d'algorithme

Exercice

Écrire un algorithme pour tester si une formule a 'AND' comme symbole à la racine.

```
def ANDracine(formule):
    if formule.value.name=='AND' :
        return True
    else :
        return False
```

Exemple d'algorithme

Exercice

Écrire un algorithme pour tester si une formule contient au moins un `'AND'`.

Exemple d'algorithme

Exercice

Écrire un algorithme pour tester si une formule contient au moins un 'AND'.

```
def ANDpresent(formule):
    if formule == None:
        return False
    if formule.value.name == 'AND':
        return True
    if formule.value.arity == 0:
        return False
    if formule.value.arity == 1:
        return ANDpresent(formule.left)
    if formule.value.arity == 2:
        return ANDpresent(formule.left) or
                          ANDpresent(formule.right)

myf=Node.fromString('OR(NOT(TRUE),NOT(NOT(FALSE)))')
print (ANDpresent(myf))
myf2=Node.fromString('OR(NOT(TRUE),NOT(NOT(AND(FALSE,TRUE))))')
print (ANDpresent(myf2))
```

```
False
True
```

Plan

Exemples

La réécriture est un modèle d'automatisation du calcul particulièrement adapté au traitement sur les arbres et donc en logique.

Exemple

Toute formule de logique propositionnelle est équivalente à un formule propositionnelle n'utilisant que des $\vee$, $\wedge$ et des négations uniquement sur les variables.

Preuve :

On applique les équivalences suivantes :

$\neg\neg\varphi \sim \varphi$

$\neg(\varphi_1 \vee \varphi_2) \sim \neg\varphi_1 \wedge \neg\varphi_2$

$\neg(\varphi_1 \wedge \varphi_2) \sim \neg\varphi_1 \vee \neg\varphi_2$

$\varphi_1 \Rightarrow \varphi_2 \sim \neg\varphi_1 \vee \varphi_2$

$\varphi_1 \Leftrightarrow \varphi_2 \sim (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$

$\neg\top \sim \bot$

$\neg\bot \sim \top$

Exemple

Exemple

$\neg\neg\varphi \sim \varphi$

$\neg(\varphi_1 \vee \varphi_2) \sim \neg\varphi_1 \wedge \neg\varphi_2$

$\neg(\varphi_1 \wedge \varphi_2) \sim \neg\varphi_1 \vee \neg\varphi_2$

$\varphi_1 \Rightarrow \varphi_2 \sim \neg\varphi_1 \vee \varphi_2$

$\varphi_1 \Leftrightarrow \varphi_1 \sim (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$

$$(A \wedge \neg B) \Rightarrow (\neg C \Leftrightarrow A)$$

Exemple

Exemple

$\neg\neg\varphi \sim \varphi$

$\neg(\varphi_1 \vee \varphi_2) \sim \neg\varphi_1 \wedge \neg\varphi_2$

$\neg(\varphi_1 \wedge \varphi_2) \sim \neg\varphi_1 \vee \neg\varphi_2$

$\varphi_1 \Rightarrow \varphi_2 \sim \neg\varphi_1 \vee \varphi_2$

$\varphi_1 \Leftrightarrow \varphi_2 \sim (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$

$$(A \wedge \neg B) \Rightarrow (\neg C \Leftrightarrow A)$$

Exemple

Exemple

$\neg\neg\varphi \sim \varphi$

$\neg(\varphi_1 \vee \varphi_2) \sim \neg\varphi_1 \wedge \neg\varphi_2$

$\neg(\varphi_1 \wedge \varphi_2) \sim \neg\varphi_1 \vee \neg\varphi_2$

$\varphi_1 \Rightarrow \varphi_2 \sim \neg\varphi_1 \vee \varphi_2$

$\varphi_1 \Leftrightarrow \varphi_2 \sim (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$

$$(A \wedge \neg B) \Rightarrow (\neg C \Leftrightarrow A)$$
$$\sim (A \wedge \neg B) \Rightarrow ((\neg C \Rightarrow A) \wedge (A \Rightarrow \neg C))$$

Exemple

Exemple

$\neg\neg\varphi \sim \varphi$

$\neg(\varphi_1 \vee \varphi_2) \sim \neg\varphi_1 \wedge \neg\varphi_2$

$\neg(\varphi_1 \wedge \varphi_2) \sim \neg\varphi_1 \vee \neg\varphi_2$

$\varphi_1 \Rightarrow \varphi_2 \sim \neg\varphi_1 \vee \varphi_2$

$\varphi_1 \Leftrightarrow \varphi_1 \sim (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$

$$(A \wedge \neg B) \Rightarrow (\neg C \Leftrightarrow A)$$
$$\sim (A \wedge \neg B) \Rightarrow ((\neg C \Rightarrow A) \wedge (A \Rightarrow \neg C))$$

Exemple

Exemple

$\neg\neg\varphi \sim \varphi$

$\neg(\varphi_1 \vee \varphi_2) \sim \neg\varphi_1 \wedge \neg\varphi_2$

$\neg(\varphi_1 \wedge \varphi_2) \sim \neg\varphi_1 \vee \neg\varphi_2$

$\varphi_1 \Rightarrow \varphi_2 \sim \neg\varphi_1 \vee \varphi_2$

$\varphi_1 \Leftrightarrow \varphi_1 \sim (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$

$$
\begin{aligned}
& (A \wedge \neg B) \Rightarrow (\neg C \Leftrightarrow A) \\
\sim\ & (A \wedge \neg B) \Rightarrow ((\neg C \Rightarrow A) \wedge (A \Rightarrow \neg C)) \\
\sim\ & \neg(A \wedge \neg B) \vee ((\neg\neg C \vee A) \wedge (\neg A \vee \neg C))
\end{aligned}
$$

Exemple

Exemple

$\neg\neg\varphi \sim \varphi$

$\neg(\varphi_1 \vee \varphi_2) \sim \neg\varphi_1 \wedge \neg\varphi_2$

$\neg(\varphi_1 \wedge \varphi_2) \sim \neg\varphi_1 \vee \neg\varphi_2$

$\varphi_1 \Rightarrow \varphi_2 \sim \neg\varphi_1 \vee \varphi_2$

$\varphi_1 \Leftrightarrow \varphi_1 \sim (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$

$$
\begin{aligned}
& (A \wedge \neg B) \Rightarrow (\neg C \Leftrightarrow A) \\
\sim\ & (A \wedge \neg B) \Rightarrow ((\neg C \Rightarrow A) \wedge (A \Rightarrow \neg C)) \\
\sim\ & \neg(A \wedge \neg B) \vee ((\neg\neg C \vee A) \wedge (\neg A \vee \neg C))
\end{aligned}
$$

Exemple

Exemple

$\neg\neg\varphi \sim \varphi$

$\neg(\varphi_1 \vee \varphi_2) \sim \neg\varphi_1 \wedge \neg\varphi_2$

$\neg(\varphi_1 \wedge \varphi_2) \sim \neg\varphi_1 \vee \neg\varphi_2$

$\varphi_1 \Rightarrow \varphi_2 \sim \neg\varphi_1 \vee \varphi_2$

$\varphi_1 \Leftrightarrow \varphi_1 \sim (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$

$$
\begin{aligned}
& (A \wedge \neg B) \Rightarrow (\neg C \Leftrightarrow A) \\
\sim\ & (A \wedge \neg B) \Rightarrow ((\neg C \Rightarrow A) \wedge (A \Rightarrow \neg C)) \\
\sim\ & \neg(A \wedge \neg B) \vee ((\neg\neg C \vee A) \wedge (\neg A \vee \neg C)) \\
\sim\ & (\neg A \vee \neg\neg B) \vee ((\neg\neg C \vee A) \wedge (\neg A \vee \neg C))
\end{aligned}
$$

Exemple

Exemple

$\neg\neg\varphi \sim \varphi$

$\neg(\varphi_1 \vee \varphi_2) \sim \neg\varphi_1 \wedge \neg\varphi_2$

$\neg(\varphi_1 \wedge \varphi_2) \sim \neg\varphi_1 \vee \neg\varphi_2$

$\varphi_1 \Rightarrow \varphi_2 \sim \neg\varphi_1 \vee \varphi_2$

$\varphi_1 \Leftrightarrow \varphi_1 \sim (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$

$$
\begin{aligned}
& (A \wedge \neg B) \Rightarrow (\neg C \Leftrightarrow A) \\
\sim\ & (A \wedge \neg B) \Rightarrow ((\neg C \Rightarrow A) \wedge (A \Rightarrow \neg C)) \\
\sim\ & \neg(A \wedge \neg B) \vee ((\neg\neg C \vee A) \wedge (\neg A \vee \neg C)) \\
\sim\ & (\neg A \vee \neg\neg B) \vee ((\neg\neg C \vee A) \wedge (\neg A \vee \neg C))
\end{aligned}
$$

Exemple

Exemple

$\neg\neg\varphi \sim \varphi$

$\neg(\varphi_1 \vee \varphi_2) \sim \neg\varphi_1 \wedge \neg\varphi_2$

$\neg(\varphi_1 \wedge \varphi_2) \sim \neg\varphi_1 \vee \neg\varphi_2$

$\varphi_1 \Rightarrow \varphi_2 \sim \neg\varphi_1 \vee \varphi_2$

$\varphi_1 \Leftrightarrow \varphi_1 \sim (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$

$$
\begin{aligned}
& (A \wedge \neg B) \Rightarrow (\neg C \Leftrightarrow A) \\
\sim\ & (A \wedge \neg B) \Rightarrow ((\neg C \Rightarrow A) \wedge (A \Rightarrow \neg C)) \\
\sim\ & \neg(A \wedge \neg B) \vee ((\neg\neg C \vee A) \wedge (\neg A \vee \neg C)) \\
\sim\ & (\neg A \vee \neg\neg B) \vee ((\neg\neg C \vee A) \wedge (\neg A \vee \neg C)) \\
\sim\ & (\neg A \vee B) \vee ((C \vee A) \wedge (\neg A \vee \neg C))
\end{aligned}
$$

Exemple

Exemple

$\neg\neg\varphi \sim \varphi$

$\neg(\varphi_1 \vee \varphi_2) \sim \neg\varphi_1 \wedge \neg\varphi_2$

$\neg(\varphi_1 \wedge \varphi_2) \sim \neg\varphi_1 \vee \neg\varphi_2$

$\varphi_1 \Rightarrow \varphi_2 \sim \neg\varphi_1 \vee \varphi_2$

$\varphi_1 \Leftrightarrow \varphi_1 \sim (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$

$$
\begin{aligned}
& (A \wedge \neg B) \Rightarrow (\neg C \Leftrightarrow A) \\
\sim\ & (A \wedge \neg B) \Rightarrow ((\neg C \Rightarrow A) \wedge (A \Rightarrow \neg C)) \\
\sim\ & \neg(A \wedge \neg B) \vee ((\neg\neg C \vee A) \wedge (\neg A \vee \neg C)) \\
\sim\ & (\neg A \vee \neg\neg B) \vee ((\neg\neg C \vee A) \wedge (\neg A \vee \neg C)) \\
\sim\ & (\neg A \vee B) \vee ((C \vee A) \wedge (\neg A \vee \neg C))
\end{aligned}
$$

Exercice

Exercice

Soit $\varphi = ((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$. Donner une formule équivalente n'utilisant que des $\vee$, $\wedge$ et des négations uniquement sur les variables.

Exercice

Exercice

Soit $\varphi = ((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$. Donner une formule équivalente n'utilisant que des $\vee$, $\wedge$ et des négations uniquement sur les variables.

$$\varphi = ((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$$

Exercice

Exercice

Soit $\varphi = ((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$. Donner une formule équivalente n'utilisant que des $\vee$, $\wedge$ et des négations uniquement sur les variables.

$$\begin{aligned} \varphi &= ((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C \\ &\sim \neg((A \vee \neg B) \wedge (\neg C \vee A)) \vee C \end{aligned}$$

Exercice

Exercice

Soit $\varphi = ((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$. Donner une formule équivalente n'utilisant que des $\vee$, $\wedge$ et des négations uniquement sur les variables.

$$\begin{aligned}\varphi &= ((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C \\ &\sim \neg((A \vee \neg B) \wedge (\neg C \vee A)) \vee C \\ &\sim (\neg(A \vee \neg B) \vee \neg(\neg C \vee A)) \vee C\end{aligned}$$

Exercice

Exercice

Soit $\varphi = ((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$. Donner une formule équivalente n'utilisant que des $\vee$, $\wedge$ et des négations uniquement sur les variables.

$$
\begin{aligned}
\varphi &= ((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C \\
&\sim \neg((A \vee \neg B) \wedge (\neg C \vee A)) \vee C \\
&\sim (\neg(A \vee \neg B) \vee \neg(\neg C \vee A)) \vee C \\
&\sim ((\neg A \wedge \neg\neg B) \vee (\neg\neg C \wedge \neg A)) \vee C
\end{aligned}
$$

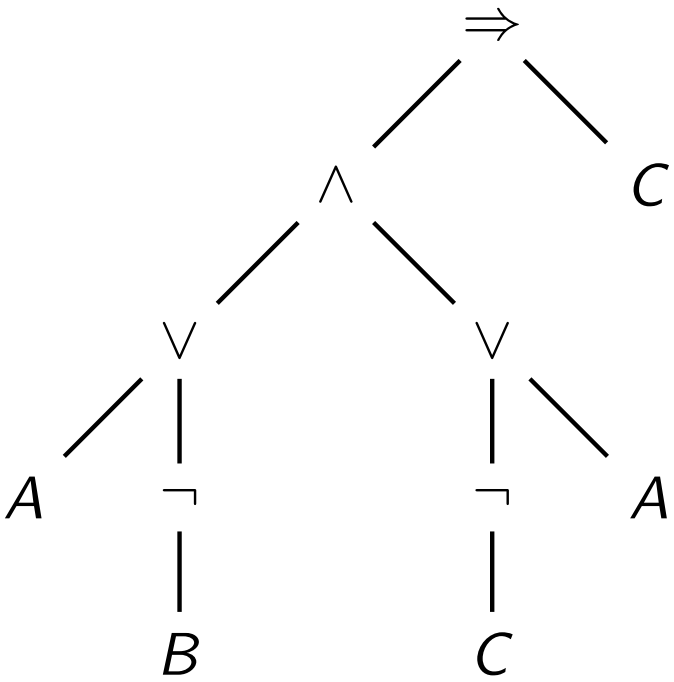
Exercice

Exercice

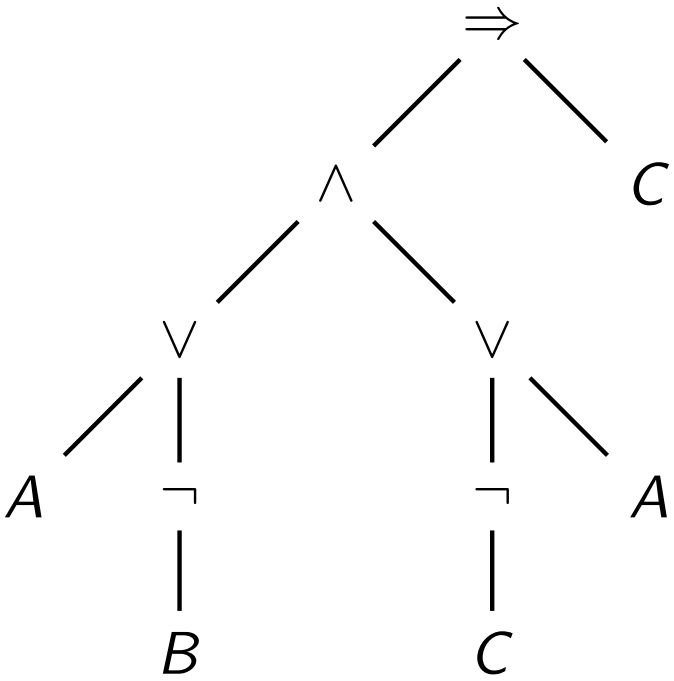
Soit $\varphi = ((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$. Donner une formule équivalente n'utilisant que des $\vee$, $\wedge$ et des négations uniquement sur les variables.

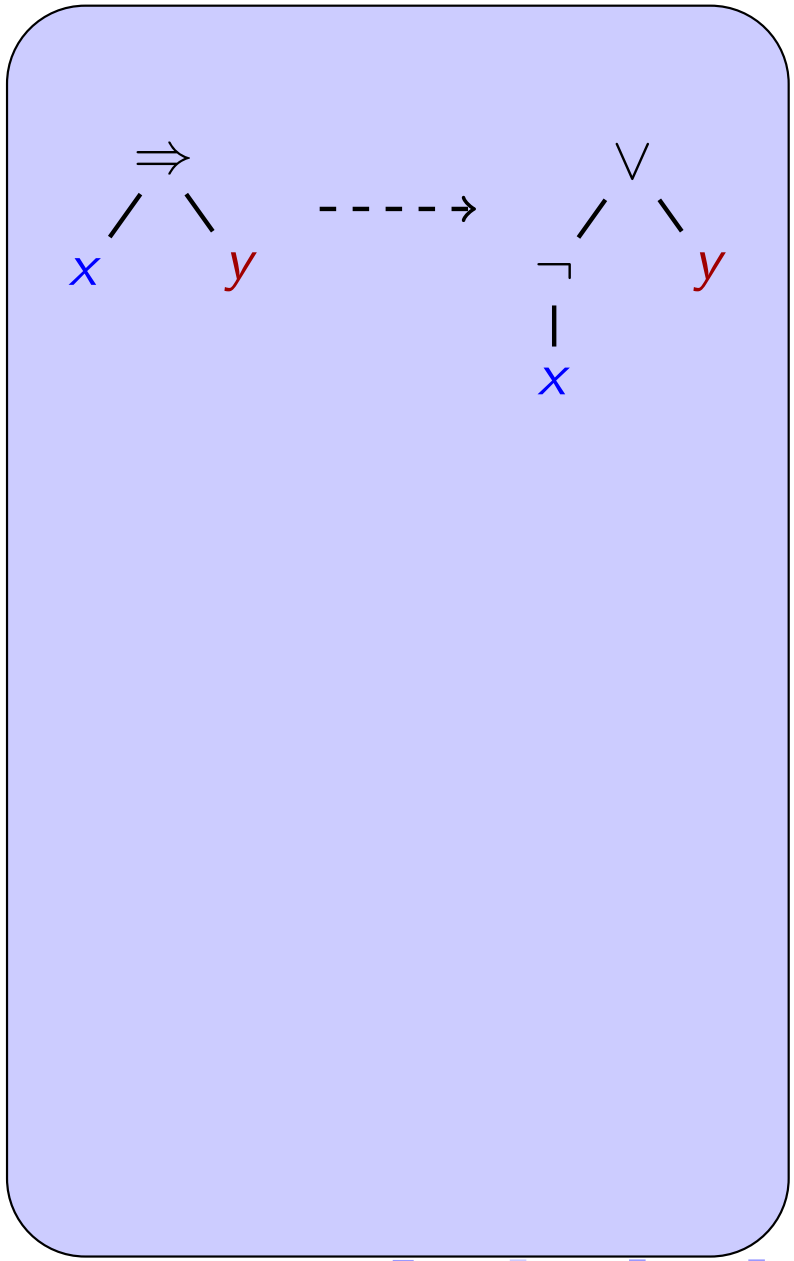
$$
\begin{aligned}
\varphi &= ((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C \\
&\sim \neg((A \vee \neg B) \wedge (\neg C \vee A)) \vee C \\
&\sim (\neg(A \vee \neg B) \vee \neg(\neg C \vee A)) \vee C \\
&\sim ((\neg A \wedge \neg\neg B) \vee (\neg\neg C \wedge \neg A)) \vee C \\
&\sim ((\neg A \wedge B) \vee (C \wedge \neg A)) \vee C
\end{aligned}
$$

Application sur un arbre : $((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$

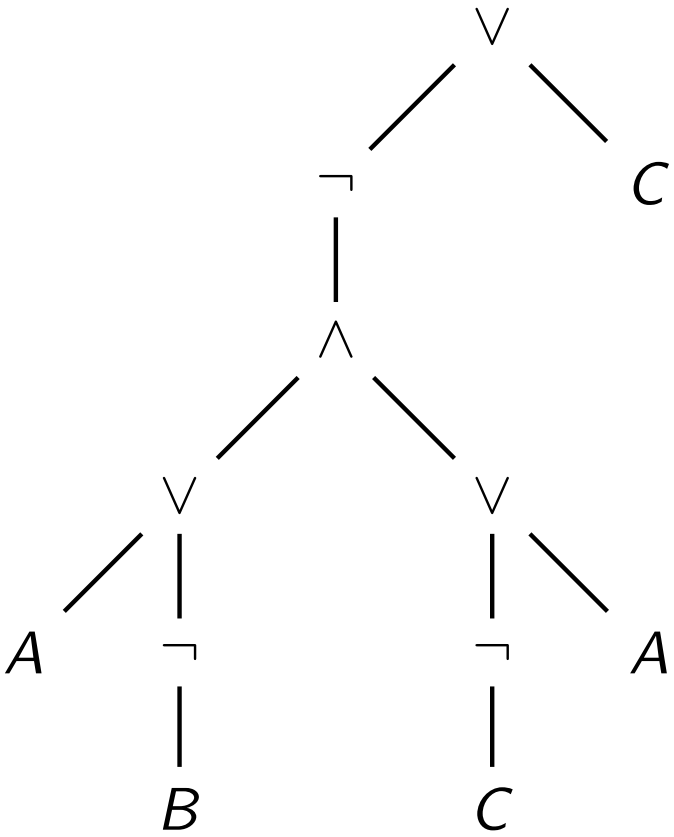


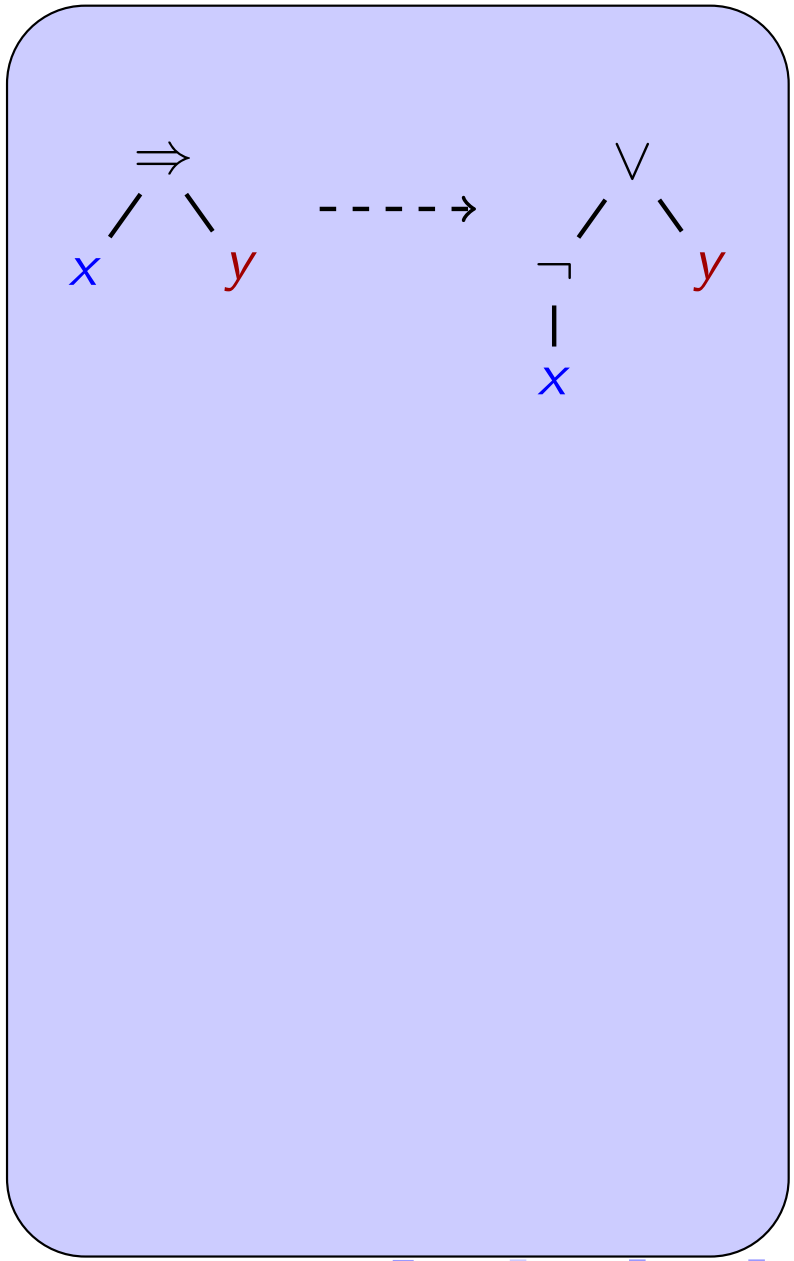
Application sur un arbre : $((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$



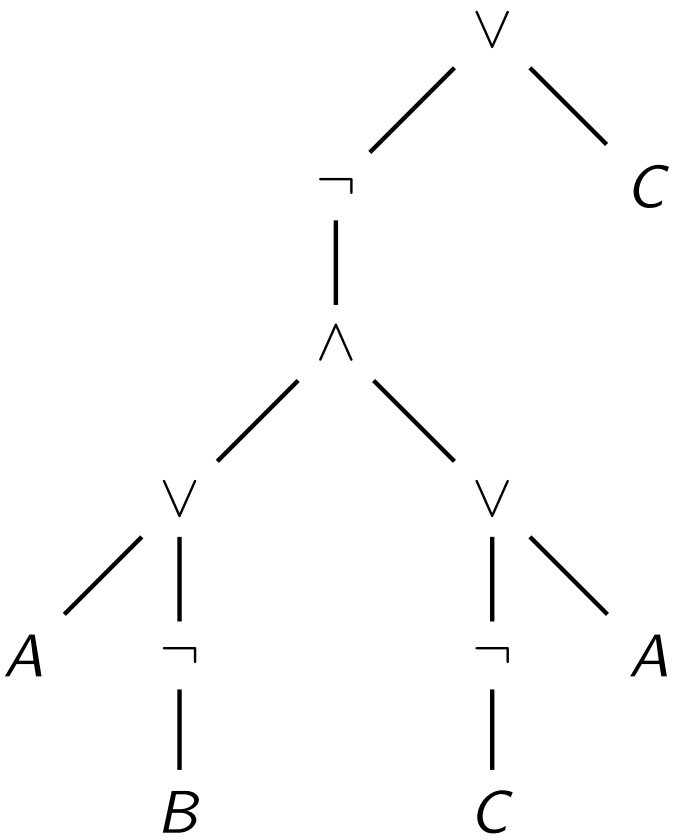


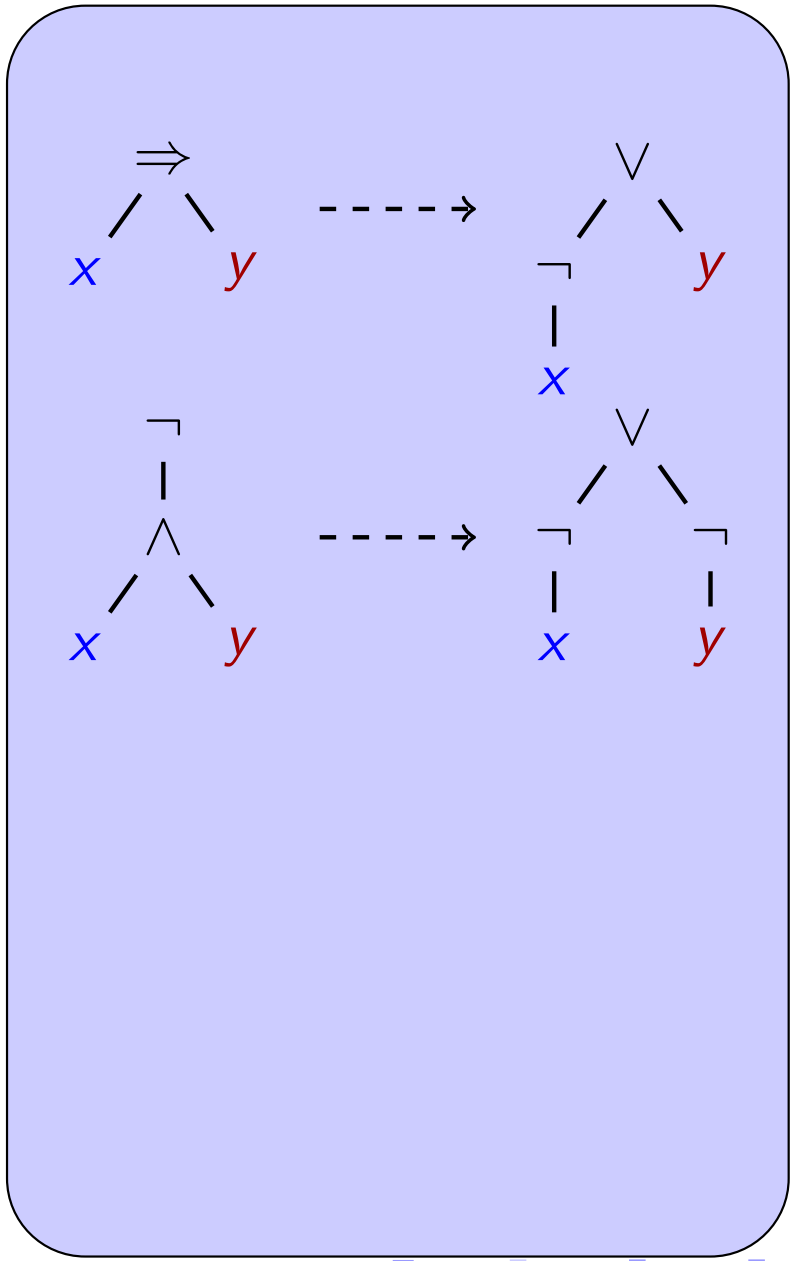
Application sur un arbre : $((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$



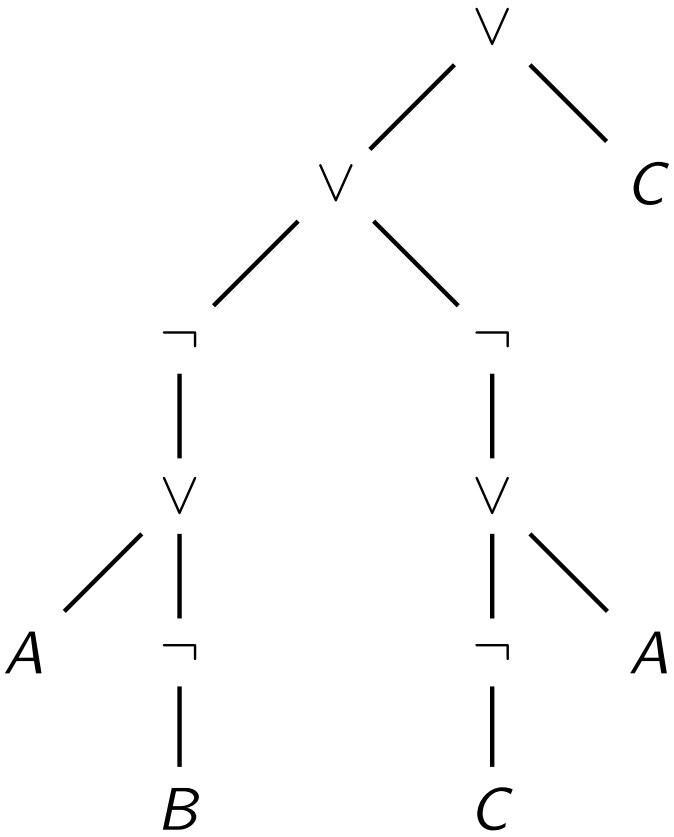


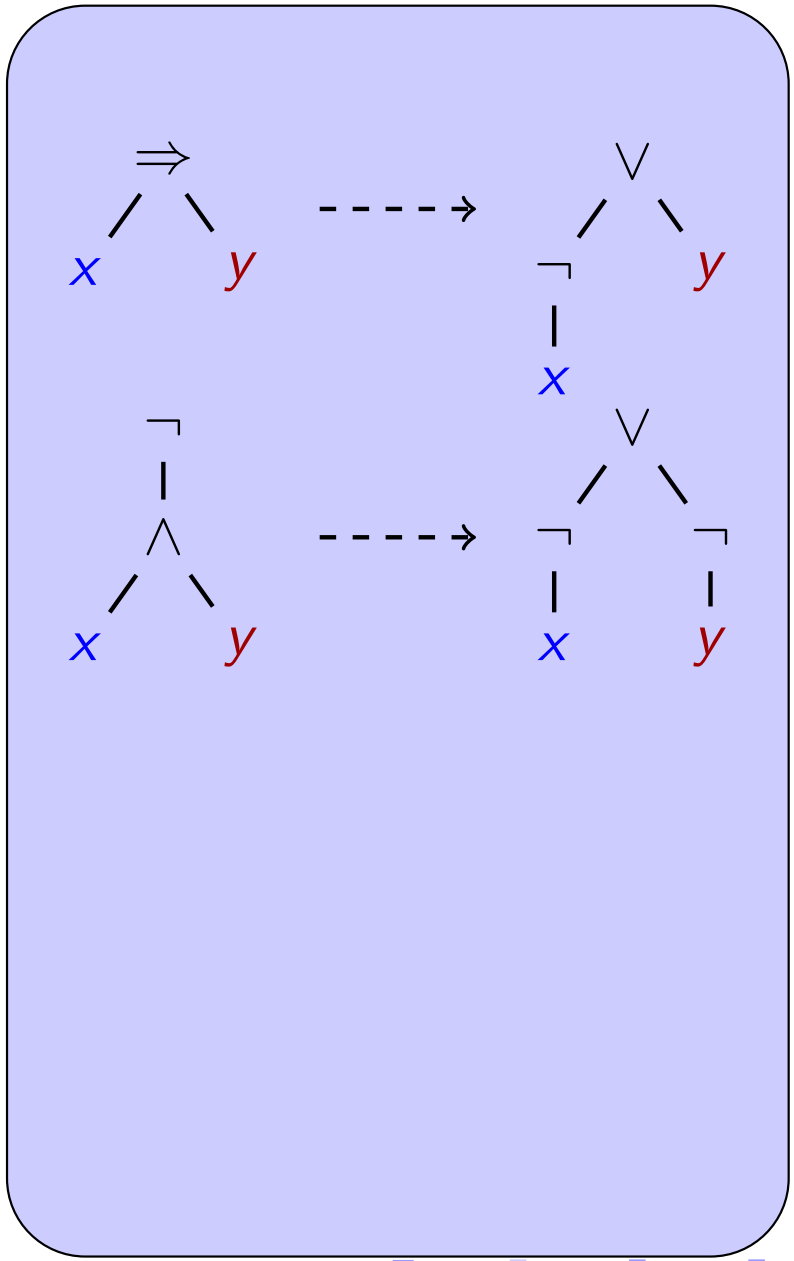
Application sur un arbre : $((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$



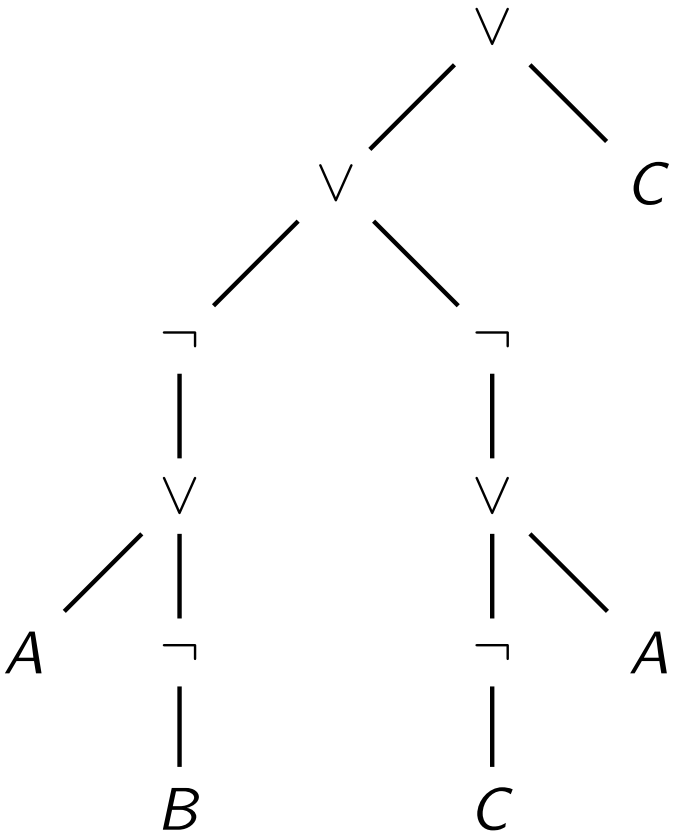


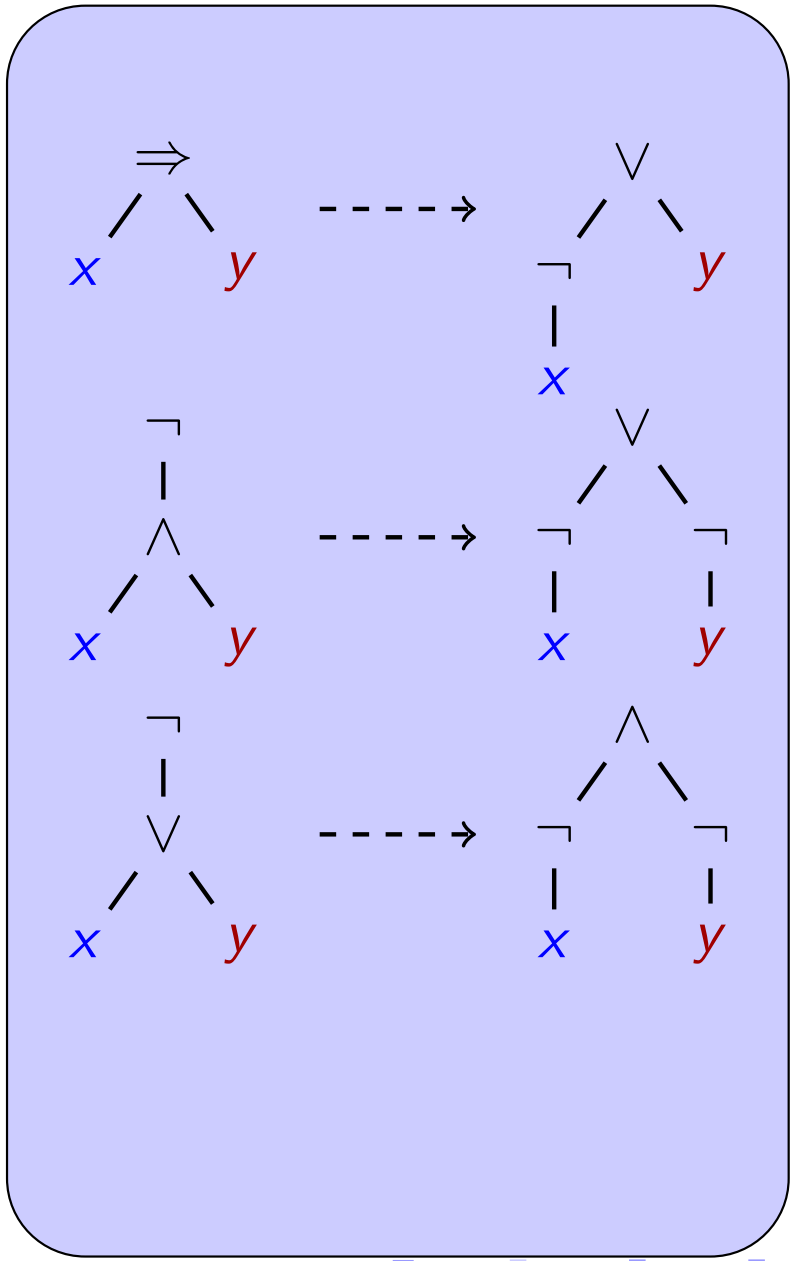
Application sur un arbre : $((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$



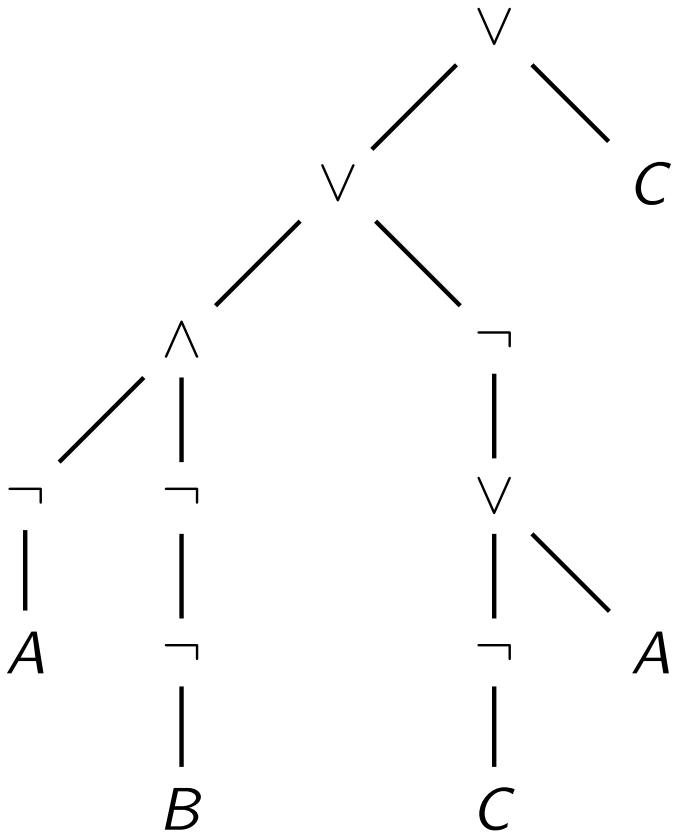


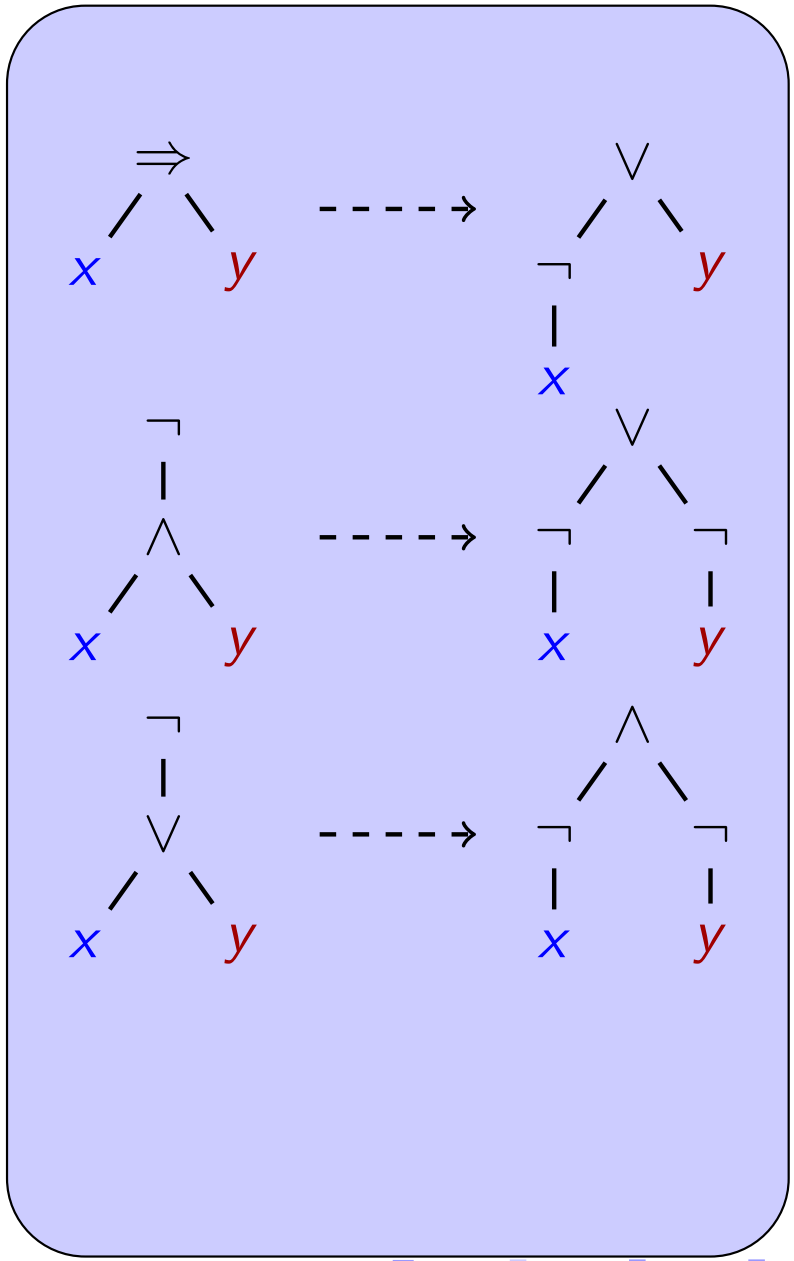
Application sur un arbre : $((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$



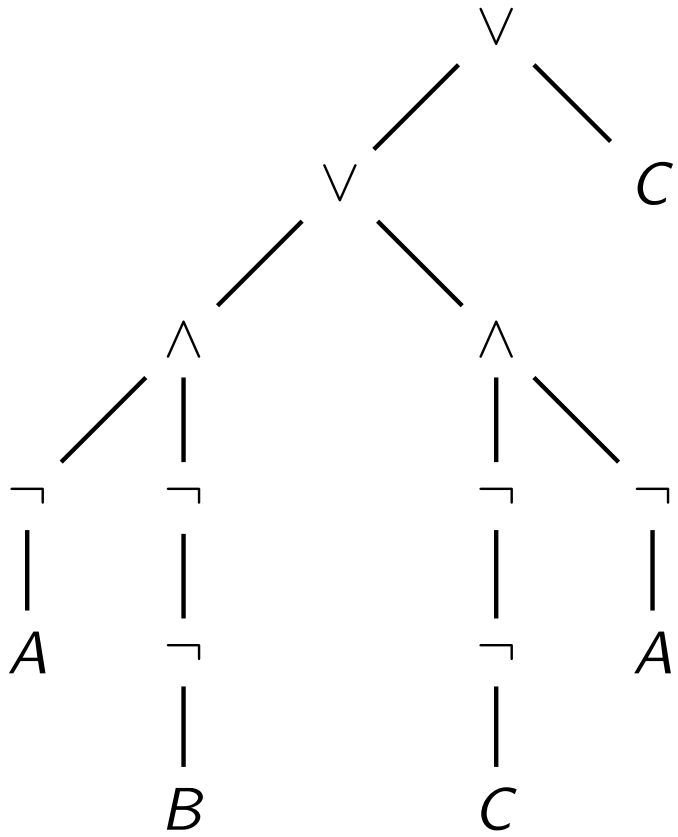


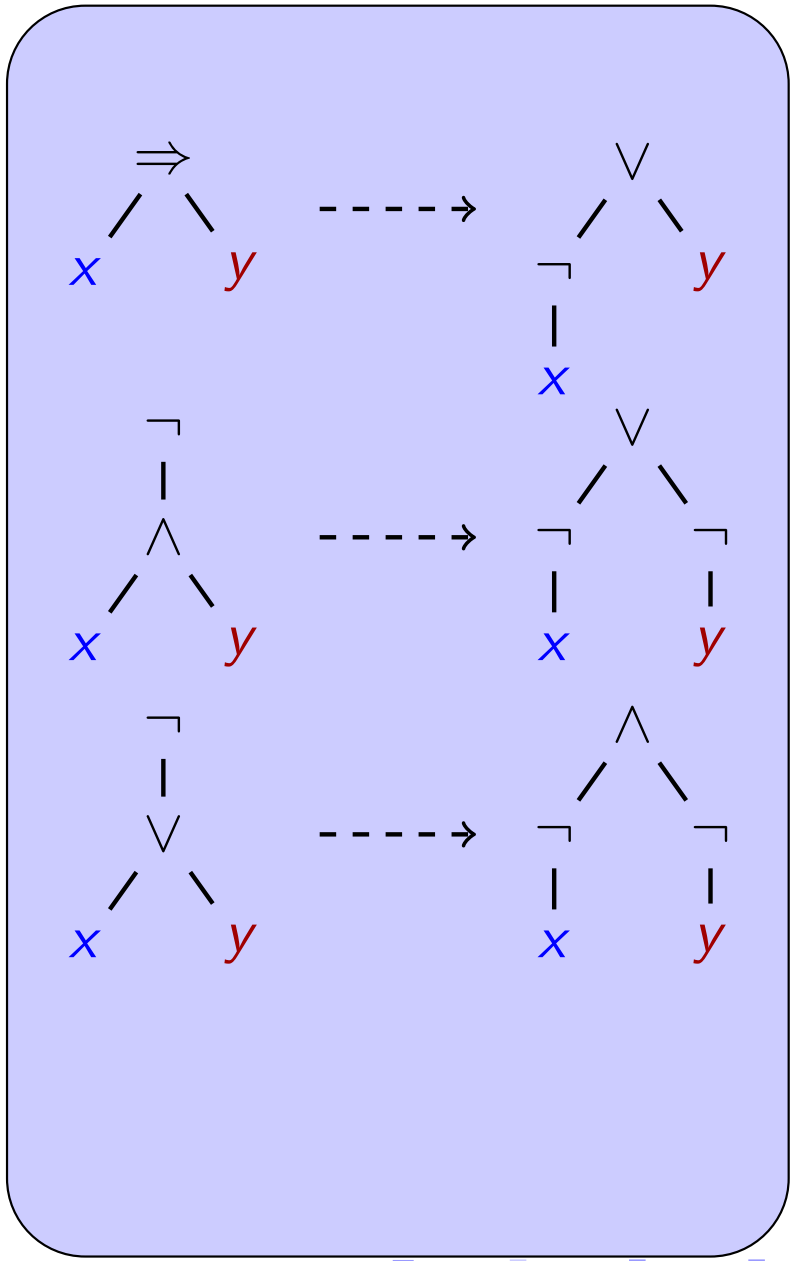
Application sur un arbre : $((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$



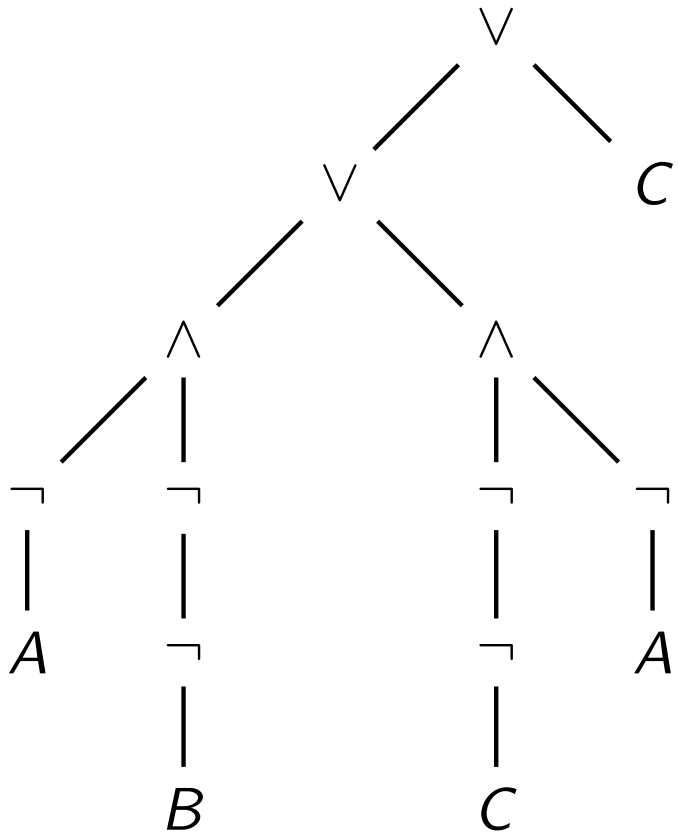


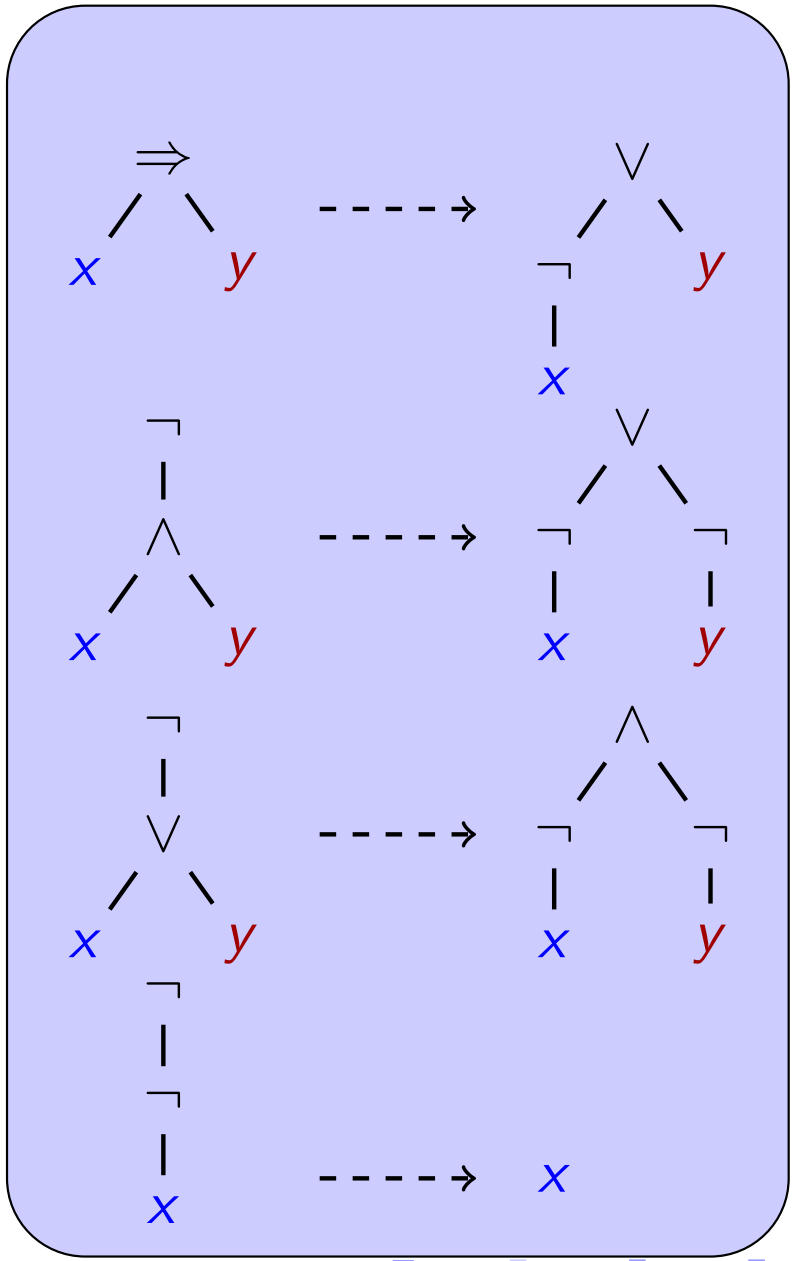
Application sur un arbre : $((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$



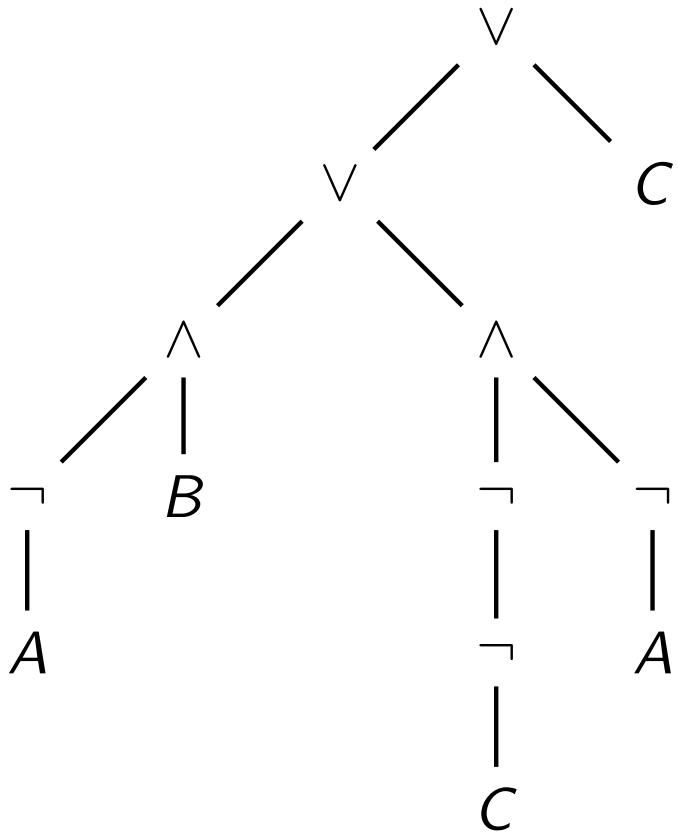


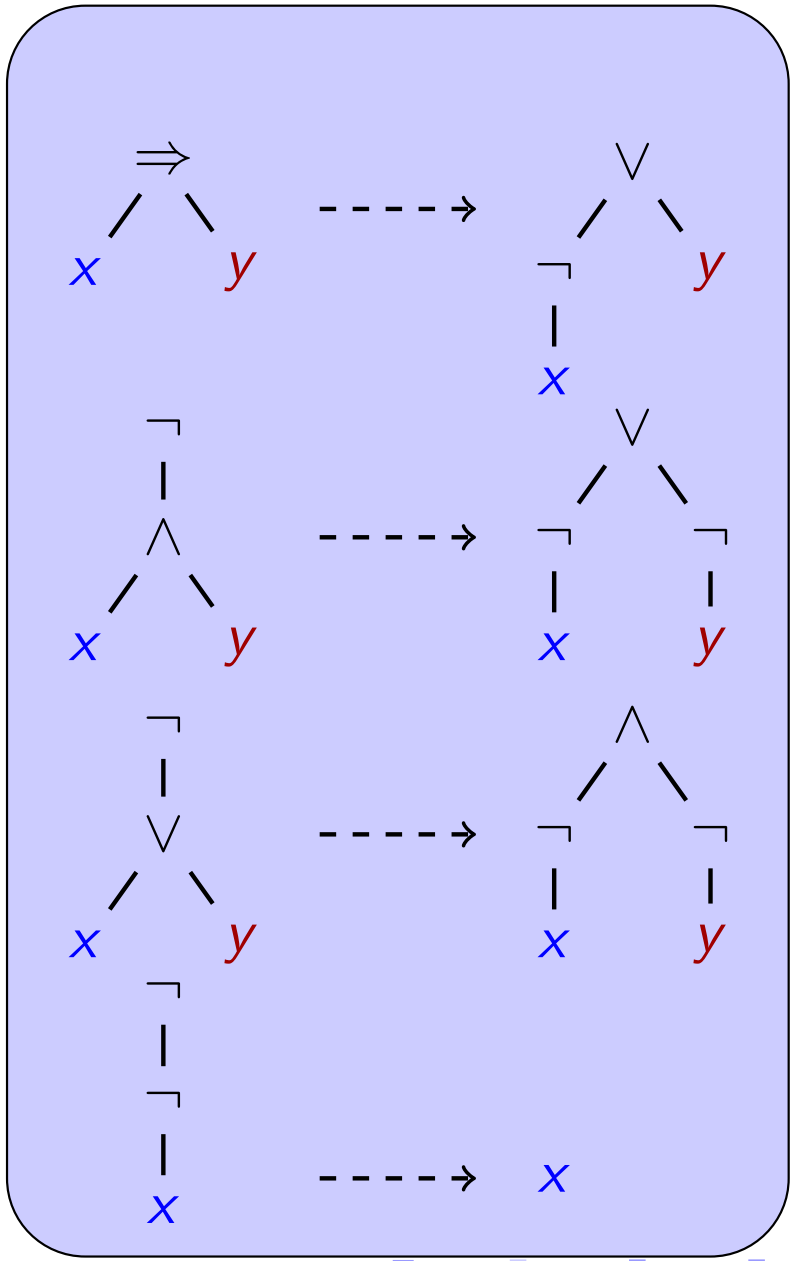
Application sur un arbre : $((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$



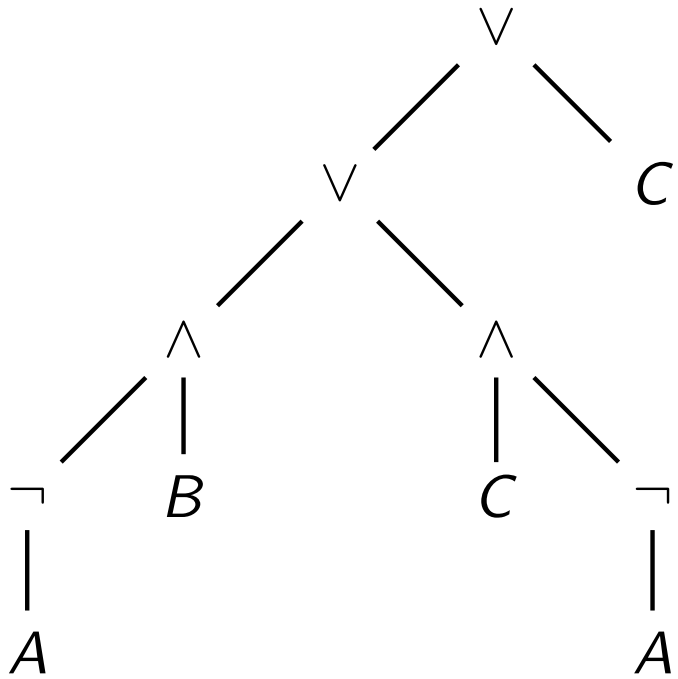


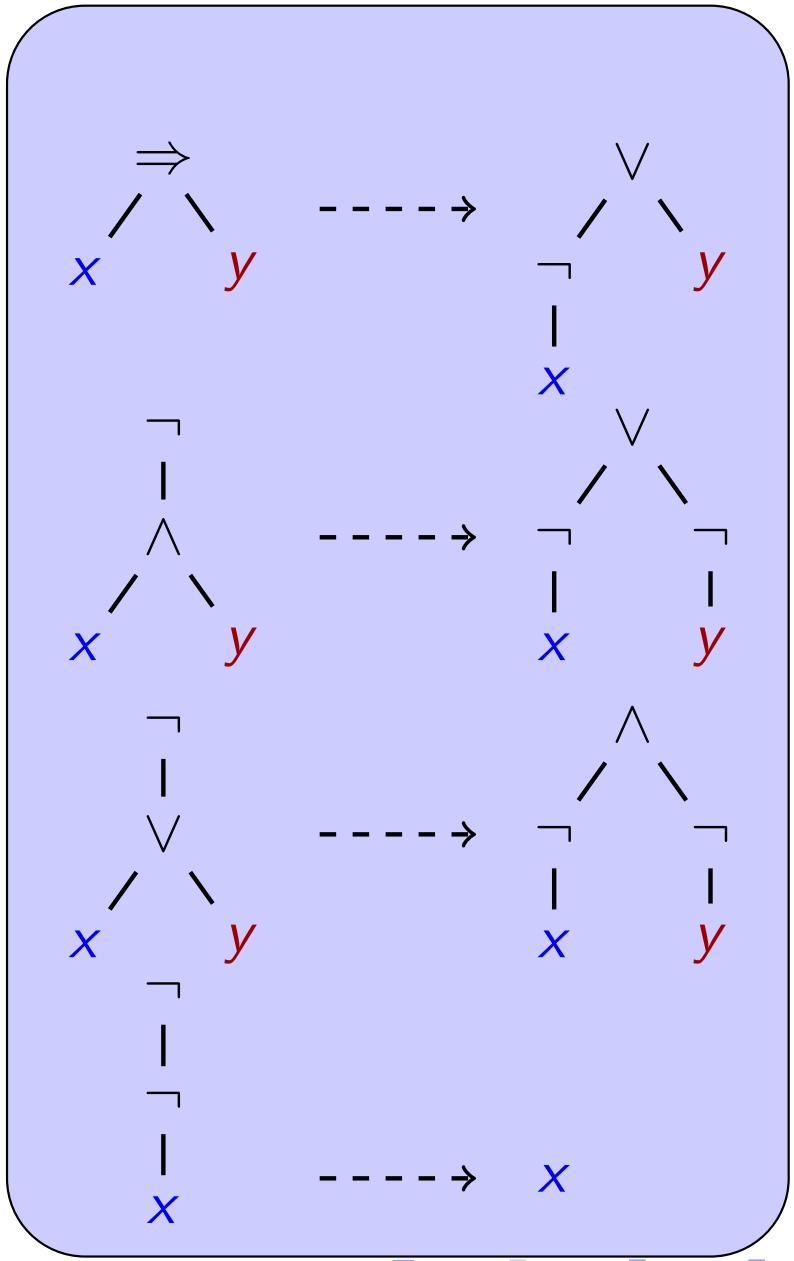
Application sur un arbre : $((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$





Application sur un arbre : $((A \vee \neg B) \wedge (\neg C \vee A)) \Rightarrow C$





Règle de réécriture

Definition

Une règle de réécriture est un couple de termes l et r, noté $l \to r$ vérifiant $\mathrm{var}(r) \subseteq \mathrm{var}(l)$.

Exemples :

- $f(x, h(y)) \to h(f(y, x))$

Règle de réécriture

Definition

Une règle de réécriture est un couple de termes l et r, noté $l \to r$ vérifiant $\mathrm{var}(r) \subseteq \mathrm{var}(l)$.

Exemples :

- $f(x, h(y)) \to h(f(y, x))$
- $h(f(x, y)) \to h(x)$

Règle de réécriture

Definition

Une règle de réécriture est un couple de termes l et r, noté $l \to r$ vérifiant $\mathrm{var}(r) \subseteq \mathrm{var}(l)$.

Exemples :

- $f(x, h(y)) \to h(f(y, x))$
- $h(f(x, y)) \to h(x)$
- $h(f(x, y)) \to f(h(x, z))$ n'est pas une règle de réécriture.

Substitution (de variables)

Définition (informelle)

Soit t un terme et σ une application de $\mathrm{var}(t)$ dans $\mathcal{T}(\Sigma)$. On note $t\sigma$ le terme clos obtenu en substituant dans t chaque occurrence d'une variable x par $\sigma(x)$.

Substitution (de variables)

Définition (informelle)

Soit t un terme et σ une application de $\mathrm{var}(t)$ dans $\mathcal{T}(\Sigma)$. On note $t\sigma$ le terme clos obtenu en substituant dans t chaque occurrence d'une variable x par $\sigma(x)$.

Exemple :

$$t = f(x, f(A, f(y, x)))$$

et $\sigma(x) = h(A)$ et $\sigma(y) = h(h(B))$, alors

$$t\sigma = f(h(A), f(A, f(h(h(B)), h(A))))$$

Substitution (définition formelle)

Soit t un terme et σ une application de $\mathrm{var}(t)$ dans $\mathcal{T}(\Sigma)$. Le terme clos $t\sigma$ est défini par :

- Si $t = x$ est une variable, alors $t\sigma = \sigma(x)$,
- Si $t = A$ est une constante, alors $t\sigma = t = A$,
- Si $t = f(t_1, \ldots, t_k)$, alors $t\sigma = f(t_1\sigma, \ldots, t_k\sigma)$,

Substitution (définition formelle)

Soit t un terme et σ une application de $\mathrm{var}(t)$ dans $\mathcal{T}(\Sigma)$. Le terme clos $t\sigma$ est défini par :

- Si $t = x$ est une variable, alors $t\sigma = \sigma(x)$,
- Si $t = A$ est une constante, alors $t\sigma = t = A$,
- Si $t = f(t_1, \ldots, t_k)$, alors $t\sigma = f(t_1\sigma, \ldots, t_k\sigma)$,

Exemple :

$$t = f(x, f(A, f(y, x)))$$

et $\sigma(x) = h(A)$ et $\sigma(y) = h(h(B))$, alors

$$t\sigma = f(x, f(A, f(y, x)))\sigma$$

Substitution (définition formelle)

Soit t un terme et σ une application de $\mathrm{var}(t)$ dans $\mathcal{T}(\Sigma)$. Le terme clos $t\sigma$ est défini par :

- Si $t = x$ est une variable, alors $t\sigma = \sigma(x)$,
- Si $t = A$ est une constante, alors $t\sigma = t = A$,
- Si $t = f(t_1, \ldots, t_k)$, alors $t\sigma = f(t_1\sigma, \ldots, t_k\sigma)$,

Exemple :

$$t = f(x, f(A, f(y, x)))$$

et $\sigma(x) = h(A)$ et $\sigma(y) = h(h(B))$, alors

$$\begin{aligned} t\sigma &= f(x, f(A, f(y, x)))\sigma \\ &= f(x\sigma, f(A, f(y, x))\sigma) \end{aligned}$$

Substitution (définition formelle)

Soit t un terme et σ une application de $\mathrm{var}(t)$ dans $\mathcal{T}(\Sigma)$. Le terme clos $t\sigma$ est défini par :

- Si $t = x$ est une variable, alors $t\sigma = \sigma(x)$,
- Si $t = A$ est une constante, alors $t\sigma = t = A$,
- Si $t = f(t_1, \ldots, t_k)$, alors $t\sigma = f(t_1\sigma, \ldots, t_k\sigma)$,

Exemple :

$$t = f(x, f(A, f(y, x)))$$

et $\sigma(x) = h(A)$ et $\sigma(y) = h(h(B))$, alors

$$\begin{aligned} t\sigma &= f(x, f(A, f(y, x)))\sigma \\ &= f(x\sigma, f(A, f(y, x))\sigma) \\ &= f(\sigma(x), f(A\sigma, f(y, x)\sigma)) \end{aligned}$$

Substitution (définition formelle)

Soit t un terme et σ une application de $\mathrm{var}(t)$ dans $\mathcal{T}(\Sigma)$. Le terme clos $t\sigma$ est défini par :

- Si $t = x$ est une variable, alors $t\sigma = \sigma(x)$,
- Si $t = A$ est une constante, alors $t\sigma = t = A$,
- Si $t = f(t_1, \ldots, t_k)$, alors $t\sigma = f(t_1\sigma, \ldots, t_k\sigma)$,

Exemple :

$$t = f(x, f(A, f(y, x)))$$

et $\sigma(x) = h(A)$ et $\sigma(y) = h(h(B))$, alors

$$\begin{aligned} t\sigma &= f(x, f(A, f(y, x)))\sigma \\ &= f(x\sigma, f(A, f(y, x))\sigma) \\ &= f(\sigma(x), f(A\sigma, f(y, x)\sigma)) \\ &= f(h(A), f(A, f(y\sigma, x\sigma))) \end{aligned}$$

Substitution (définition formelle)

Soit t un terme et σ une application de $\mathrm{var}(t)$ dans $\mathcal{T}(\Sigma)$. Le terme clos $t\sigma$ est défini par :

- Si $t = x$ est une variable, alors $t\sigma = \sigma(x)$,
- Si $t = A$ est une constante, alors $t\sigma = t = A$,
- Si $t = f(t_1, \ldots, t_k)$, alors $t\sigma = f(t_1\sigma, \ldots, t_k\sigma)$,

Exemple :

$$t = f(x, f(A, f(y, x)))$$

et $\sigma(x) = h(A)$ et $\sigma(y) = h(h(B))$, alors

$$\begin{aligned} t\sigma &= f(x, f(A, f(y, x)))\sigma \\ &= f(x\sigma, f(A, f(y, x))\sigma) \\ &= f(\sigma(x), f(A\sigma, f(y, x)\sigma)) \\ &= f(h(A), f(A, f(y\sigma, x\sigma))) \\ &= f(h(A), f(A, f(h(h(B)), h(A)))) \end{aligned}$$

Exercice

Soit $t = f(x, f(x, y))$ et σ définie par

- $\sigma(x) = f(h(A), h(B))$
- $\sigma(y) = B$

Que vaut $t\sigma$?

Exercice

Soit $t = f(x, f(x, y))$ et σ définie par

- $\sigma(x) = f(h(A), h(B))$
- $\sigma(y) = B$

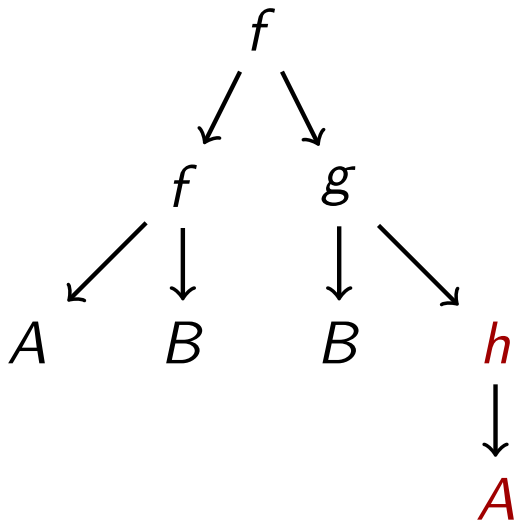
Que vaut $t\sigma$?
$f(f(h(A), h(B)), f(f(h(A), h(B)), B))$

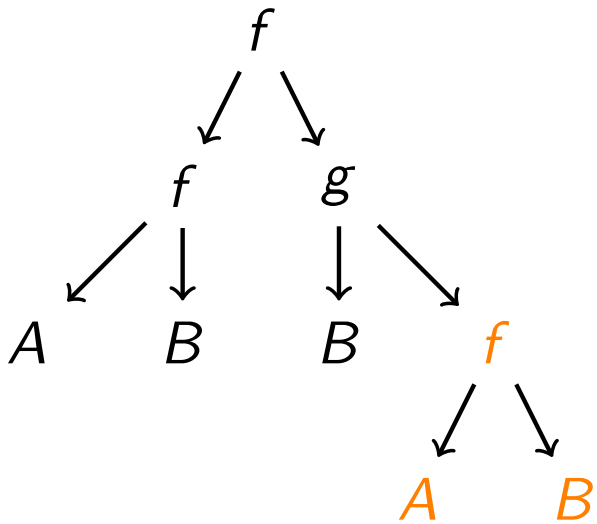
Substitution dans un terme

Définition (informelle)

La substitution d'un sous-terme à la position p dans un terme t par un terme t' est le terme obtenu à partir de t en remplaçant $t_{|p}$ par t'. On le note $t[t']_p$.

$t = f(f(A, B), g(B, h(A)))$ $\quad$ $t[f(A, B)]_{2\cdot 2} = f(f(A, B), g(B, f(A, B)))$





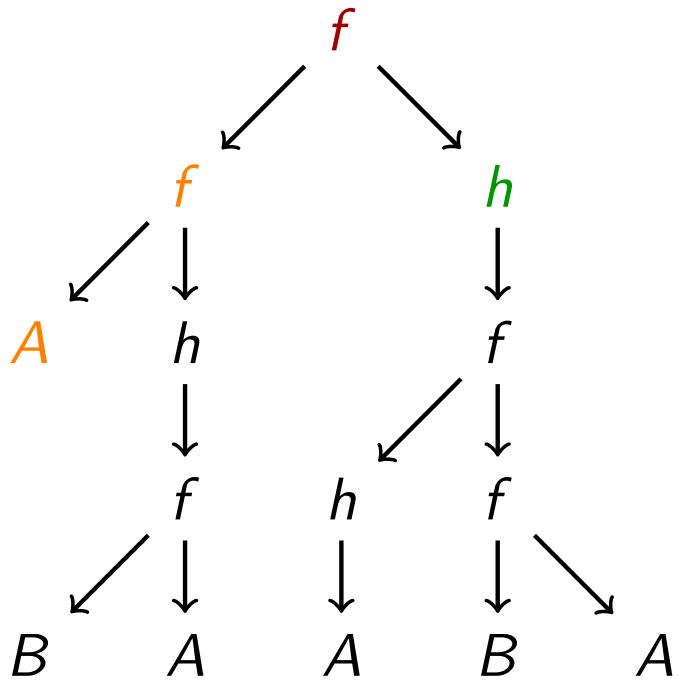
Exercice

On considère le terme

$$t = f(f(A,h(f(B,A))),h(f(h(A),f(B,B))))$$

Que vaut

1. $t[f(A,B)]_{\varepsilon}$?
2. $t[f(A,B)]_{1}$?
3. $t[f(A,B)]_{2\cdot 1\cdot 1}$?



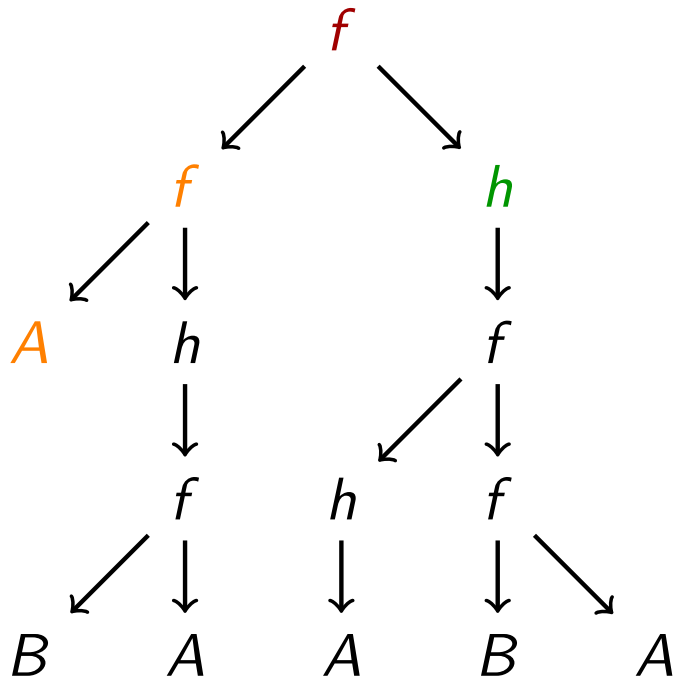
Exercice

On considère le terme

$$t = f(f(A,h(f(B,A))),h(f(h(A),f(B,B))))$$

Que vaut

1. $t[f(A,B)]_\varepsilon$?
   $f(A,B)$
2. $t[f(A,B)]_1$?
3. $t[f(A,B)]_{2\cdot 1\cdot 1}$?



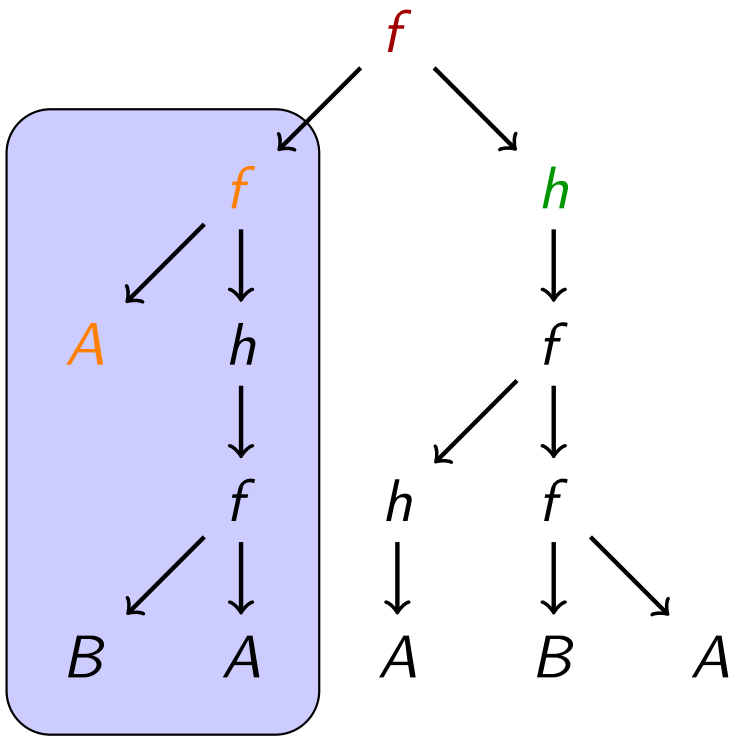
Exercice

On considère le terme

$$t = f(f(A,h(f(B,A))),h(f(h(A),f(B,B))))$$

Que vaut

1. $t[f(A,B)]_{\varepsilon}$?
   $f(A,B)$
2. $t[f(A,B)]_{1}$?
3. $t[f(A,B)]_{2\cdot 1\cdot 1}$?



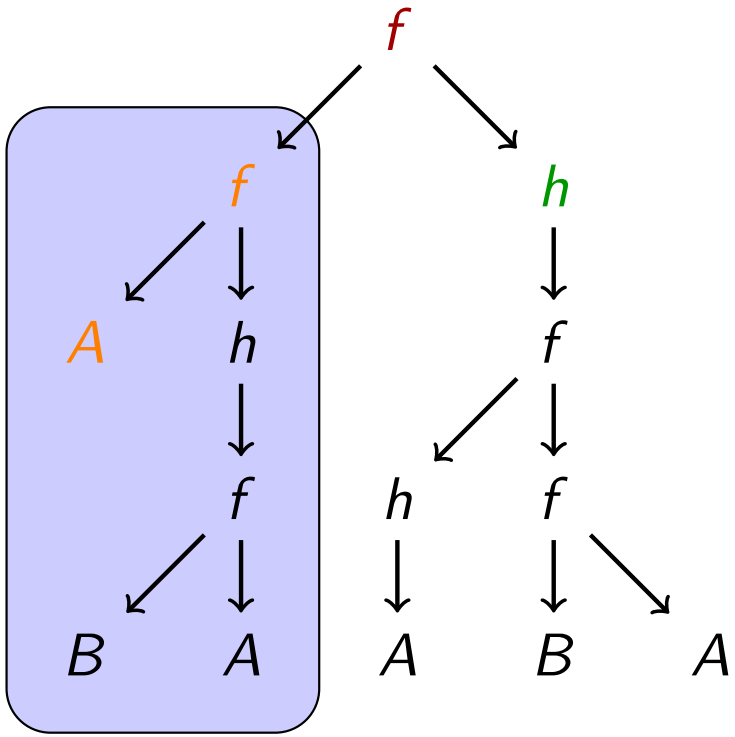
Exercice

On considère le terme

$$t = f(f(A,h(f(B,A))),h(f(h(A),f(B,B))))$$

Que vaut

1. $t[f(A,B)]_{\varepsilon}$?
   $f(A,B)$
2. $t[f(A,B)]_{1}$?
   $f(f(A,B),h(f(h(A),f(B,B))))$
3. $t[f(A,B)]_{2\cdot1\cdot1}$?



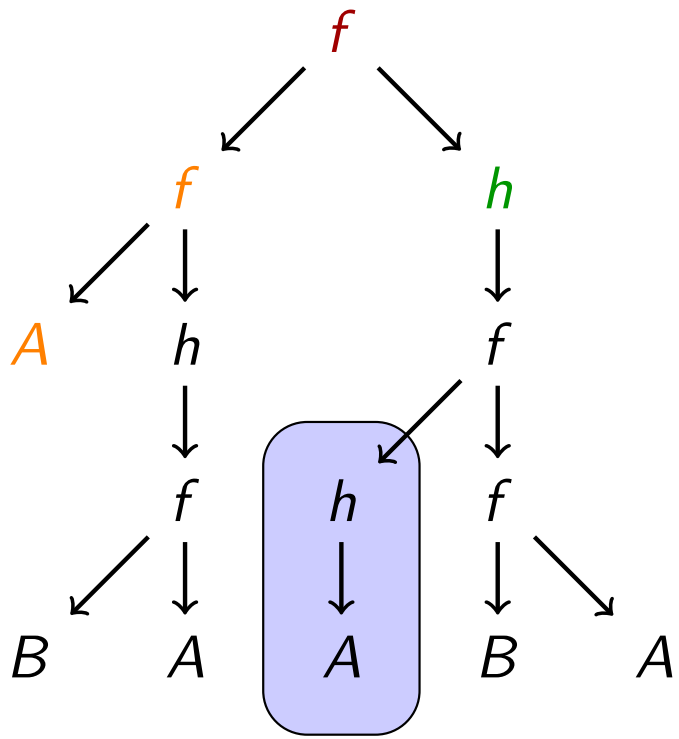
Exercice

On considère le terme

$$t = f(f(A,h(f(B,A))),h(f(h(A),f(B,B))))$$

Que vaut

1. $t[f(A,B)]_{\varepsilon}$?
   $f(A,B)$
2. $t[f(A,B)]_{1}$?
   $f(f(A,B),h(f(h(A),f(B,B))))$
3. $t[f(A,B)]_{2\cdot 1\cdot 1}$?



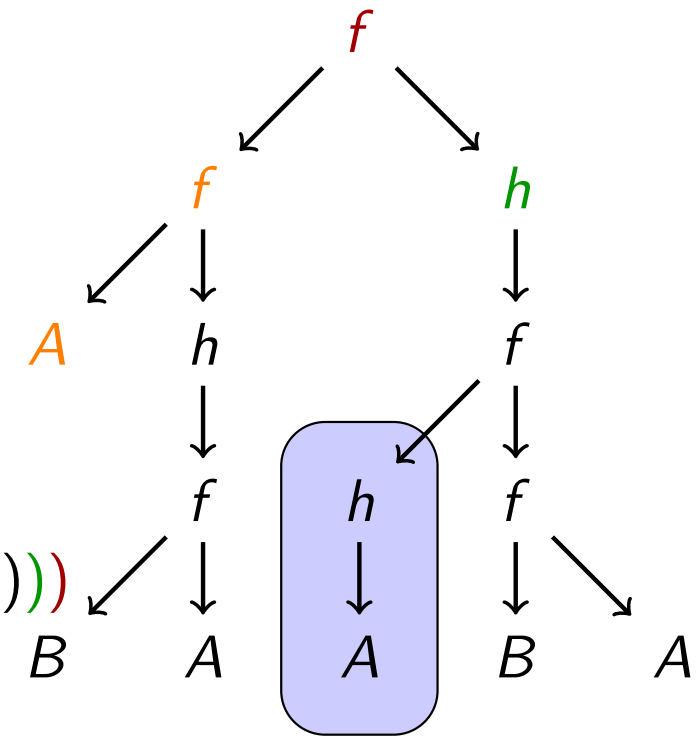
Exercice

On considère le terme

$$t = f(f(A,h(f(B,A))),h(f(h(A),f(B,B))))$$

Que vaut

1. $t[f(A,B)]_\varepsilon$?
   $f(A,B)$
2. $t[f(A,B)]_1$?
   $f(f(A,B),h(f(h(A),f(B,B))))$
3. $t[f(A,B)]_{2\cdot1\cdot1}$?
   $f(f(A,h(f(B,A))),h(f(f(A,B),f(B,B))))$



Réécriture

Définition

Soient t et t' deux termes clos et $l \to r$ une règle de réécriture. On dit que t se **réécrit** en t' en une étape s'il existe une position p de t et une substitution σ telles que

- $t_{|p} = l\sigma$ et
- $t' = t[r\sigma]_p$

On écrit alors $t \to_{l \to r} t'$ ou simplement $t \to t'$ s'il n'y a pas d'ambiguïté.

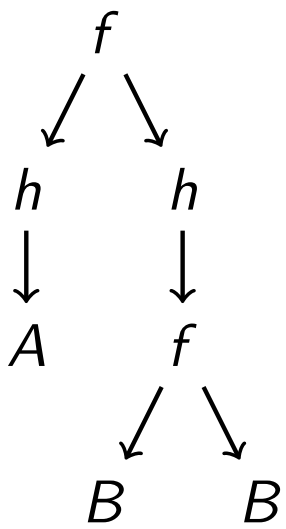
Réécriture

Définition

Soient t et t' deux termes clos et $l \to r$ une règle de réécriture. On dit que t se **réécrit** en t' en une étape s'il existe une position p de t et une substitution σ telles que

- $t_{|p} = l\sigma$ et
- $t' = t[r\sigma]_p$

On écrit alors $t \to_{l \to r} t'$ ou simplement $t \to t'$ s'il n'y a pas d'ambiguïté.



$t = f(h(A), h(f(B, B)))$ et $h(x) \to f(x, x)$

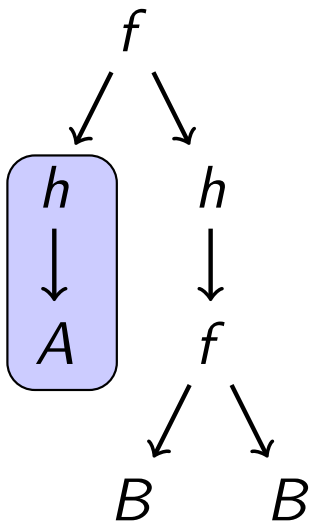
Réécriture

Définition

Soient t et t' deux termes clos et $l \to r$ une règle de réécriture. On dit que t se **réécrit** en t' en une étape s'il existe une position p de t et une substitution σ telles que

- $t_{|p} = l\sigma$ et
- $t' = t[r\sigma]_p$

On écrit alors $t \to_{l \to r} t'$ ou simplement $t \to t'$ s'il n'y a pas d'ambiguïté.



$t = f(h(A), h(f(B, B)))$ et $h(x) \to f(x, x)$
$p = 1$, $\sigma(x) = A$

- $t_{|p} = t_{|1} = h(A) = h(x)\sigma$
- $f(x, x)\sigma = f(A, A)$

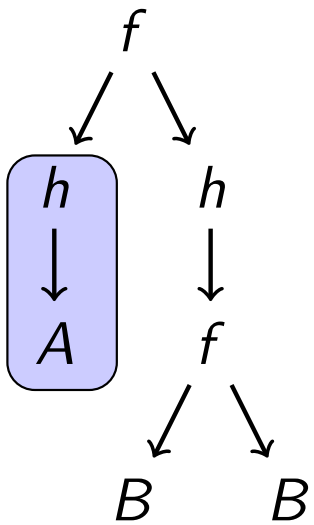
Réécriture

Définition

Soient t et t' deux termes clos et $l \to r$ une règle de réécriture. On dit que t se **réécrit** en t' en une étape s'il existe une position p de t et une substitution σ telles que

- $t_{|p} = l\sigma$ et
- $t' = t[r\sigma]_p$

On écrit alors $t \to_{l \to r} t'$ ou simplement $t \to t'$ s'il n'y a pas d'ambiguïté.



$t = f(h(A), h(f(B, B)))$ et $h(x) \to f(x, x)$
$p = 1$, $\sigma(x) = A$

- $t_{|p} = t_{|1} = h(A) = h(x)\sigma$
- $f(x, x)\sigma = f(A, A)$
- $t[r\sigma]_p = f(f(A, A), h(f(B, B)))$

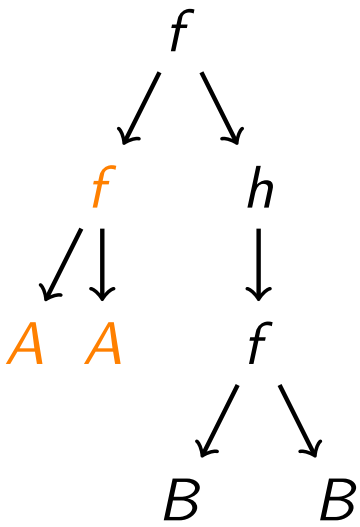
Réécriture

Définition

Soient t et t' deux termes clos et $l \to r$ une règle de réécriture. On dit que t se **réécrit** en t' en une étape s'il existe une position p de t et une substitution σ telles que

- $t_{|p} = l\sigma$ et
- $t' = t[r\sigma]_p$

On écrit alors $t \to_{l \to r} t'$ ou simplement $t \to t'$ s'il n'y a pas d'ambiguïté.



$t = f(h(A), h(f(B, B)))$ et $h(x) \to f(x, x)$
$p = 1$, $\sigma(x) = A$

- $t_{|p} = t_{|1} = h(A) = h(x)\sigma$
- $f(x, x)\sigma = f(A, A)$
- $t[r\sigma]_p = f(f(A, A), h(f(B, B)))$

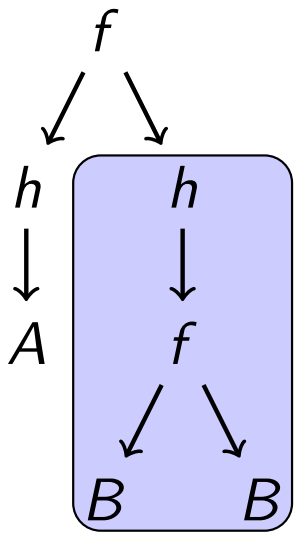
Réécriture

Définition

Soient t et t' deux termes clos et $l \to r$ une règle de réécriture. On dit que t se **réécrit** en t' en une étape s'il existe une position p de t et une substitution σ telles que

- $t_{|p} = l\sigma$ et
- $t' = t[r\sigma]_p$

On écrit alors $t \to_{l \to r} t'$ ou simplement $t \to t'$ s'il n'y a pas d'ambiguïté.



$t = f(h(A), h(f(B, B)))$ et $h(x) \to f(x, x)$
$p = 2$, $\sigma(x) = f(B, B)$

- $t_{|p} = t_{|2} = h(f(B, B)) = h(x)\sigma$
- $f(x, x)\sigma = f(f(B, B), f(B, B))$
- $t[r\sigma]_p = f(h(A), f(f(B, B), f(B, B)))$

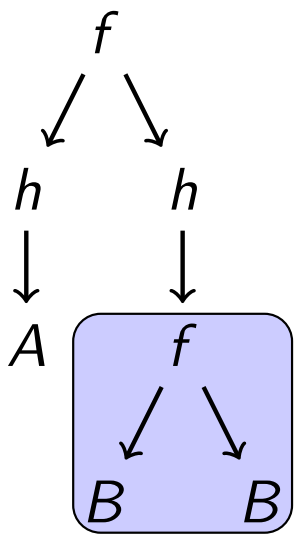
Réécriture

Définition

Soient t et t' deux termes clos et $l \to r$ une règle de réécriture. On dit que t se **réécrit** en t' en une étape s'il existe une position p de t et une substitution σ telles que

- $t_{|p} = l\sigma$ et
- $t' = t[r\sigma]_p$

On écrit alors $t \to_{l \to r} t'$ ou simplement $t \to t'$ s'il n'y a pas d'ambiguïté.



$t = f(h(A), h(f(B, B)))$ et $f(x, x) \to h(x)$

$t \to_{f(x,x) \to h(x)} f(h(A), h(h(B)))$

En revanche, impossible de réécriture à la racine car les sous-arbres sont différents.

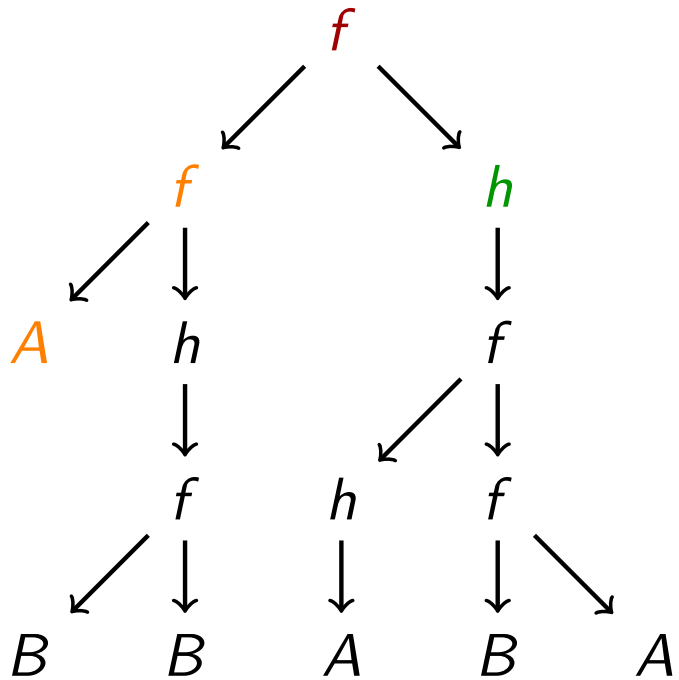
Exercice

On considère le terme

$$t = f(f(A,h(f(B,B))),h(f(h(A),f(B,B))))$$

Comment peut-on le réécrire en utilisant

1. $h(x) \to h(h(x))$
2. $f(x,y) \to h(x)$
3. $h(f(x,y)) \to f(y,x)$
4. $f(x,x) \to h(x)$
5. $f(h(x),f(y,x)) \to h(y)$



Systèmes de réécriture

Définitions

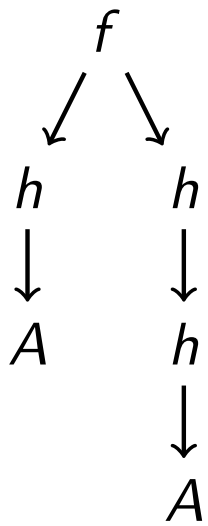
Un système de réécriture $\mathcal{R}$ est un ensemble **fini** de règles de réécriture.
On dit qu'une terme t se réécrit en t' par $\mathcal{R}$, noté $t \to_{\mathcal{R}} t'$, s'il existe une règle $l \to r$ de $\mathcal{R}$ telle que $t \to_{l \to r} t'$.

Systèmes de réécriture

Définitions

Un système de réécriture $\mathcal{R}$ est un ensemble **fini** de règles de réécriture.
On dit qu'une terme t se réécrit en t' par $\mathcal{R}$, noté $t \to_{\mathcal{R}} t'$, s'il existe une règle $l \to r$ de $\mathcal{R}$ telle que $t \to_{l \to r} t'$.

Avec $\mathcal{R} = \{h(x) \to h(h(x)),\ f(x,x) \to h(x)\}$, on a par exemple

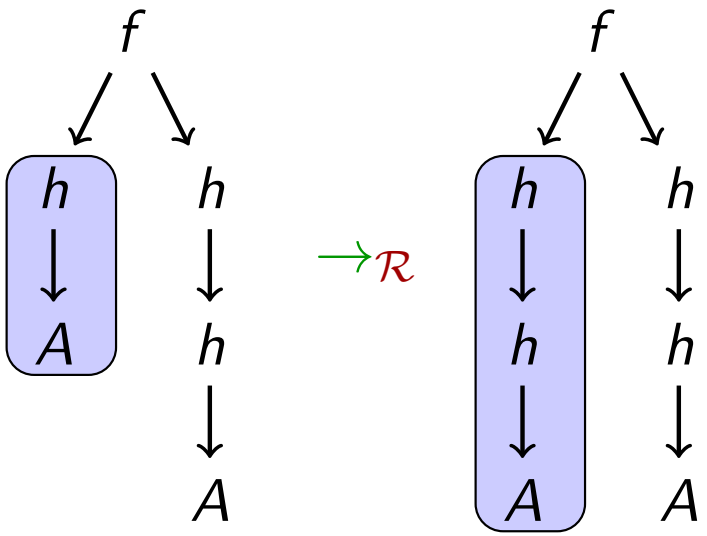


Systèmes de réécriture

Définitions

Un système de réécriture $\mathcal{R}$ est un ensemble **fini** de règles de réécriture.
On dit qu'une terme t se réécrit en t' par $\mathcal{R}$, noté $t \to_{\mathcal{R}} t'$, s'il existe une règle $l \to r$ de $\mathcal{R}$ telle que $t \to_{l \to r} t'$.

Avec $\mathcal{R} = \{h(x) \to h(h(x)),\ f(x,x) \to h(x)\}$, on a par exemple

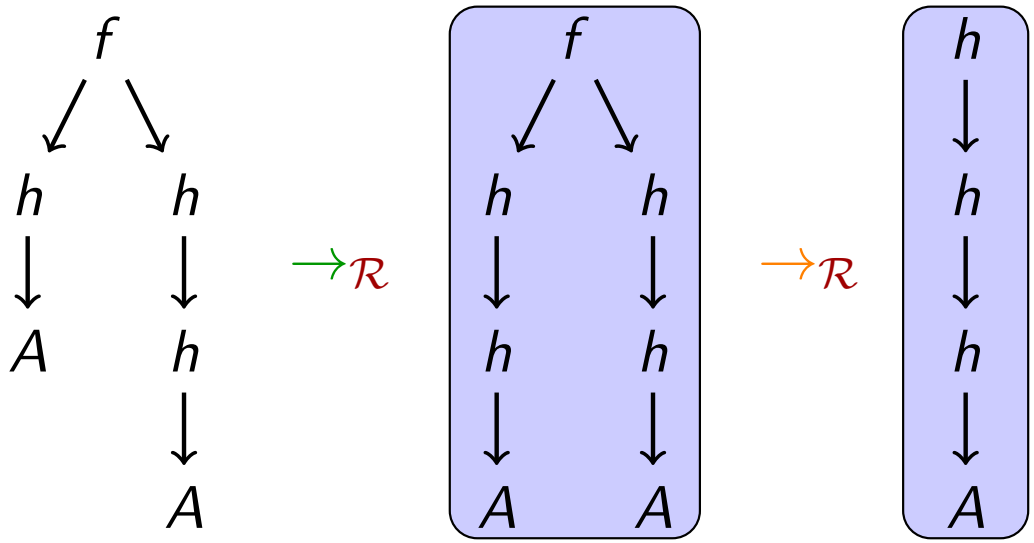


Systèmes de réécriture

Définitions

Un système de réécriture $\mathcal{R}$ est un ensemble **fini** de règles de réécriture.
On dit qu'une terme t se réécrit en t' par $\mathcal{R}$, noté $t \to_{\mathcal{R}} t'$, s'il existe une règle $l \to r$ de $\mathcal{R}$ telle que $t \to_{l \to r} t'$.

Avec $\mathcal{R} = \{h(x) \to h(h(x)),\ f(x,x) \to h(x)\}$, on a par exemple



Clôture réflexive transitive

Définition

On note $\rightarrow_{\mathcal{R}}^{*}$ la clôture réflexive transitive de la relation $\rightarrow_{\mathcal{R}}$.

- Pour tout terme clos t, on a $t \rightarrow_{\mathcal{R}}^{*} t$.
- Pour tous termes clos t et t', on a $t \rightarrow_{\mathcal{R}}^{*} t'$, s'il existe des termes clos $t_1, \ldots, t_k$ tels que
  - $t = t_1$,
  - $t_1 \rightarrow_{\mathcal{R}} t_2 \rightarrow_{\mathcal{R}} \ldots \rightarrow_{\mathcal{R}} t_k$
  - $t' = t_k$,

Clôture réflexive transitive

Définition

On note $\rightarrow_{\mathcal{R}}^{*}$ la clôture réflexive transitive de la relation $\rightarrow_{\mathcal{R}}$.

- Pour tout terme clos t, on a $t \rightarrow_{\mathcal{R}}^{*} t$.
- Pour tous termes clos t et t', on a $t \rightarrow_{\mathcal{R}}^{*} t'$, s'il existe des termes clos $t_1, \ldots, t_k$ tels que
  - $t = t_1$,
  - $t_1 \rightarrow_{\mathcal{R}} t_2 \rightarrow_{\mathcal{R}} \ldots \rightarrow_{\mathcal{R}} t_k$
  - $t' = t_k$,

$$\mathcal{R} = \{h(h(x)) \rightarrow h(x),\ h(f(x,x)) \rightarrow h(x)\}$$

$$h(h(f(h(A), h(h(A))))) \rightarrow h(h(f(h(A), h(A))))$$

Clôture réflexive transitive

Définition

On note $\rightarrow_{\mathcal{R}}^{*}$ la clôture réflexive transitive de la relation $\rightarrow_{\mathcal{R}}$.

- Pour tout terme clos t, on a $t \rightarrow_{\mathcal{R}}^{*} t$.
- Pour tous termes clos t et t', on a $t \rightarrow_{\mathcal{R}}^{*} t'$, s'il existe des termes clos $t_1, \ldots, t_k$ tels que
  - $t = t_1$,
  - $t_1 \rightarrow_{\mathcal{R}} t_2 \rightarrow_{\mathcal{R}} \ldots \rightarrow_{\mathcal{R}} t_k$
  - $t' = t_k$,

$\mathcal{R} = \{h(h(x)) \rightarrow h(x),\ h(f(x,x)) \rightarrow h(x)\}$

$h(h(f(h(A), h(h(A))))) \rightarrow h(h(f(h(A), h(A)))) \rightarrow h(h(A))$

Clôture réflexive transitive

Définition

On note $\rightarrow_{\mathcal{R}}^{*}$ la clôture réflexive transitive de la relation $\rightarrow_{\mathcal{R}}$.

- Pour tout terme clos t, on a $t \rightarrow_{\mathcal{R}}^{*} t$.
- Pour tous termes clos t et t', on a $t \rightarrow_{\mathcal{R}}^{*} t'$, s'il existe des termes clos $t_1, \ldots, t_k$ tels que
  - $t = t_1$,
  - $t_1 \rightarrow_{\mathcal{R}} t_2 \rightarrow_{\mathcal{R}} \ldots \rightarrow_{\mathcal{R}} t_k$
  - $t' = t_k$,

$$\mathcal{R} = \{h(h(x)) \rightarrow h(x),\ h(f(x,x)) \rightarrow h(x)\}$$

$$h(h(f(h(A), h(h(A))))) \rightarrow h(h(f(h(A), h(A)))) \rightarrow h(h(A)) \rightarrow h(A)$$

Clôture réflexive transitive

Définition

On note $\rightarrow_{\mathcal{R}}^{*}$ la clôture réflexive transitive de la relation $\rightarrow_{\mathcal{R}}$.

- Pour tout terme clos t, on a $t \rightarrow_{\mathcal{R}}^{*} t$.
- Pour tous termes clos t et t', on a $t \rightarrow_{\mathcal{R}}^{*} t'$, s'il existe des termes clos $t_1, \ldots, t_k$ tels que
  - $t = t_1$,
  - $t_1 \rightarrow_{\mathcal{R}} t_2 \rightarrow_{\mathcal{R}} \ldots \rightarrow_{\mathcal{R}} t_k$
  - $t' = t_k$,

$\mathcal{R} = \{h(h(x)) \rightarrow h(x),\ h(f(x,x)) \rightarrow h(x)\}$

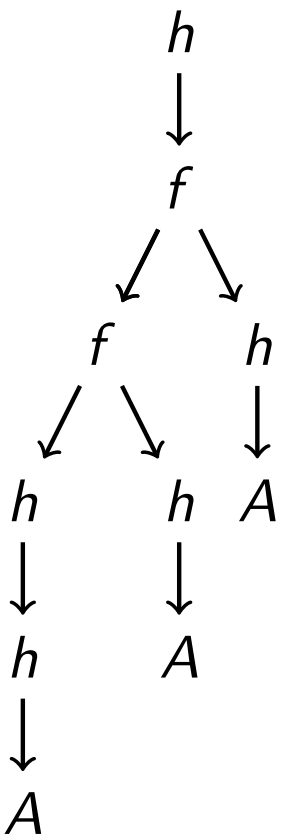
$h(h(f(h(A), h(h(A))))) \rightarrow h(h(f(h(A), h(A)))) \rightarrow h(h(A)) \rightarrow h(A)$

Donc $h(h(f(h(A), h(h(A))))) \rightarrow^{*} h(A)$.

Exercice

Exercice

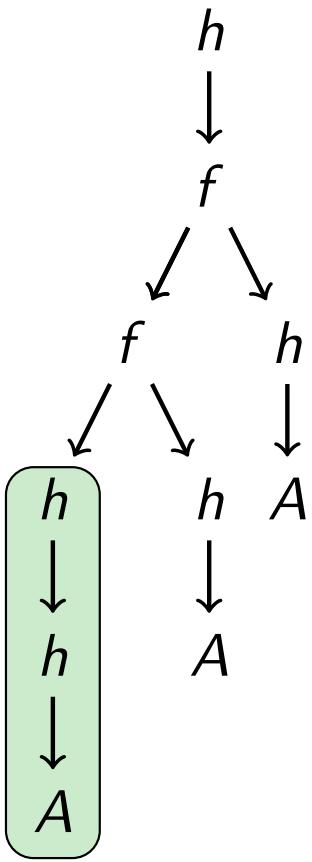
Soient $\mathcal{R} = \{h(h(x)) \to h(x),\ f(x,x)) \to x\}$ et t le terme si dessous.
Montrer que $t \to^*_{\mathcal{R}} h(A)$



Exercice

Exercice

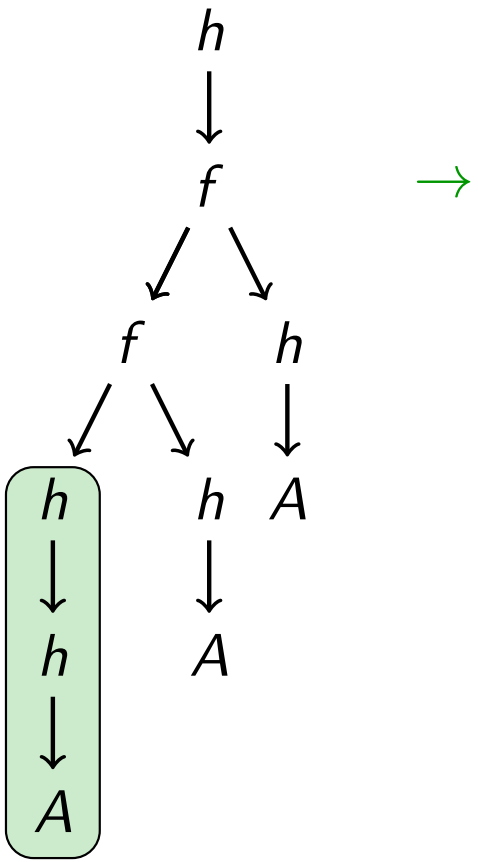
Soient $\mathcal{R} = \{h(h(x)) \to h(x),\ f(x,x)) \to x\}$ et t le terme si dessous.
Montrer que $t \to^*_{\mathcal{R}} h(A)$



Exercice

Exercice

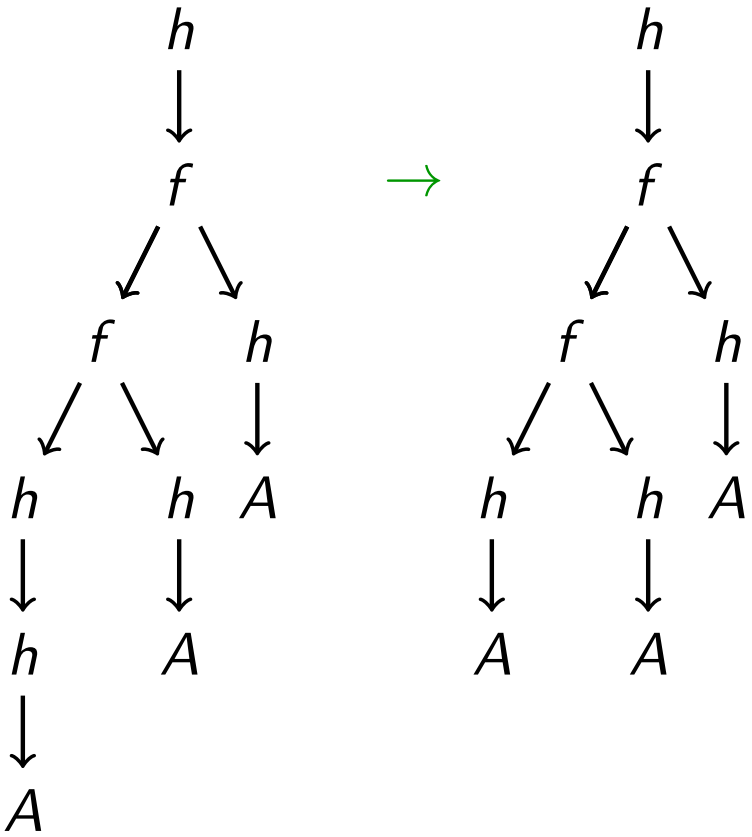
Soient $\mathcal{R} = \{h(h(x)) \to h(x),\ f(x,x)) \to x\}$ et t le terme si dessous.
Montrer que $t \to^*_{\mathcal{R}} h(A)$



Exercice

Exercice

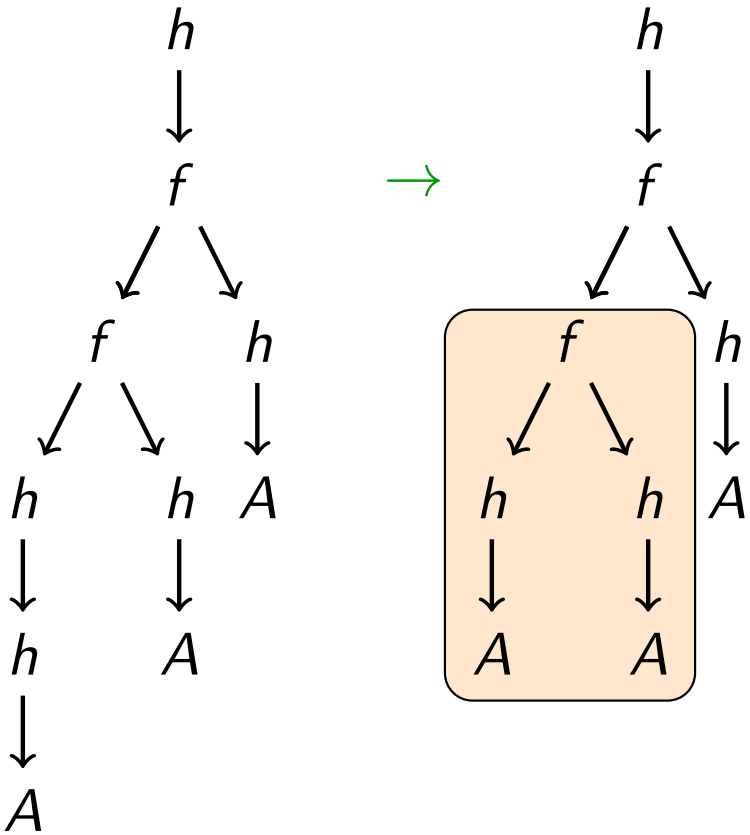
Soient $\mathcal{R} = \{h(h(x)) \to h(x),\ f(x,x)) \to x\}$ et t le terme si dessous.
Montrer que $t \to^*_{\mathcal{R}} h(A)$



Exercice

Exercice

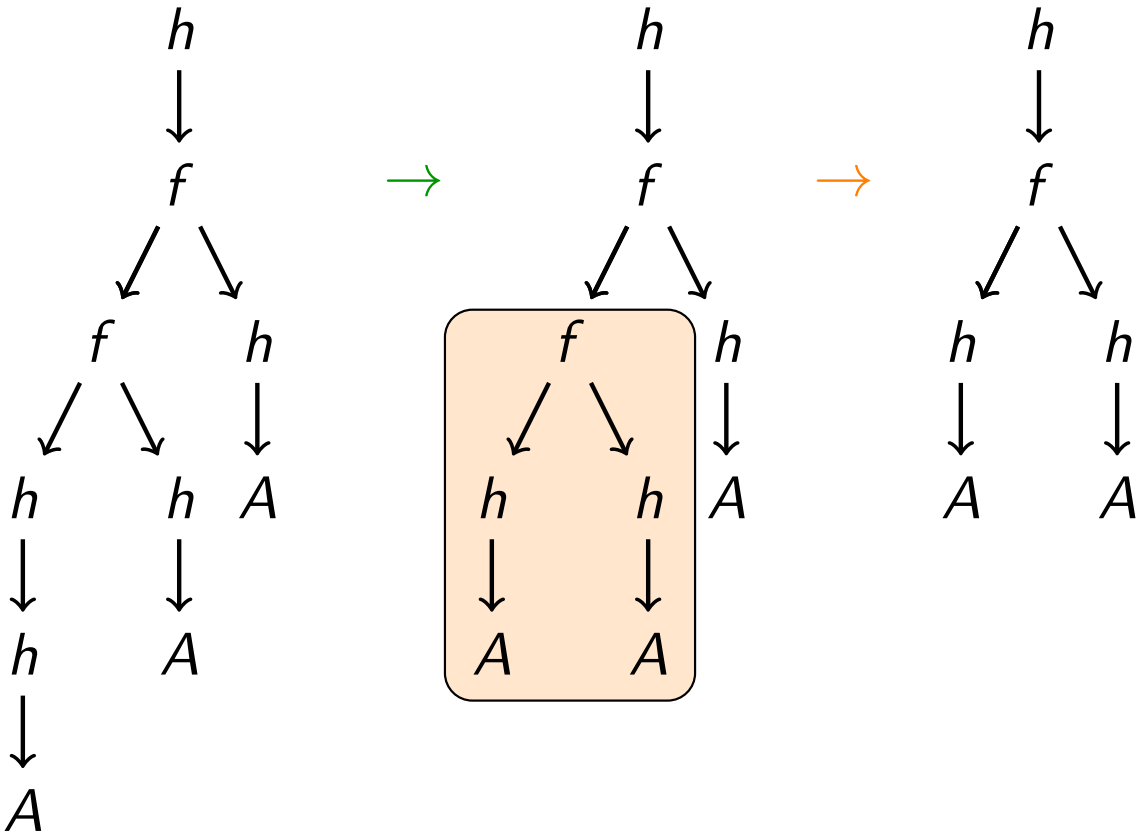
Soient $\mathcal{R} = \{h(h(x)) \to h(x),\ f(x,x)) \to x\}$ et t le terme si dessous.
Montrer que $t \to^*_{\mathcal{R}} h(A)$



Exercice

Exercice

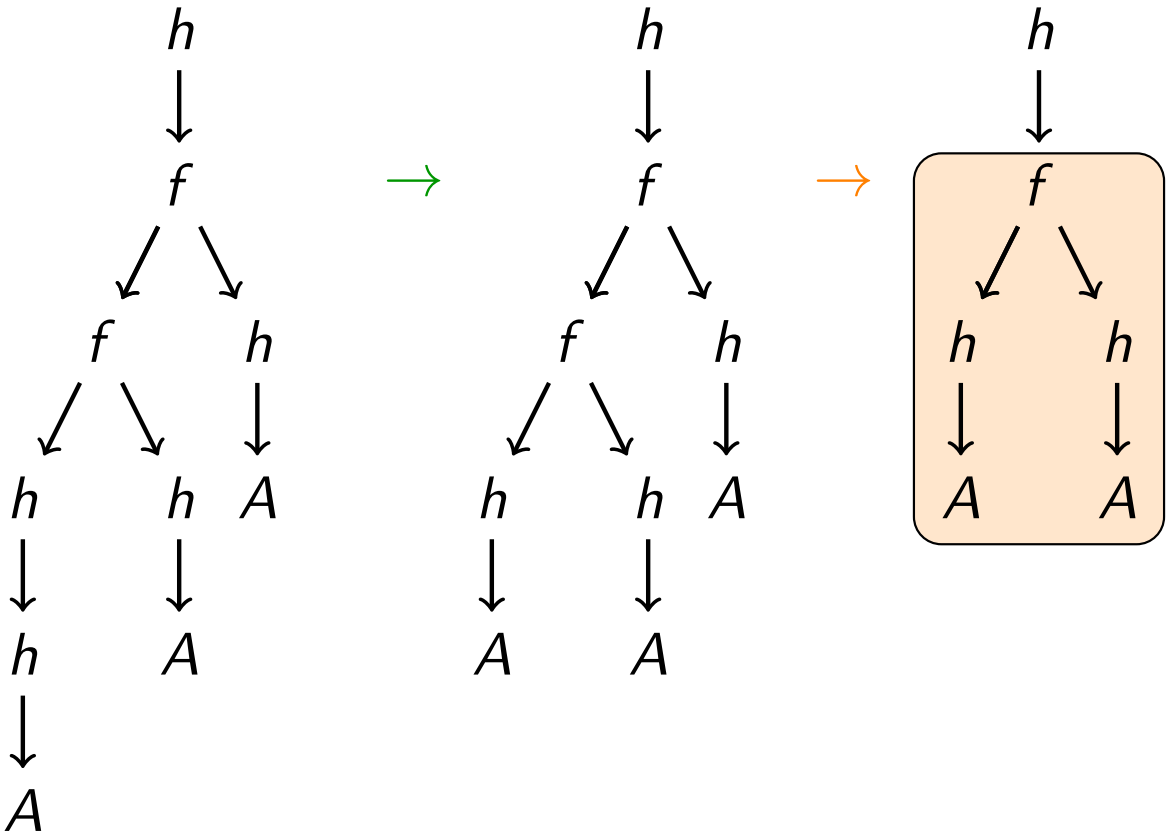
Soient $\mathcal{R} = \{h(h(x)) \to h(x),\ f(x,x)) \to x\}$ et t le terme si dessous.
Montrer que $t \to^*_{\mathcal{R}} h(A)$



Exercice

Exercice

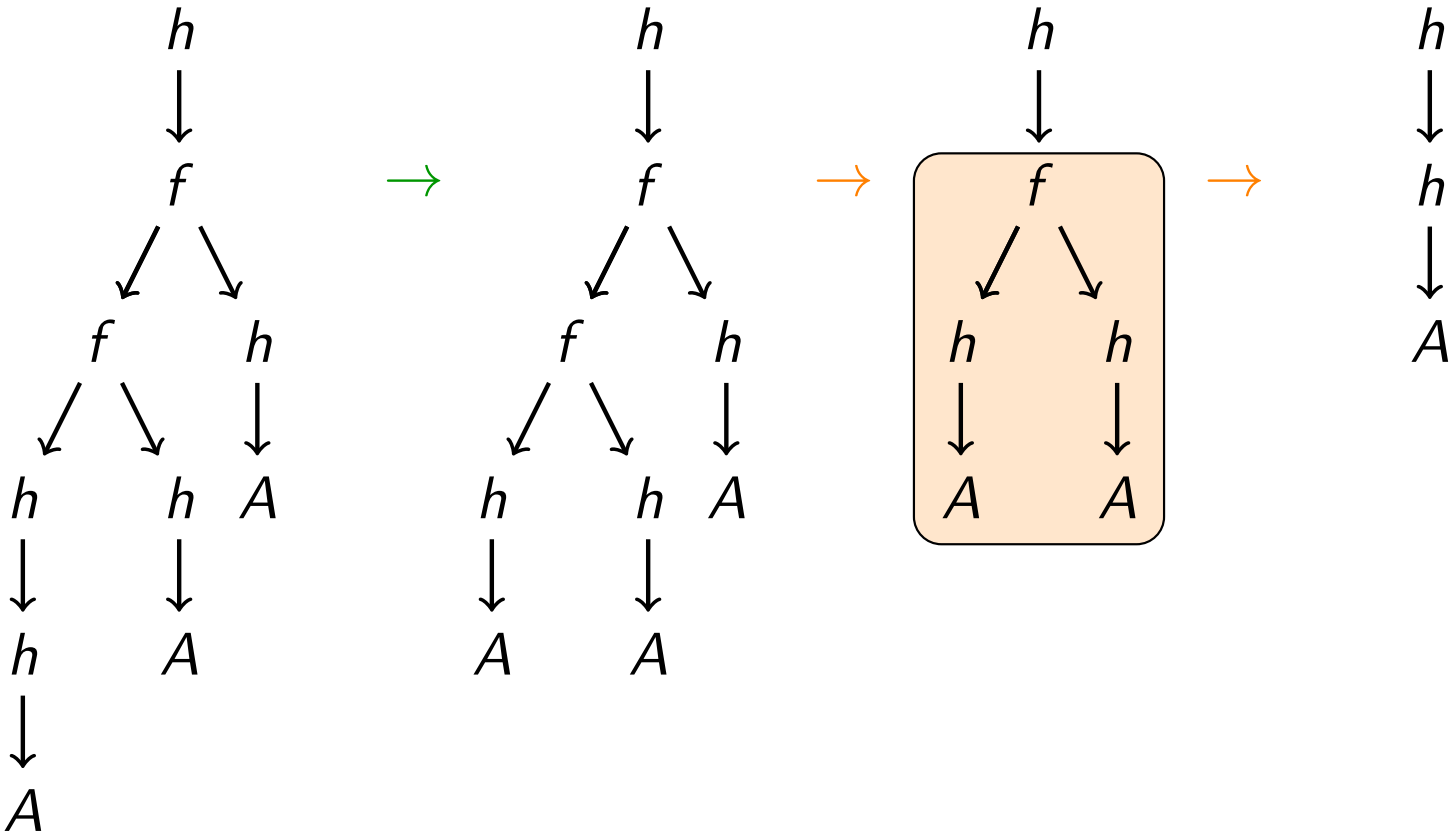
Soient $\mathcal{R} = \{h(h(x)) \to h(x),\ f(x,x)) \to x\}$ et t le terme si dessous.
Montrer que $t \to^*_{\mathcal{R}} h(A)$



Exercice

Exercice

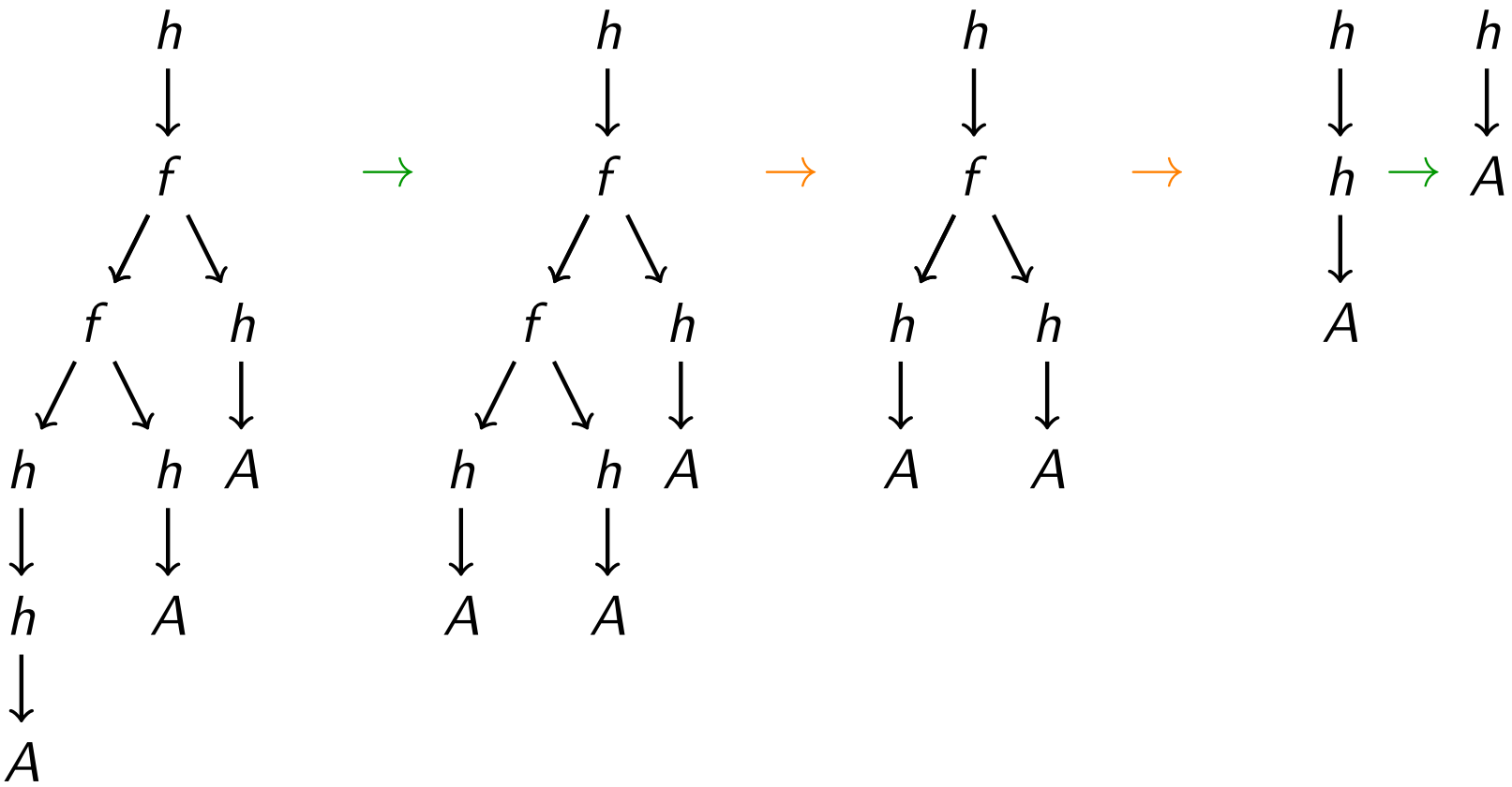
Soient $\mathcal{R} = \{h(h(x)) \to h(x),\ f(x,x)) \to x\}$ et t le terme si dessous.
Montrer que $t \to^*_{\mathcal{R}} h(A)$



Exercice

Exercice

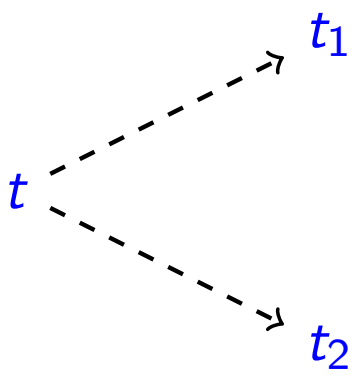
Soient $\mathcal{R} = \{h(h(x)) \to h(x),\ f(x,x)) \to x\}$ et t le terme si dessous.
Montrer que $t \to^*_{\mathcal{R}} h(A)$



Confluence

Définition *Confluent*

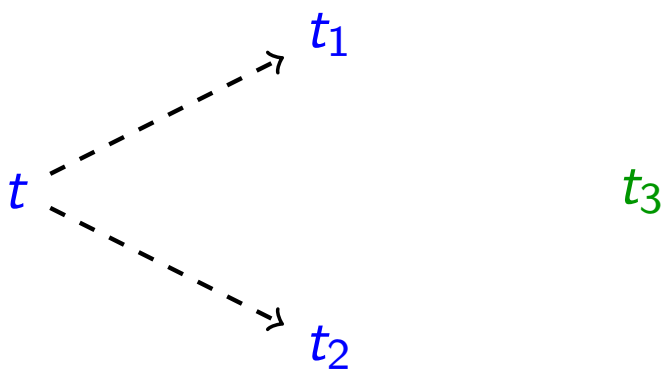
Un système de réécriture $\mathcal{R}$ est dit confluent si pour termes clos t, t_1 et t_2 tels que $t \to^*_{\mathcal{R}} t_1$ et $t \to^*_{\mathcal{R}} t_2$, il existe un terme clos t_3 tel que $t_1 \to^*_{\mathcal{R}} t_3$ et $t_2 \to^*_{\mathcal{R}} t_3$,



Confluence

Définition *Confluent*

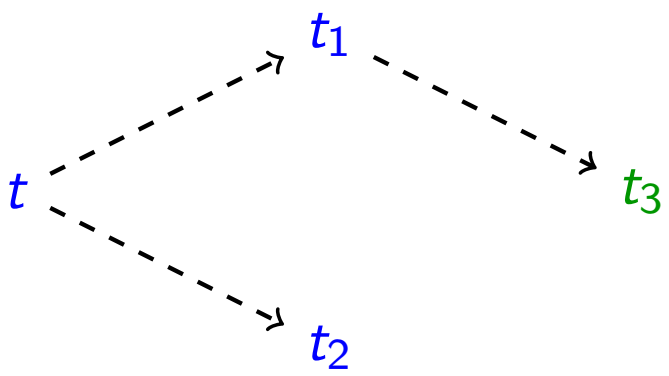
Un système de réécriture $\mathcal{R}$ est dit confluent si pour termes clos t, t_1 et t_2 tels que $t \to^*_{\mathcal{R}} t_1$ et $t \to^*_{\mathcal{R}} t_2$, il existe un terme clos t_3 tel que $t_1 \to^*_{\mathcal{R}} t_3$ et $t_2 \to^*_{\mathcal{R}} t_3$,



Confluence

Définition *Confluent*

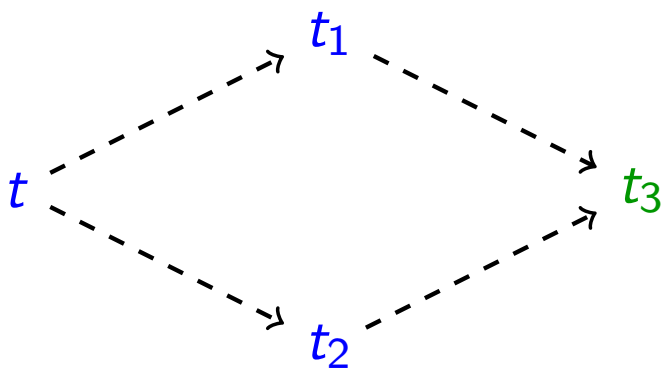
Un système de réécriture $\mathcal{R}$ est dit confluent si pour termes clos t, t_1 et t_2 tels que $t \to^*_{\mathcal{R}} t_1$ et $t \to^*_{\mathcal{R}} t_2$, il existe un terme clos t_3 tel que $t_1 \to^*_{\mathcal{R}} t_3$ et $t_2 \to^*_{\mathcal{R}} t_3$,



Confluence

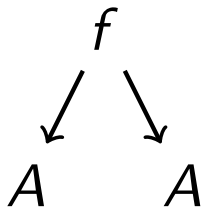
Définition *Confluent*

Un système de réécriture $\mathcal{R}$ est dit confluent si pour termes clos t, t_1 et t_2 tels que $t \to^*_{\mathcal{R}} t_1$ et $t \to^*_{\mathcal{R}} t_2$, il existe un terme clos t_3 tel que $t_1 \to^*_{\mathcal{R}} t_3$ et $t_2 \to^*_{\mathcal{R}} t_3$,



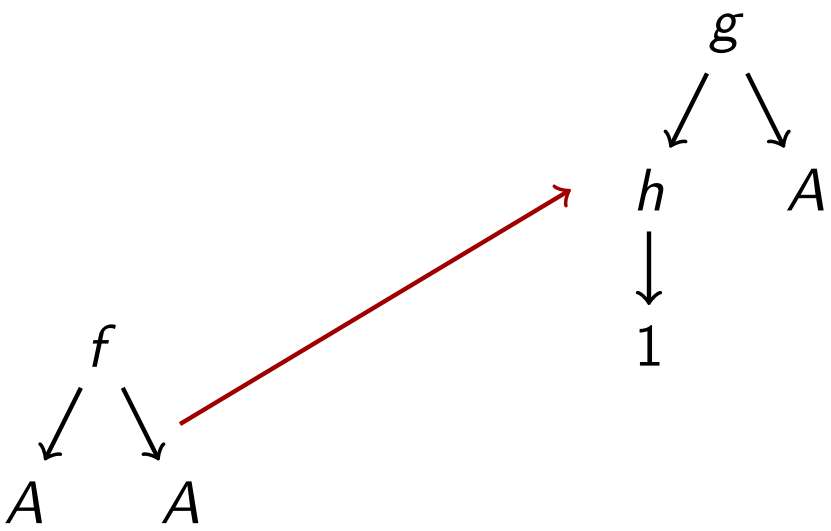
Exemple (non-confluence)

Le système de réécriture $\{f(x,y) \to g(h(x),y),\ f(x,y) \to g(x,h(y))\}$ n'est pas confluent.



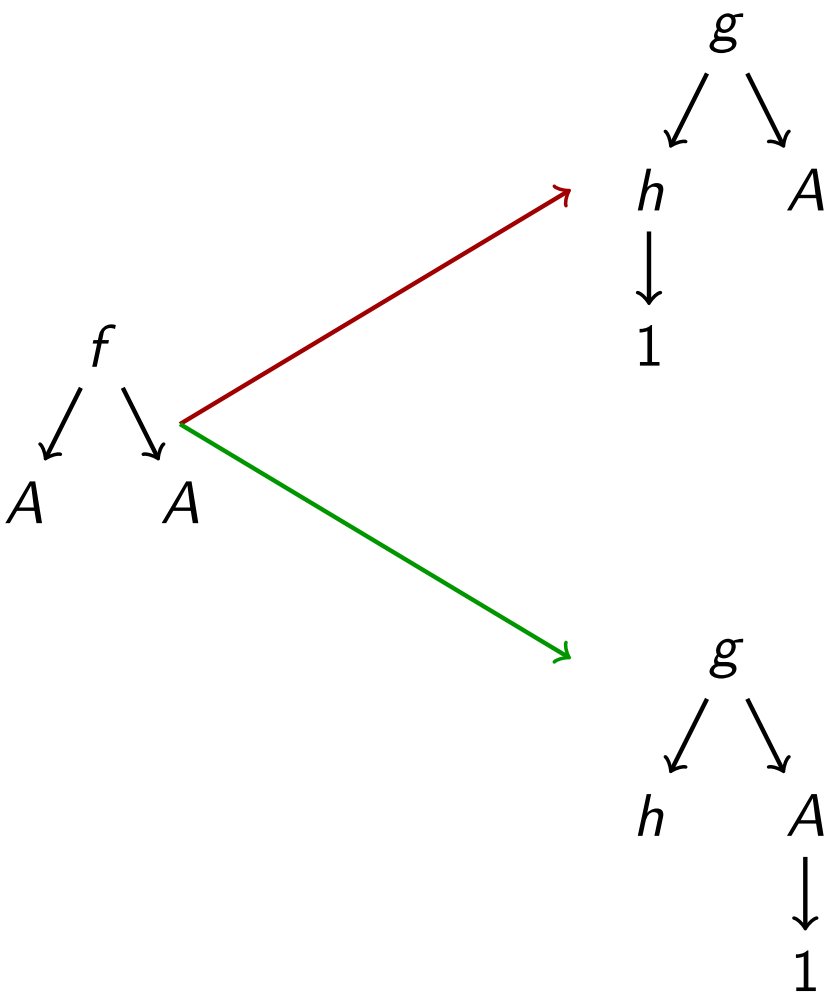
Exemple (non-confluence)

Le système de réécriture $\{f(x,y) \to g(h(x),y),\ f(x,y) \to g(x,h(y))\}$ n'est pas confluent.



Exemple (non-confluence)

Le système de réécriture $\{f(x,y) \to g(h(x),y),\ f(x,y) \to g(x,h(y))\}$ n'est pas confluent.



Exemple (confluence)

Le système de réécriture constitué des règles :

- $\neg(\wedge(x, y)) \rightarrow \vee(\neg(x), \neg(y))$
- $\neg(\vee(x, y)) \rightarrow \wedge(\neg(x), \neg(y))$
- $\neg(\neg(x)) \rightarrow x$

est confluent pour l'ensemble des formules propositionnelles sans $\Rightarrow$ ni $\Leftrightarrow$.

Exemple (confluence)

Le système de réécriture constitué des règles :

- $\neg(\wedge(x, y)) \rightarrow \vee(\neg(x), \neg(y))$
- $\neg(\vee(x, y)) \rightarrow \wedge(\neg(x), \neg(y))$
- $\neg(\neg(x)) \rightarrow x$

est confluent pour l'ensemble des formules propositionnelles sans $\Rightarrow$ ni $\Leftrightarrow$.

Remarque

Prouver la confluence d'un système de réécriture peut être difficile (propriété indécidable).

Terminaison

Définition

Un système de réécriture est dit terminant s'il n'existe pas de chaîne de réécriture infinie.

Terminaison

Définition

Un système de réécriture est dit terminant s'il n'existe pas de chaîne de réécriture infinie.

Le système $\{h(x) \to h(h(x)\}$ n'est par terminant.

Terminaison

Définition

Un système de réécriture est dit terminant s'il n'existe pas de chaîne de réécriture infinie.

Le système $\{h(x) \to h(h(x)\}$ n'est par terminant.

Le système $\{h(h(x)) \to h(x), f(h(x), y) \to h(y)\}$ est terminant (diminution du nombre de positions).

Terminaison

Définition

Un système de réécriture est dit terminant s'il n'existe pas de chaîne de réécriture infinie.

Le système $\{h(x) \to h(h(x)\}$ n'est par terminant.

Le système $\{h(h(x)) \to h(x), f(h(x), y) \to h(y)\}$ est terminant (diminution du nombre de positions).

Remarque

Prouver la terminaison d'un système de réécriture peut être difficile (propriété indécidable).

Forme Normale

Définition (irréductible)

Un terme t est irréductible pour un système de réécriture $\mathcal{R}$ s'il n'existe aucun terme t' tel que $t \rightarrow_{\mathcal{R}} t'$.

Le terme $f(A, B)$ est irréductible pour le système $\{f(x, x) \rightarrow h(x)\}$.

Forme Normale

Définition (irréductible)

Un terme t est irréductible pour un système de réécriture $\mathcal{R}$ s'il n'existe aucun terme t' tel que $t \to_{\mathcal{R}} t'$.

Le terme $f(A, B)$ est irréductible pour le système $\{f(x, x) \to h(x)\}$.

Définition (forme normale)

Une forme normale d'un terme t pour un système de réécriture $\mathcal{R}$ est terme t' **irréductible** tel que $t \to_{\mathcal{R}} t'$.

Le terme $h(A)$ est une forme normale de $h(h(f(A, A)))$ pour le système $\{f(x, x) \to h(x), h(h(x)) \to h(x)\}$.

Forme Normale

Définition (irréductible)

Un terme t est irréductible pour un système de réécriture $\mathcal{R}$ s'il n'existe aucun terme t' tel que $t \to_{\mathcal{R}} t'$.

Le terme $f(A, B)$ est irréductible pour le système $\{f(x, x) \to h(x)\}$.

Définition (forme normale)

Une forme normale d'un terme t pour un système de réécriture $\mathcal{R}$ est terme t' **irréductible** tel que $t \to_{\mathcal{R}} t'$.

Le terme $h(A)$ est une forme normale de $h(h(f(A, A)))$ pour le système $\{f(x, x) \to h(x), h(h(x)) \to h(x)\}$.

Un terme peut avoir aucune, une ou plusieurs forme normales.

Exercice

On considère le système contenant les trois règles de réécritures suivantes :

- $h(h(h(x) \rightarrow h(x)$
- $f(x, h(x)) \rightarrow f(x, x)$
- $g(x, y) \rightarrow h(g, (x, y))$

1. Donner deux termes irréductibles, dont un contenant le symbole f.
2. Donner un terme non irréductible n'ayant qu'une forme normale.
3. Donner un terme ayant plusieurs formes normales.
4. Donner un terme n'ayant aucune forme normale.

Exercice

On considère le système contenant les trois règles de réécritures suivantes :

- $h(h(h(x) \rightarrow h(x)$
- $f(x, h(x)) \rightarrow f(x, x)$
- $g(x, y) \rightarrow h(g, (x, y))$

1. Donner deux termes irréductibles, dont un contenant le symbole f.
   $f(A, B)$, $h(A)$ (par exemple)
2. Donner un terme non irréductible n'ayant qu'une forme normale.
3. Donner un terme ayant plusieurs formes normales.
4. Donner un terme n'ayant aucune forme normale.

Exercice

On considère le système contenant les trois règles de réécritures suivantes :

- $h(h(h(x) \rightarrow h(x)$
- $f(x, h(x)) \rightarrow f(x, x)$
- $g(x, y) \rightarrow h(g, (x, y))$

1. Donner deux termes irréductibles, dont un contenant le symbole f.
   $f(A, B)$, $h(A)$ (par exemple)
2. Donner un terme non irréductible n'ayant qu'une forme normale.
   $h(h(h(A)))$ (par exemple)
3. Donner un terme ayant plusieurs formes normales.

4. Donner un terme n'ayant aucune forme normale.

Exercice

On considère le système contenant les trois règles de réécritures suivantes :

- $h(h(h(x) \rightarrow h(x)$
- $f(x, h(x)) \rightarrow f(x, x)$
- $g(x, y) \rightarrow h(g, (x, y))$

1. Donner deux termes irréductibles, dont un contenant le symbole f.
   $f(A, B)$, $h(A)$ (par exemple)
2. Donner un terme non irréductible n'ayant qu'une forme normale.
   $h(h(h(A)))$ (par exemple)
3. Donner un terme ayant plusieurs formes normales.
   $f(h(h(A)), h(h(h(A))))$ qui a pour formes normales $f(h(h(A)), h(A))$ en utilisant la première règle, et $f(h(h(A)), h(h(A)))$ en utilisant la seconde règle.
4. Donner un terme n'ayant aucune forme normale.

Exercice

On considère le système contenant les trois règles de réécritures suivantes :

- $h(h(h(x) \rightarrow h(x)$
- $f(x, h(x)) \rightarrow f(x, x)$
- $g(x, y) \rightarrow h(g, (x, y))$

1. Donner deux termes irréductibles, dont un contenant le symbole f.
   $f(A, B)$, $h(A)$ (par exemple)
2. Donner un terme non irréductible n'ayant qu'une forme normale.
   $h(h(h(A)))$ (par exemple)
3. Donner un terme ayant plusieurs formes normales.
   $f(h(h(A)), h(h(h(A))))$ qui a pour formes normales $f(h(h(A)), h(A))$ en utilisant la première règle, et $f(h(h(A)), h(h(A)))$ en utilisant la seconde règle.
4. Donner un terme n'ayant aucune forme normale.
   $g(A, B)$ car $g(A, B) \rightarrow h(g(A, B)) \rightarrow h(h(g(A, B))) \rightarrow h(h(h(g(A, B)))) \rightarrow h(g(A, B))$.

Résultat important

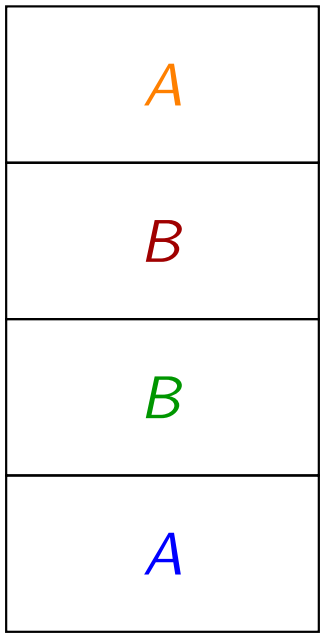
Théorème

Pour un système de réécriture à la fois **terminant** et **confluent**, tout terme possède une unique forme normale.

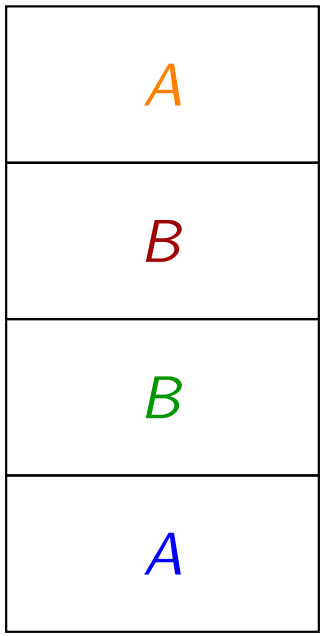
Le système de réécriture constitué des règles est confluent et terminant :

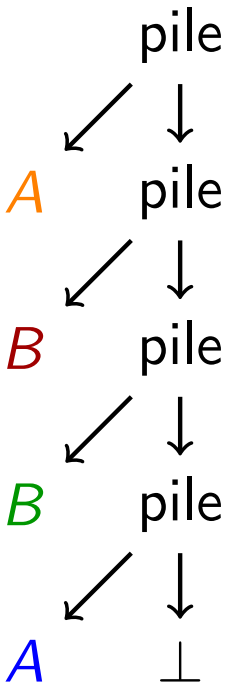
- $\neg(\wedge(x, y)) \rightarrow \vee(\neg(x), \neg(y))$
- $\neg(\vee(x, y)) \rightarrow \wedge(\neg(x), \neg(y))$
- $\neg(\neg(x)) \rightarrow x$
- $\Rightarrow (x, y) \rightarrow \wedge(\neg(x), y)$
- $\Leftrightarrow (x, y) \rightarrow \wedge(\Rightarrow (x, y), \Rightarrow (y, x))$

Exemple : les piles

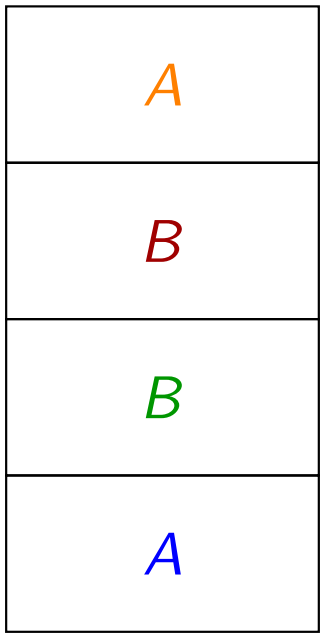


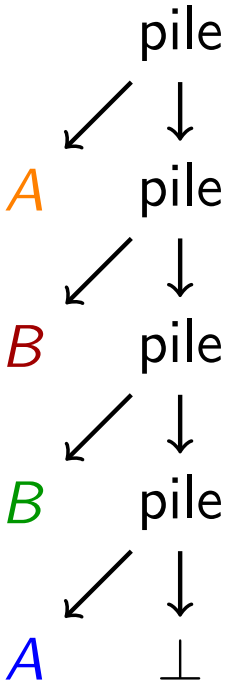
Exemple : les piles





Exemple : les piles





$$\text{pop}(\text{pile}(x, y)) \rightarrow y$$

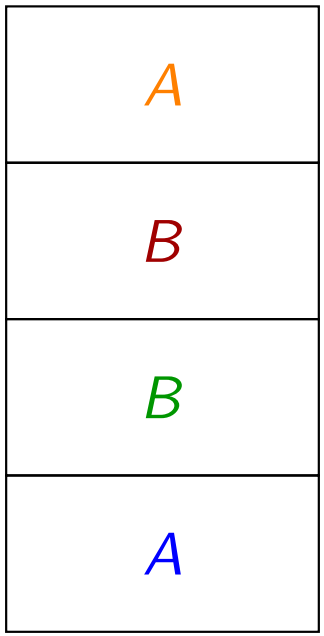
$$\text{push}(A, (\text{pile}(x, y))) \rightarrow \text{pile}(A, (\text{pile}(x, y)))$$

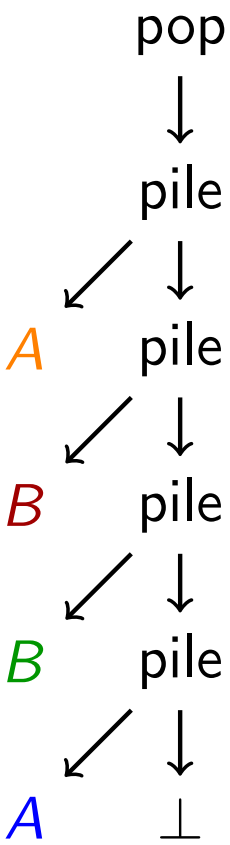
$$\text{push}(B, (\text{pile}(x, y))) \rightarrow \text{pile}(B, (\text{pile}(x, y)))$$

$$\text{push}(A, \bot) \rightarrow \text{pile}(A, \bot)$$

$$\text{push}(B, \bot) \rightarrow \text{pile}(B, \bot)$$

Exemple : les piles





$\text{pop}(\text{pile}(x, y)) \rightarrow y$

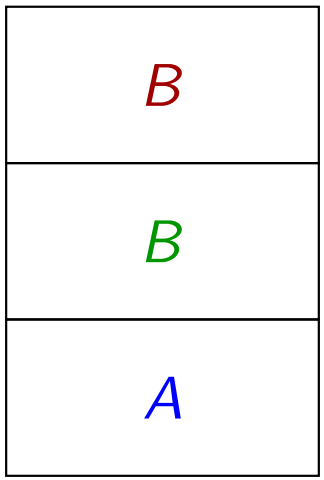
$\text{push}(A, (\text{pile}(x, y))) \rightarrow \text{pile}(A, (\text{pile}(x, y)))$

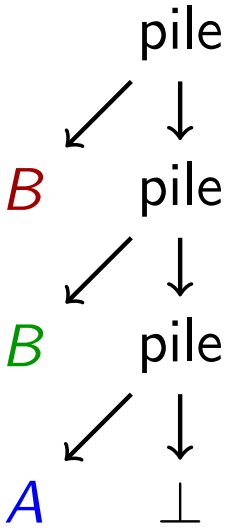
$\text{push}(B, (\text{pile}(x, y))) \rightarrow \text{pile}(B, (\text{pile}(x, y)))$

$\text{push}(A, \bot) \rightarrow \text{pile}(A, \bot)$

$\text{push}(B, \bot) \rightarrow \text{pile}(B, \bot)$

Exemple : les piles





$\text{pop}(\text{pile}(x, y)) \rightarrow y$

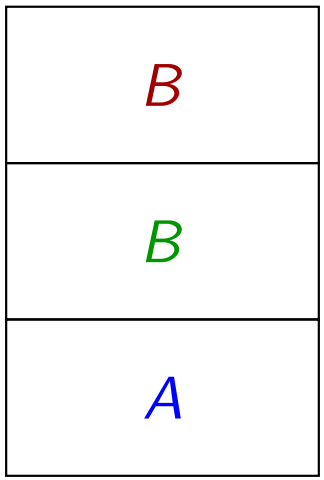
$\text{push}(A, (\text{pile}(x, y))) \rightarrow \text{pile}(A, (\text{pile}(x, y)))$

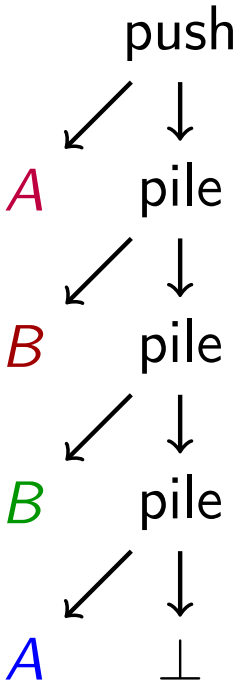
$\text{push}(B, (\text{pile}(x, y))) \rightarrow \text{pile}(B, (\text{pile}(x, y)))$

$\text{push}(A, \bot) \rightarrow \text{pile}(A, \bot)$

$\text{push}(B, \bot) \rightarrow \text{pile}(B, \bot)$

Exemple : les piles





$\text{pop}(\text{pile}(x, y)) \rightarrow y$

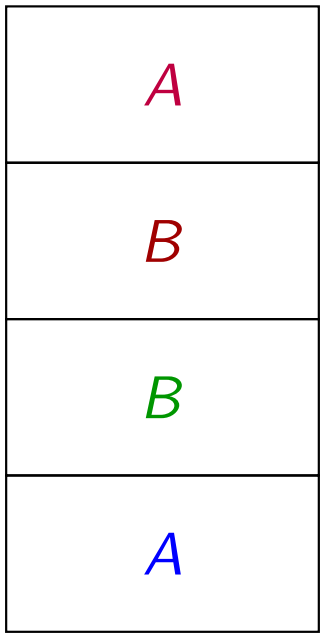
$\text{push}(A, (\text{pile}(x, y))) \rightarrow \text{pile}(A, (\text{pile}(x, y)))$

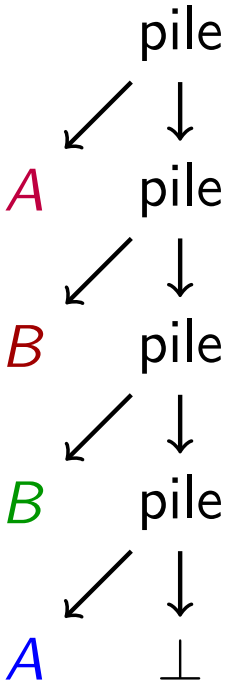
$\text{push}(B, (\text{pile}(x, y))) \rightarrow \text{pile}(B, (\text{pile}(x, y)))$

$\text{push}(A, \bot) \rightarrow \text{pile}(A, \bot)$

$\text{push}(B, \bot) \rightarrow \text{pile}(B, \bot)$

Exemple : les piles





$\text{pop}(\text{pile}(x, y)) \rightarrow y$

$\text{push}(A, (\text{pile}(x, y))) \rightarrow \text{pile}(A, (\text{pile}(x, y)))$

$\text{push}(B, (\text{pile}(x, y))) \rightarrow \text{pile}(B, (\text{pile}(x, y)))$

$\text{push}(A, \bot) \rightarrow \text{pile}(A, \bot)$

$\text{push}(B, \bot) \rightarrow \text{pile}(B, \bot)$

Conclusion

Les notions de **termes**, **arbres**, et la **réécritures** sont intensivement utilisés en informatique, parfois de façon implicite, par exemple pour :

- XSLT,
- Calcul formel (MAPPLE par exemple),
- Transformation de modèles,
- refactoring de codes,
- compilation
- langages fonctionnels (OCAML, Javascript,...)
- etc.