Logique pour l'informatique

Nils Gesbert

Ensimag 1A Alternance

Qu'est-ce que la logique ?

Qu'est-ce que la logique ?

La logique est l'étude des raisonnements (passage d'un ensemble d'hypothèses à une conclusion).

Sans logique, on peut quand même raisonner, mais comment savoir si on s'est trompé ?

Qu'est-ce que la logique ?

La logique est l'étude des raisonnements (passage d'un ensemble d'hypothèses à une conclusion).

Sans logique, on peut quand même raisonner, mais comment savoir si on s'est trompé ?

But : déterminer quand il est légitime de faire ce passage : notion de conséquence logique.

Remarque : Il ne s'agit pas de déterminer ce qui est vrai ou faux : on s'intéresse au raisonnement lui-même, non à ce sur quoi il raisonne. On fait abstraction de l'objet.

Raisonner sur les raisonnements : prérequis

Un langage formel pour éviter les ambiguïtés, limité pour éviter les paradoxes :

- « Cette phrase est fausse »
- « Le plus petit entier qui ne peut être défini en moins de quinze mots »

Raisonner sur les raisonnements : prérequis

Un langage formel pour éviter les ambiguïtés, limité pour éviter les paradoxes :

- « Cette phrase est fausse »
- « Le plus petit entier qui ne peut être défini en moins de quinze mots »

Nécessité de se placer à l'extérieur : métalangage, le langage qu'on utilise pour parler de la logique.

Suivant le langage choisi, différentes logiques... mais elles ont toutes en commun.

Approches sémantique et déductive

Comment définir la relation de conséquence logique ?

- Approche sémantique : idée de mondes possibles, de cas.
  B conséquence (sémantique) de A si dans tous les cas où A est vrai, B est vrai : « A ne peut être vrai sans que B le soit »

Approches sémantique et déductive

Comment définir la relation de conséquence logique ?

- Approche sémantique : idée de mondes possibles, de cas.
  B conséquence (sémantique) de A si dans tous les cas où A est vrai, B est vrai : « A ne peut être vrai sans que B le soit »
  Notation usuelle : $A \models B$

Approches sémantique et déductive

Comment définir la relation de conséquence logique ?

- Approche sémantique : idée de mondes possibles, de cas.
  B conséquence (sémantique) de A si dans tous les cas où A est vrai, B est vrai : « A ne peut être vrai sans que B le soit »
  Notation usuelle : $A \models B$
- Approche déductive : idée de règles d'inférence, de preuves formelles.
  B se déduit de A si on peut passer de A à B en suivant les règles

Approches sémantique et déductive

Comment définir la relation de conséquence logique ?

- Approche sémantique : idée de mondes possibles, de cas.
  B conséquence (sémantique) de A si dans tous les cas où A est vrai, B est vrai : « A ne peut être vrai sans que B le soit »
  Notation usuelle : $A \models B$
- Approche déductive : idée de règles d'inférence, de preuves formelles.
  B se déduit de A si on peut passer de A à B en suivant les règles
  Notation usuelle : $A \vdash B$

Approches sémantique et déductive

Comment définir la relation de conséquence logique ?

- Approche sémantique : idée de mondes possibles, de cas.
  B conséquence (sémantique) de A si dans tous les cas où A est vrai, B est vrai : « A ne peut être vrai sans que B le soit »
  Notation usuelle : $A \models B$
- Approche déductive : idée de règles d'inférence, de preuves formelles.
  B se déduit de A si on peut passer de A à B en suivant les règles
  Notation usuelle : $A \vdash B$

Idéalement les deux coïncident !
Procédure classique : on utilise des règles d'inférence dont on a déterminé qu'elles sont correctes pour prouver que B se déduit de A.
Si les règles sont effectivement correctes, B est alors une conséquence sémantique de A.

Un exemple historique : les syllogismes d'Aristote

Le langage comprend des phrases de la forme :

- Tout P est Q
- Aucun P n'est Q
- Certains P sont Q
- Certains P ne sont pas Q

Les raisonnements sont de la forme :
Si phrase(P,Q) et phrase(Q,R) alors phrase(P,R)

Exemple valide :

- Tous les hiboux voient la nuit,
- or tous les hiboux sont des oiseaux,
- donc certains oiseaux voient la nuit.

Exemple invalide :

- Certains animaux sont roses,
- or tous les éléphants sont des animaux,
- donc certains éléphants sont roses.

256 combinaisons possibles dont 24 valides.

Un exemple historique : les syllogismes d'Aristote

Le langage comprend des phrases de la forme :

- Tout P est Q
- Aucun P n'est Q
- Certains P sont Q
- Certains P ne sont pas Q

Les raisonnements sont de la forme :
Si phrase(P,Q) et phrase(Q,R) alors phrase(P,R)

Exemple valide :

- Tous les animaux sont roses,
- or tous les animaux sont des éléphants,
- donc certains éléphants sont roses.

Exemple invalide :

- Certains oiseaux voient la nuit,
- or tous les hiboux sont des oiseaux,
- donc certains hiboux voient la nuit.

256 combinaisons possibles dont 24 valides.

Différents langages logiques, différents degrés d'expressivité

- logique propositionnelle : propositions vraies ou fausses reliées par des connecteurs

Différents langages logiques, différents degrés d'expressivité

- logique propositionnelle : propositions vraies ou fausses reliées par des connecteurs

- logique du premier ordre : ajoute des prédicats reliant des variables, et les quantificateurs $\forall$ et $\exists$

Différents langages logiques, différents degrés d'expressivité

- logique propositionnelle : propositions vraies ou fausses reliées par des connecteurs
- logique du premier ordre : ajoute des prédicats reliant des variables, et les quantificateurs $\forall$ et $\exists$
- logique du second ordre : autorise à quantifier non seulement les variables mais aussi les prédicats

Logiques « non classiques »

Extensions des logiques classiques :

- Logiques multivaluées (avec valeurs intermédiaires entre vrai et faux, par exemple « inconnu »)
- Logiques modales (par exemple temporelles, « demain il fera beau ». Très utiles pour raisonner sur des graphes ou des arbres)

Restrictions :

- Logique « intuitionniste » : interdit de déduire A à partir de $\neg\neg A$.

Plan du cours

Introduction

Période 1 : Logique propositionnelle

- Syntaxe, sémantique
- Systèmes formels
- Le problème SAT

Période 3 : Logique du premier ordre

- Syntaxe et sémantique
- Preuves et théories
- Satisfaisabilité

Langage de la logique propositionnelle

- Des propositions atomiques
  - Représentent des faits qui peuvent être vrais ou faux
  - « atomiques » : on ne regarde pas leur contenu
  - Désignées par des lettres $P, Q, \ldots$ (« symboles propositionnels »)

- Des connecteurs logiques
  - Conjonction : $\wedge$
  - Disjonction : $\vee$
  - Négation : $\neg$
  - Implication : $\Rightarrow$
  - Équivalence : $\Leftrightarrow$
  - Contradiction (toujours faux) : $\bot$; toujours vrai : $\top$

- Des règles de bonne formation des formules
  - Respect de l'arité des opérateurs (binaires, unaire)

Tables de vérité

On attribue à une formule une valeur de vérité en fonction de celles des propositions atomiques qu'elle contient.

L'ensemble des possibilités peut être représenté
dans une table de vérité :

P	Q	R	$\varphi = (P \wedge Q) \vee R$
V	V	V	V
V	V	F	V
V	F	V	V
V	F	F	F
F	V	V	V
F	V	F	F
F	F	V	V
F	F	F	F

Tables de vérité

On attribue à une formule une valeur de vérité en fonction de celles des propositions atomiques qu'elle contient.

L'ensemble des possibilités peut être représenté dans une table de vérité :

	P	Q	R	$\varphi = (P \wedge Q) \vee R$
	V	V	V	V
	V	V	F	V
interprétation $\rightarrow$	V	F	V	V
	V	F	F	F
	F	V	V	V
	F	V	F	F
	F	F	V	V
	F	F	F	F

Tables de vérité

On attribue à une formule une valeur de vérité en fonction de celles des propositions atomiques qu'elle contient.

L'ensemble des possibilités peut être représenté dans une table de vérité :

interprétation $\rightarrow$
$\mathrm{I} : \{P, Q, R\} \rightarrow \{\mathrm{V}, \mathrm{F}\}$

P	Q	R	$\varphi = (P \wedge Q) \vee R$
V	V	V	V
V	V	F	V
V	F	V	V
V	F	F	F
F	V	V	V
F	V	F	F
F	F	V	V
F	F	F	F

Tables de vérité

On attribue à une formule une valeur de vérité en fonction de celles des propositions atomiques qu'elle contient.

L'ensemble des possibilités peut être représenté dans une table de vérité :

interprétation $\rightarrow$
$\mathrm{I} : \{P, Q, R\} \rightarrow \{\mathrm{V}, \mathrm{F}\}$

P	Q	R	$\varphi = (P \wedge Q) \vee R$
V	V	V	V
V	V	F	V
V	F	V	V
V	F	F	F
F	V	V	V
F	V	F	F
F	F	V	V
F	F	F	F

$\leftarrow$ évaluation

Tables de vérité

On attribue à une formule une valeur de vérité en fonction de celles des propositions atomiques qu'elle contient.

L'ensemble des possibilités peut être représenté dans une table de vérité :

interprétation $\rightarrow$
$\mathrm{I} : \{P, Q, R\} \rightarrow \{\mathrm{V}, \mathrm{F}\}$

P	Q	R	$\varphi = (P \wedge Q) \vee R$
V	V	V	V
V	V	F	V
V	F	V	V
V	F	F	F
F	V	V	V
F	V	F	F
F	F	V	V
F	F	F	F

$\leftarrow$ évaluation $\mathcal{E}(\varphi, \mathrm{I})$

La fonction d'évaluation

Une fonction de vérité n-aire est une fonction : $\{\mathrm{V}, \mathrm{F}\}^n \to \{\mathrm{V}, \mathrm{F}\}$.
À chaque connecteur logique correspond une fonction de vérité :

φ	ψ	$\neg\varphi$	$\varphi \wedge \psi$	$\varphi \vee \psi$	$\varphi \Rightarrow \psi$	$\varphi \Leftrightarrow \psi$
V	V	F	V	V	V	V
V	F	F	F	V	F	F
F	V	V	F	V	V	F
F	F	V	F	F	V	V

Soit I une interprétation, c.-à-d. une fonction qui à toute prop. atomique associe une valeur de vérité. $\mathcal{E}(\varphi, \mathrm{I})$ est le résultat obtenu en remplaçant les prop. par leur valeur et en appliquant les fonctions des connecteurs.

Quelques définitions supplémentaires

Soient φ une formule et I une interprétation.

- Si $\mathcal{E}(\varphi, \mathrm{I}) = \mathrm{V}$ on dit que I satisfait φ, ou est un modèle de φ. Cela se note : $\mathrm{I} \models \varphi$
- Si $\mathcal{E}(\varphi, \mathrm{I}) = \mathrm{F}$ on dit que I falsifie φ, ou est un contre-modèle ou contre-exemple de φ.
- φ est dite satisfaisable si elle a au moins un modèle. Elle est dite falsifiable si elle a au moins un contre-modèle.

Soient φ et ψ deux formules, on dit que ψ est une conséquence logique de φ si tout modèle de φ est un modèle de ψ. Cela se note : $\varphi \models \psi$.

Une formule φ est valide si toute interprétation la satisfait. On dit que φ est une tautologie et cela se note : $\models \varphi$.

Une formule insatisfaisable est aussi dite contradictoire.

Quelques définitions supplémentaires

Soient φ une formule et I une interprétation.

- Si $\mathcal{E}(\varphi, \mathrm{I}) = \mathrm{V}$ on dit que I satisfait φ, ou est un modèle de φ. Cela se note : $\mathrm{I} \models \varphi$
- Si $\mathcal{E}(\varphi, \mathrm{I}) = \mathrm{F}$ on dit que I falsifie φ, ou est un contre-modèle ou contre-exemple de φ.
- φ est dite satisfaisable si elle a au moins un modèle. Elle est dite falsifiable si elle a au moins un contre-modèle.

Soient φ et ψ deux formules, on dit que ψ est une conséquence logique de φ si tout modèle de φ est un modèle de ψ. Cela se note : $\varphi \models \psi$.

Une formule φ est valide si toute interprétation la satisfait. On dit que φ est une tautologie et cela se note : $\models \varphi$.

Une formule insatisfaisable est aussi dite contradictoire.
Propriété : φ est valide si et seulement si $\neg\varphi$ est contradictoire.

Règles d'inférence

On a défini la correction d'un raisonnement en termes sémantiques (tout modèle des hypothèses est un modèle de la conclusion). Mais quelles sont les règles qui nous permettent de déduire la conclusion des hypothèses ?

Règles d'inférence

On a défini la correction d'un raisonnement en termes sémantiques (tout modèle des hypothèses est un modèle de la conclusion). Mais quelles sont les règles qui nous permettent de déduire la conclusion des hypothèses ?

Idée : on construit un raisonnement en combinant des règles élémentaires de déduction dont on sait qu'elles sont correctes.
Exemples :

$$\frac{\varphi \qquad \psi}{\varphi \wedge \psi} \qquad \frac{\varphi \wedge \psi}{\varphi} \qquad \frac{\varphi}{\varphi \vee \psi} \qquad \frac{\varphi \Rightarrow \psi \qquad \psi \Rightarrow \chi}{\varphi \Rightarrow \chi}$$

Règles d'inférence

On a défini la correction d'un raisonnement en termes sémantiques (tout modèle des hypothèses est un modèle de la conclusion). Mais quelles sont les règles qui nous permettent de déduire la conclusion des hypothèses ?

Idée : on construit un raisonnement en combinant des règles élémentaires de déduction dont on sait qu'elles sont correctes.
Exemples :

$$\frac{\varphi \qquad \psi}{\varphi \wedge \psi} \qquad \frac{\varphi \wedge \psi}{\varphi} \qquad \frac{\varphi}{\varphi \vee \psi} \qquad \frac{\varphi \Rightarrow \psi \qquad \psi \Rightarrow \chi}{\varphi \Rightarrow \chi}$$

Exemple plus sophistiqué (hors de la LP) :

$$\frac{\mathcal{P}(0) \qquad \forall n \in \mathbb{N}, \mathcal{P}(n) \Rightarrow \mathcal{P}(n+1)}{\forall n \in \mathbb{N}, \mathcal{P}(n)}$$

Axiomes, schémas, théorèmes

Les règles d'inférence permettent de tirer des conclusions à partir d'hypothèses. Si l'on veut démontrer des formules qui sont toujours vraies (sans hypothèse), la solution la plus simple est d'ajouter un point de départ : les axiomes.

Exemples : $\varphi \Rightarrow \varphi$, $\varphi \vee \neg\varphi$

Axiomes, schémas, théorèmes

Les règles d'inférence permettent de tirer des conclusions à partir d'hypothèses. Si l'on veut démontrer des formules qui sont toujours vraies (sans hypothèse), la solution la plus simple est d'ajouter un point de départ : les axiomes.
Exemples : $\varphi \Rightarrow \varphi$, $\varphi \vee \neg\varphi$

Ici φ peut être remplacé par n'importe quelle formule, on parle donc de schéma d'axiomes. φ est une (méta-)variable.
Une instance du schéma est obtenue en attribuant une formule à chaque variable et en remplaçant toutes les occurrences de la variable par cette même formule.
Par exemple : $(P \Rightarrow Q) \Rightarrow (P \Rightarrow Q)$ est une instance de $\varphi \Rightarrow \varphi$.

Axiomes, schémas, théorèmes

Les règles d'inférence permettent de tirer des conclusions à partir d'hypothèses. Si l'on veut démontrer des formules qui sont toujours vraies (sans hypothèse), la solution la plus simple est d'ajouter un point de départ : les axiomes.
Exemples : $\varphi \Rightarrow \varphi$, $\varphi \vee \neg\varphi$

Ici φ peut être remplacé par n'importe quelle formule, on parle donc de schéma d'axiomes. φ est une (méta-)variable.
Une instance du schéma est obtenue en attribuant une formule à chaque variable et en remplaçant toutes les occurrences de la variable par cette même formule.
Par exemple : $(P \Rightarrow Q) \Rightarrow (P \Rightarrow Q)$ est une instance de $\varphi \Rightarrow \varphi$.
De même pour les règles d'inférence :
$\dfrac{P \Rightarrow Q \qquad R}{(P \Rightarrow Q) \wedge R}$ est une instance de $\dfrac{\varphi \qquad \psi}{\varphi \wedge \psi}$.

Axiomes, schémas, théorèmes

Les règles d'inférence permettent de tirer des conclusions à partir d'hypothèses. Si l'on veut démontrer des formules qui sont toujours vraies (sans hypothèse), la solution la plus simple est d'ajouter un point de départ : les axiomes.
Exemples : $\varphi \Rightarrow \varphi$, $\varphi \vee \neg\varphi$

Ici φ peut être remplacé par n'importe quelle formule, on parle donc de schéma d'axiomes. φ est une (méta-)variable.
Une instance du schéma est obtenue en attribuant une formule à chaque variable et en remplaçant toutes les occurrences de la variable par cette même formule.
Par exemple : $(P \Rightarrow Q) \Rightarrow (P \Rightarrow Q)$ est une instance de $\varphi \Rightarrow \varphi$.
De même pour les règles d'inférence :
$\dfrac{P \Rightarrow Q \qquad R}{(P \Rightarrow Q) \wedge R}$ est une instance de $\dfrac{\varphi \qquad \psi}{\varphi \wedge \psi}$.

Les formules que l'on peut déduire des axiomes en utilisant les règles d'inférence sont appelées théorèmes.

Systèmes formels

Un système formel simple est la donnée :

- d'un langage formel $\mathcal{L}$ définissant l'ensemble des formules ;
- d'un ensemble $\mathcal{A}$ de formules du langage qu'on admet, appelées axiomes ;
- d'un ensemble $\mathcal{R}$ de règles d'inférence permettant de déduire d'un nombre fini de formules appelées prémisses une formule appelée conclusion.

Systèmes formels

Un système formel simple est la donnée :

- d'un langage formel $\mathcal{L}$ définissant l'ensemble des formules ;
- d'un ensemble $\mathcal{A}$ de formules du langage qu'on admet, appelées axiomes ;
- d'un ensemble $\mathcal{R}$ de règles d'inférence permettant de déduire d'un nombre fini de formules appelées prémisses une formule appelée conclusion.

- Le langage formel est défini par un nombre fini de règles syntaxiques ;
- Les axiomes et les règles d'inférences sont définis par un nombre fini de schémas (mais ont généralement une infinité d'instances).

Preuve dans un système formel

On dit qu'une règle d'inférence R s'applique à un ensemble F de formules s'il existe une instance de R dont les prémisses sont les formules de F.

Preuve dans un système formel

On dit qu'une règle d'inférence R s'applique à un ensemble F de formules s'il existe une instance de R dont les prémisses sont les formules de F.
Le résultat de l'application de R à F est la conclusion de l'instance de R en question.

Preuve dans un système formel

On dit qu'une règle d'inférence R s'applique à un ensemble F de formules s'il existe une instance de R dont les prémisses sont les formules de F.
Le résultat de l'application de R à F est la conclusion de l'instance de R en question.

Exemple : si $R = \frac{\varphi \Rightarrow \psi \quad \varphi}{\psi}$ et $F = \{P \Leftrightarrow Q, (P \Leftrightarrow Q) \Rightarrow (Q \Rightarrow P)\}$, la règle R s'applique à F, et le résultat de cette application est la formule $Q \Rightarrow P$.

Preuve dans un système formel

Une déduction dans un système formel, à partir d'un ensemble de formules appelées hypothèses, est une suite d'étapes de raisonnement, chaque étape étant :

Preuve dans un système formel

Une déduction dans un système formel, à partir d'un ensemble de formules appelées hypothèses, est une suite d'étapes de raisonnement, chaque étape étant :

- soit une des hypothèses,

Preuve dans un système formel

Une déduction dans un système formel, à partir d'un ensemble de formules appelées hypothèses, est une suite d'étapes de raisonnement, chaque étape étant :

- soit une des hypothèses,
- soit une instance d'un axiome du système,

Preuve dans un système formel

Une déduction dans un système formel, à partir d'un ensemble de formules appelées hypothèses, est une suite d'étapes de raisonnement, chaque étape étant :

- soit une des hypothèses,
- soit une instance d'un axiome du système,
- soit le résultat de l'application d'une règle à des formules obtenues lors d'étapes précédentes.

Preuve dans un système formel

Une déduction dans un système formel, à partir d'un ensemble de formules appelées hypothèses, est une suite d'étapes de raisonnement, chaque étape étant :

- soit une des hypothèses,
- soit une instance d'un axiome du système,
- soit le résultat de l'application d'une règle à des formules obtenues lors d'étapes précédentes.

La conclusion de la déduction est le résultat de la dernière étape.

Preuve dans un système formel : exemple

Exemple de déduction : soit le système formel défini par :

Preuve dans un système formel : exemple

Exemple de déduction : soit le système formel défini par :

- le langage de la logique propositionnelle

Preuve dans un système formel : exemple

Exemple de déduction : soit le système formel défini par :

- le langage de la logique propositionnelle
- l'axiome $A = (\varphi \wedge \psi) \Rightarrow (\varphi \vee \psi)$

Preuve dans un système formel : exemple

Exemple de déduction : soit le système formel défini par :

- le langage de la logique propositionnelle
- l'axiome $A = (\varphi \wedge \psi) \Rightarrow (\varphi \vee \psi)$
- les règles d'inférence $R_1 = \frac{\varphi \quad \psi}{\varphi \wedge \psi}$ et $R_2 = \frac{\varphi \quad \varphi \Rightarrow \psi}{\psi}$

Preuve dans un système formel : exemple

Exemple de déduction : soit le système formel défini par :

- le langage de la logique propositionnelle
- l'axiome $A = (\varphi \wedge \psi) \Rightarrow (\varphi \vee \psi)$
- les règles d'inférence $R_1 = \frac{\varphi \quad \psi}{\varphi \wedge \psi}$ et $R_2 = \frac{\varphi \quad \varphi \Rightarrow \psi}{\psi}$

En partant des hypothèses P et $\neg Q$, on peut obtenir la déduction suivante :

1. P hypothèse
2. $\neg Q$ hypothèse

Preuve dans un système formel : exemple

Exemple de déduction : soit le système formel défini par :

- le langage de la logique propositionnelle
- l'axiome $A = (\varphi \wedge \psi) \Rightarrow (\varphi \vee \psi)$
- les règles d'inférence $R_1 = \frac{\varphi \quad \psi}{\varphi \wedge \psi}$ et $R_2 = \frac{\varphi \quad \varphi \Rightarrow \psi}{\psi}$

En partant des hypothèses P et $\neg Q$, on peut obtenir la déduction suivante :

1. P — hypothèse
2. $\neg Q$ — hypothèse
3. $P \wedge \neg Q$ — appl. de R_1 à 1 et 2

Preuve dans un système formel : exemple

Exemple de déduction : soit le système formel défini par :

- le langage de la logique propositionnelle
- l'axiome $A = (\varphi \wedge \psi) \Rightarrow (\varphi \vee \psi)$
- les règles d'inférence $R_1 = \frac{\varphi \quad \psi}{\varphi \wedge \psi}$ et $R_2 = \frac{\varphi \quad \varphi \Rightarrow \psi}{\psi}$

En partant des hypothèses P et $\neg Q$, on peut obtenir la déduction suivante :

1. P — hypothèse
2. $\neg Q$ — hypothèse
3. $P \wedge \neg Q$ — appl. de R_1 à 1 et 2
4. $(P \wedge \neg Q) \Rightarrow (P \vee \neg Q)$ — instance de A

Preuve dans un système formel : exemple

Exemple de déduction : soit le système formel défini par :

- le langage de la logique propositionnelle
- l'axiome $A = (\varphi \wedge \psi) \Rightarrow (\varphi \vee \psi)$
- les règles d'inférence $R_1 = \frac{\varphi \quad \psi}{\varphi \wedge \psi}$ et $R_2 = \frac{\varphi \quad \varphi \Rightarrow \psi}{\psi}$

En partant des hypothèses P et $\neg Q$, on peut obtenir la déduction suivante :

1. P — hypothèse
2. $\neg Q$ — hypothèse
3. $P \wedge \neg Q$ — appl. de R_1 à 1 et 2
4. $(P \wedge \neg Q) \Rightarrow (P \vee \neg Q)$ — instance de A
5. $P \vee \neg Q$ — appl. de R_2 à 3 et 4

Preuve dans un système formel : notations

On écrit $\Gamma \vdash_{\mathcal{S}} \chi$ s'il existe une déduction allant de Γ à χ dans le système $\mathcal{S}$.

Preuve dans un système formel : notations

On écrit $\Gamma \vdash_{\mathcal{S}} \chi$ s'il existe une déduction allant de Γ à χ dans le système $\mathcal{S}$.

Une démonstration dans un système formel simple est une déduction sans hypothèse (elle ne part donc que des axiomes).

Un théorème de $\mathcal{S}$ est une formule $\mathcal{T}$ telle que $\vdash_{\mathcal{S}} \mathcal{T}$.

Preuve dans un système formel : notations

On écrit $\Gamma \vdash_{\mathcal{S}} \chi$ s'il existe une déduction allant de Γ à χ dans le système $\mathcal{S}$.

Une démonstration dans un système formel simple est une déduction sans hypothèse (elle ne part donc que des axiomes).

Un théorème de $\mathcal{S}$ est une formule $\mathcal{T}$ telle que $\vdash_{\mathcal{S}} \mathcal{T}$.

Propriété : Toute formule déduite de théorèmes est un théorème.

Consistance d'un système formel

Remarque : La définition d'un système formel ne nécessite pas de sémantique. $\vdash$ définit une relation de conséquence purement syntaxique.

Consistance d'un système formel

Remarque : La définition d'un système formel ne nécessite pas de sémantique. $\vdash$ définit une relation de conséquence purement syntaxique.

Les deux propriétés suivantes peuvent également être définies sans sémantique :

- Un système est dit consistant ou cohérent s'il ne permet jamais de prouver à la fois une formule et sa négation.
- Il est dit absolument consistant s'il existe au moins une formule du langage qui n'est pas un théorème.

Pour les systèmes utilisant la logique classique, les deux sont équivalents (d'une contradiction on peut déduire n'importe quoi).

Consistance d'un système formel

Remarque : La définition d'un système formel ne nécessite pas de sémantique. $\vdash$ définit une relation de conséquence purement syntaxique.

Les deux propriétés suivantes peuvent également être définies sans sémantique :

- Un système est dit consistant ou cohérent s'il ne permet jamais de prouver à la fois une formule et sa négation.
- Il est dit absolument consistant s'il existe au moins une formule du langage qui n'est pas un théorème.

Pour les systèmes utilisant la logique classique, les deux sont équivalents (d'une contradiction on peut déduire n'importe quoi).

Propriété fondamentale... mais pas toujours facile d'être sûr.
Schéma d'axiomes de compréhension naïf en théorie des ensembles :
$\exists E, \forall x, (x \in E \Leftrightarrow \mathcal{P}(x))$ (affirme l'existence de l'ens. $\{x \mid \mathcal{P}(x)\}$)

Consistance d'un système formel

Remarque : La définition d'un système formel ne nécessite pas de sémantique. $\vdash$ définit une relation de conséquence purement syntaxique.

Les deux propriétés suivantes peuvent également être définies sans sémantique :

- Un système est dit consistant ou cohérent s'il ne permet jamais de prouver à la fois une formule et sa négation.
- Il est dit absolument consistant s'il existe au moins une formule du langage qui n'est pas un théorème.

Pour les systèmes utilisant la logique classique, les deux sont équivalents (d'une contradiction on peut déduire n'importe quoi).

Propriété fondamentale... mais pas toujours facile d'être sûr.
Schéma d'axiomes de compréhension naïf en théorie des ensembles :
$\exists E, \forall x, (x \in E \Leftrightarrow \mathcal{P}(x))$ (affirme l'existence de l'ens. $\{x \mid \mathcal{P}(x)\}$)
Autorise à instancier $\mathcal{P}(x) \leftarrow x \notin x$ et à choisir $x = E$ pour déduire $E \in E \Leftrightarrow E \notin E$ (paradoxe de Russell).

Exemples de systèmes formels

Souvent on ne spécifie pas les règles d'inférence (on utilise celles de la logique classique) : on se contente d'un système axiomatique.
Exemple historique : les *Éléments* d'Euclide ($\sim$ 300 av. J-C).

Exemples de systèmes formels

Souvent on ne spécifie pas les règles d'inférence (on utilise celles de la logique classique) : on se contente d'un système axiomatique.
Exemple historique : les *Éléments* d'Euclide ($\sim$ 300 av. J-C).
Axiomes mais aussi « postulats » considérés comme moins évidents, surtout le cinquième postulat, reformulé plus tard en :

Par un point extérieur à une droite passe une unique parallèle à cette droite.

Question : peut-on démontrer ce postulat ?

Exemples de systèmes formels

Souvent on ne spécifie pas les règles d'inférence (on utilise celles de la logique classique) : on se contente d'un système axiomatique.
Exemple historique : les *Éléments* d'Euclide ($\sim$ 300 av. J-C).
Axiomes mais aussi « postulats » considérés comme moins évidents, surtout le cinquième postulat, reformulé plus tard en :

Par un point extérieur à une droite passe une unique parallèle à cette droite.

Question : peut-on démontrer ce postulat ?
De nombreux mathématiciens ont essayé, notamment par l'absurde.
En supposant le postulat faux, on arrive à des résultats « manifestement faux » (triangles avec trois angles droits...) mais pas à une contradiction.

Exemples de systèmes formels

Souvent on ne spécifie pas les règles d'inférence (on utilise celles de la logique classique) : on se contente d'un système axiomatique.
Exemple historique : les *Éléments* d'Euclide ($\sim$ 300 av. J-C).
Axiomes mais aussi « postulats » considérés comme moins évidents, surtout le cinquième postulat, reformulé plus tard en :

Par un point extérieur à une droite passe une unique parallèle à cette droite.

Question : peut-on démontrer ce postulat ?
De nombreux mathématiciens ont essayé, notamment par l'absurde.
En supposant le postulat faux, on arrive à des résultats « manifestement faux » (triangles avec trois angles droits...) mais pas à une contradiction.

Le système reste cohérent en supposant la négation du postulat.
Question : cet autre système a-t-il un sens ?

Exemples de systèmes formels

Souvent on ne spécifie pas les règles d'inférence (on utilise celles de la logique classique) : on se contente d'un système axiomatique.
Exemple historique : les *Éléments* d'Euclide ($\sim$ 300 av. J-C).
Axiomes mais aussi « postulats » considérés comme moins évidents, surtout le cinquième postulat, reformulé plus tard en :

Par un point extérieur à une droite passe une unique parallèle à cette droite.

Question : peut-on démontrer ce postulat ?
De nombreux mathématiciens ont essayé, notamment par l'absurde.
En supposant le postulat faux, on arrive à des résultats « manifestement faux » (triangles avec trois angles droits...) mais pas à une contradiction.

Le système reste cohérent en supposant la négation du postulat.
Question : cet autre système a-t-il un sens ?
Réponse : Oui ! il représente la géométrie sphérique ou hyperbolique suivant les cas (sémantique différente).

Lien avec la sémantique

Si le langage d'un système a une sémantique permettant de définir les notions de conséquence sémantique et formules valides, on dit que le système est :

- correct si la conclusion d'une déduction est toujours une conséquence logique des hypothèses (en particulier tout théorème est valide)
- complet (pour cette sémantique) si toute formule valide est un théorème.

Lien avec la sémantique

Si le langage d'un système a une sémantique permettant de définir les notions de conséquence sémantique et formules valides, on dit que le système est :

- correct si la conclusion d'une déduction est toujours une conséquence logique des hypothèses (en particulier tout théorème est valide)
- complet (pour cette sémantique) si toute formule valide est un théorème.

Souvent, on a une sémantique et on cherche un système déductif correspondant.
Parfois, on a un système et on lui cherche une sémantique...

Des notions relatives...

Le système euclidien avec la négation du cinquième postulat est incorrect pour la géométrie usuelle du plan... mais pas pour la géométrie sphérique.

Des notions relatives...

Le système euclidien avec la négation du cinquième postulat est incorrect pour la géométrie usuelle du plan... mais pas pour la géométrie sphérique.

La logique intuitionniste utilise le langage de la LP mais des règles d'inférence moins puissantes (par exemple $P \vee \neg P$ n'est pas un théorème de cette logique).

Elle est donc incomplète... pour la sémantique usuelle.

Mais on peut définir une sémantique de Kripke où cette logique est complète.

Système formel pour la logique propositionnelle

Il est possible de définir un système formel simple qui soit complet pour la logique propositionnelle.

Système formel pour la logique propositionnelle

Il est possible de définir un système formel simple qui soit complet pour la logique propositionnelle.

Approche « à la Hilbert » : une seule règle d'inférence

$$\frac{\varphi \qquad \varphi \Rightarrow \psi}{\psi}\ \textit{(Modus ponens)}$$

Système formel pour la logique propositionnelle

Il est possible de définir un système formel simple qui soit complet pour la logique propositionnelle.

Approche « à la Hilbert » : une seule règle d'inférence

$$\frac{\varphi \qquad \varphi \Rightarrow \psi}{\psi}$$ *(Modus ponens)*

et plusieurs axiomes. Par exemple :

1. $\varphi \Rightarrow (\psi \Rightarrow \varphi)$
2. $(\varphi \Rightarrow (\psi \Rightarrow \chi)) \Rightarrow ((\varphi \Rightarrow \psi) \Rightarrow (\varphi \Rightarrow \chi))$
3. $(\neg\psi \Rightarrow \neg\varphi) \Rightarrow ((\neg\psi \Rightarrow \varphi) \Rightarrow \psi)$
4. . . . et les règles pour les autres connecteurs

Système formel pour la logique propositionnelle

Il est possible de définir un système formel simple qui soit complet pour la logique propositionnelle.

Approche « à la Hilbert » : une seule règle d'inférence

$$\frac{\varphi \qquad \varphi \Rightarrow \psi}{\psi}\ \textit{(Modus ponens)}$$

et plusieurs axiomes. Par exemple :

1. $\varphi \Rightarrow (\psi \Rightarrow \varphi)$
2. $(\varphi \Rightarrow (\psi \Rightarrow \chi)) \Rightarrow ((\varphi \Rightarrow \psi) \Rightarrow (\varphi \Rightarrow \chi))$
3. $(\neg\psi \Rightarrow \neg\varphi) \Rightarrow ((\neg\psi \Rightarrow \varphi) \Rightarrow \psi)$
4. . . . et les règles pour les autres connecteurs

Inconvénient : les preuves dans ces systèmes ne sont pas très naturelles, il faut construire les bonnes instances d'axiomes pour pouvoir utiliser la règle.

« Déduction naturelle »

Raisonnement conditionnel : Un raisonnement classique pour prouver $\varphi \Rightarrow \psi$ est de supposer φ et d'en déduire ψ. On prouve ψ *sous la condition* que φ soit vrai.

« Déduction naturelle »

Raisonnement conditionnel : Un raisonnement classique pour prouver $\varphi \Rightarrow \psi$ est de supposer φ et d'en déduire ψ. On prouve ψ *sous la condition* que φ soit vrai.
φ est une hypothèse intermédiaire, ce n'est ni une hypothèse de départ, ni un axiome, ni une conclusion provisoire.

« Déduction naturelle »

Raisonnement conditionnel : Un raisonnement classique pour prouver $\varphi \Rightarrow \psi$ est de supposer φ et d'en déduire ψ. On prouve ψ *sous la condition* que φ soit vrai.
φ est une hypothèse intermédiaire, ce n'est ni une hypothèse de départ, ni un axiome, ni une conclusion provisoire.
Ce type de raisonnement ne peut donc pas se traduire dans un système formel simple.

« Déduction naturelle »

Raisonnement conditionnel : Un raisonnement classique pour prouver $\varphi \Rightarrow \psi$ est de supposer φ et d'en déduire ψ. On prouve ψ *sous la condition* que φ soit vrai.
φ est une hypothèse intermédiaire, ce n'est ni une hypothèse de départ, ni un axiome, ni une conclusion provisoire.
Ce type de raisonnement ne peut donc pas se traduire dans un système formel simple.

Solution :

- Preuves structurées avec sous-preuves (déductions ayant leurs hypothèses locales) ;

« Déduction naturelle »

Raisonnement conditionnel : Un raisonnement classique pour prouver $\varphi \Rightarrow \psi$ est de supposer φ et d'en déduire ψ. On prouve ψ *sous la condition* que φ soit vrai.
φ est une hypothèse intermédiaire, ce n'est ni une hypothèse de départ, ni un axiome, ni une conclusion provisoire.
Ce type de raisonnement ne peut donc pas se traduire dans un système formel simple.

Solution :

- Preuves structurées avec sous-preuves (déductions ayant leurs hypothèses locales) ;
- Une règle d'inférence « structurelle » dont la prémisse n'est pas une formule mais une déduction entière :

« Déduction naturelle »

Raisonnement conditionnel : Un raisonnement classique pour prouver $\varphi \Rightarrow \psi$ est de supposer φ et d'en déduire ψ. On prouve ψ *sous la condition* que φ soit vrai.
φ est une hypothèse intermédiaire, ce n'est ni une hypothèse de départ, ni un axiome, ni une conclusion provisoire.
Ce type de raisonnement ne peut donc pas se traduire dans un système formel simple.

Solution :

- Preuves structurées avec sous-preuves (déductions ayant leurs hypothèses locales) ;
- Une règle d'inférence « structurelle » dont la prémisse n'est pas une formule mais une déduction entière :

$$\frac{\varphi \vdash \psi}{\varphi \Rightarrow \psi}$$

Système de Fitch

Un ensemble complet de règles, nommées en fonction du connecteur qu'elles Introduisent ou Éliminent :

Système de Fitch

Un ensemble complet de règles, nommées en fonction du connecteur qu'elles Introduisent ou Éliminent :

- Conjonction : $\text{IC}\ \dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}$

Système de Fitch

Un ensemble complet de règles, nommées en fonction du connecteur qu'elles Introduisent ou Éliminent :

- Conjonction : $\mathsf{IC}\ \dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}$ $\mathsf{EC}_1\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_1}$ $\mathsf{EC}_2\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_2}$

Système de Fitch

Un ensemble complet de règles, nommées en fonction du connecteur qu'elles Introduisent ou Éliminent :

- Conjonction : $\mathrm{IC}\ \dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}$ $\mathrm{EC}_1\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_1}$ $\mathrm{EC}_2\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_2}$
- Disjonction :

$\mathrm{ID}_1\ \dfrac{\varphi_1}{\varphi_1 \vee \varphi_2}$ $\mathrm{ID}_2\ \dfrac{\varphi_2}{\varphi_1 \vee \varphi_2}$

Système de Fitch

Un ensemble complet de règles, nommées en fonction du connecteur qu'elles Introduisent ou Éliminent :

- Conjonction : $\mathsf{IC}\ \dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}$ $\mathsf{EC}_1\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_1}$ $\mathsf{EC}_2\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_2}$
- Disjonction :

$\mathsf{ID}_1\ \dfrac{\varphi_1}{\varphi_1 \vee \varphi_2}$ $\mathsf{ID}_2\ \dfrac{\varphi_2}{\varphi_1 \vee \varphi_2}$ $\mathsf{ED}\ \dfrac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \psi \qquad \varphi_2 \Rightarrow \psi}{\psi}$

Système de Fitch

Un ensemble complet de règles, nommées en fonction du connecteur qu'elles Introduisent ou Éliminent :

- Conjonction : $\mathsf{IC}\ \dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}$ $\mathsf{EC}_1\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_1}$ $\mathsf{EC}_2\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_2}$
- Disjonction :

  $\mathsf{ID}_1\ \dfrac{\varphi_1}{\varphi_1 \vee \varphi_2}$ $\mathsf{ID}_2\ \dfrac{\varphi_2}{\varphi_1 \vee \varphi_2}$ $\mathsf{ED}\ \dfrac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \psi \qquad \varphi_2 \Rightarrow \psi}{\psi}$
- Négation : $\mathsf{IN}\ \dfrac{\varphi \Rightarrow \bot}{\neg\varphi}$

Système de Fitch

Un ensemble complet de règles, nommées en fonction du connecteur qu'elles Introduisent ou Éliminent :

- Conjonction : $\mathsf{IC}\ \dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}$ $\mathsf{EC}_1\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_1}$ $\mathsf{EC}_2\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_2}$
- Disjonction :

  $\mathsf{ID}_1\ \dfrac{\varphi_1}{\varphi_1 \vee \varphi_2}$ $\mathsf{ID}_2\ \dfrac{\varphi_2}{\varphi_1 \vee \varphi_2}$ $\mathsf{ED}\ \dfrac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \psi \qquad \varphi_2 \Rightarrow \psi}{\psi}$
- Négation : $\mathsf{IN}\ \dfrac{\varphi \Rightarrow \bot}{\neg\varphi}$ $\mathsf{EN}\ \dfrac{\neg\neg\varphi}{\varphi}$

Système de Fitch

Un ensemble complet de règles, nommées en fonction du connecteur qu'elles Introduisent ou Éliminent :

- Conjonction : $\mathsf{IC}\ \dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}$ $\mathsf{EC}_1\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_1}$ $\mathsf{EC}_2\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_2}$
- Disjonction :

  $\mathsf{ID}_1\ \dfrac{\varphi_1}{\varphi_1 \vee \varphi_2}$ $\mathsf{ID}_2\ \dfrac{\varphi_2}{\varphi_1 \vee \varphi_2}$ $\mathsf{ED}\ \dfrac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \psi \qquad \varphi_2 \Rightarrow \psi}{\psi}$
- Négation : $\mathsf{IN}\ \dfrac{\varphi \Rightarrow \bot}{\neg\varphi}$ $\mathsf{EN}\ \dfrac{\neg\neg\varphi}{\varphi}$
- Contradiction (Faux) : $\mathsf{IF}\ \dfrac{\varphi \qquad \neg\varphi}{\bot}$

Système de Fitch

Un ensemble complet de règles, nommées en fonction du connecteur qu'elles Introduisent ou Éliminent :

- Conjonction : $\mathsf{IC}\ \dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}$ $\mathsf{EC}_1\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_1}$ $\mathsf{EC}_2\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_2}$
- Disjonction :

  $\mathsf{ID}_1\ \dfrac{\varphi_1}{\varphi_1 \vee \varphi_2}$ $\mathsf{ID}_2\ \dfrac{\varphi_2}{\varphi_1 \vee \varphi_2}$ $\mathsf{ED}\ \dfrac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \psi \qquad \varphi_2 \Rightarrow \psi}{\psi}$
- Négation : $\mathsf{IN}\ \dfrac{\varphi \Rightarrow \bot}{\neg\varphi}$ $\mathsf{EN}\ \dfrac{\neg\neg\varphi}{\varphi}$
- Contradiction (Faux) : $\mathsf{IF}\ \dfrac{\varphi \qquad \neg\varphi}{\bot}$ $\mathsf{EF}\ \dfrac{\bot}{\varphi}$

Système de Fitch

Un ensemble complet de règles, nommées en fonction du connecteur qu'elles Introduisent ou Éliminent :

- Conjonction : $\text{IC}\ \dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}$ $\qquad \text{EC}_1\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_1}$ $\qquad \text{EC}_2\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_2}$
- Disjonction :

  $\text{ID}_1\ \dfrac{\varphi_1}{\varphi_1 \vee \varphi_2}$ $\qquad \text{ID}_2\ \dfrac{\varphi_2}{\varphi_1 \vee \varphi_2}$ $\qquad \text{ED}\ \dfrac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \psi \qquad \varphi_2 \Rightarrow \psi}{\psi}$
- Négation : $\text{IN}\ \dfrac{\varphi \Rightarrow \bot}{\neg\varphi}$ $\qquad \text{EN}\ \dfrac{\neg\neg\varphi}{\varphi}$
- Contradiction (Faux) : $\text{IF}\ \dfrac{\varphi \qquad \neg\varphi}{\bot}$ $\qquad \text{EF}\ \dfrac{\bot}{\varphi}$
- Implication :

Système de Fitch

Un ensemble complet de règles, nommées en fonction du connecteur qu'elles Introduisent ou Éliminent :

- Conjonction : $\mathsf{IC}\ \dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}$ $\mathsf{EC}_1\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_1}$ $\mathsf{EC}_2\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_2}$
- Disjonction :

  $\mathsf{ID}_1\ \dfrac{\varphi_1}{\varphi_1 \vee \varphi_2}$ $\mathsf{ID}_2\ \dfrac{\varphi_2}{\varphi_1 \vee \varphi_2}$ $\mathsf{ED}\ \dfrac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \psi \qquad \varphi_2 \Rightarrow \psi}{\psi}$
- Négation : $\mathsf{IN}\ \dfrac{\varphi \Rightarrow \bot}{\neg\varphi}$ $\mathsf{EN}\ \dfrac{\neg\neg\varphi}{\varphi}$
- Contradiction (Faux) : $\mathsf{IF}\ \dfrac{\varphi \qquad \neg\varphi}{\bot}$ $\mathsf{EF}\ \dfrac{\bot}{\varphi}$
- Implication : $\mathsf{II}\ \dfrac{\varphi \vdash \psi}{\varphi \Rightarrow \psi}$

Système de Fitch

Un ensemble complet de règles, nommées en fonction du connecteur qu'elles Introduisent ou Éliminent :

- Conjonction : $\text{IC}\ \dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}$ $\qquad \text{EC}_1\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_1}$ $\qquad \text{EC}_2\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_2}$
- Disjonction :

  $\text{ID}_1\ \dfrac{\varphi_1}{\varphi_1 \vee \varphi_2}$ $\qquad \text{ID}_2\ \dfrac{\varphi_2}{\varphi_1 \vee \varphi_2}$ $\qquad \text{ED}\ \dfrac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \psi \qquad \varphi_2 \Rightarrow \psi}{\psi}$
- Négation : $\text{IN}\ \dfrac{\varphi \Rightarrow \bot}{\neg\varphi}$ $\qquad \text{EN}\ \dfrac{\neg\neg\varphi}{\varphi}$
- Contradiction (Faux) : $\text{IF}\ \dfrac{\varphi \qquad \neg\varphi}{\bot}$ $\qquad \text{EF}\ \dfrac{\bot}{\varphi}$
- Implication : $\text{II}\ \dfrac{\varphi \vdash \psi}{\varphi \Rightarrow \psi}$ $\qquad \text{EI}\ \dfrac{\varphi \Rightarrow \psi \qquad \varphi}{\psi}$

Système de Fitch

Un ensemble complet de règles, nommées en fonction du connecteur qu'elles Introduisent ou Éliminent :

- Conjonction : $\mathsf{IC}\ \dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}$ $\mathsf{EC}_1\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_1}$ $\mathsf{EC}_2\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_2}$
- Disjonction :

  $\mathsf{ID}_1\ \dfrac{\varphi_1}{\varphi_1 \vee \varphi_2}$ $\mathsf{ID}_2\ \dfrac{\varphi_2}{\varphi_1 \vee \varphi_2}$ $\mathsf{ED}\ \dfrac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \psi \qquad \varphi_2 \Rightarrow \psi}{\psi}$
- Négation : $\mathsf{IN}\ \dfrac{\varphi \Rightarrow \bot}{\neg\varphi}$ $\mathsf{EN}\ \dfrac{\neg\neg\varphi}{\varphi}$
- Contradiction (Faux) : $\mathsf{IF}\ \dfrac{\varphi \qquad \neg\varphi}{\bot}$ $\mathsf{EF}\ \dfrac{\bot}{\varphi}$
- Implication : $\mathsf{II}\ \dfrac{\varphi \vdash \psi}{\varphi \Rightarrow \psi}$ $\mathsf{EI}\ \dfrac{\varphi \Rightarrow \psi \qquad \varphi}{\psi}$
- Équivalence :

  $\mathsf{IE}\ \dfrac{\varphi \Rightarrow \psi \qquad \psi \Rightarrow \varphi}{\varphi \Leftrightarrow \psi}$ $\mathsf{EE}_1\ \dfrac{\varphi \Leftrightarrow \psi}{\varphi \Rightarrow \psi}$ $\mathsf{EE}_2\ \dfrac{\varphi \Leftrightarrow \psi}{\psi \Rightarrow \varphi}$

Système de Fitch

Un ensemble complet de règles, nommées en fonction du connecteur qu'elles Introduisent ou Éliminent :

- Conjonction : $\mathsf{IC}\ \dfrac{\varphi_1 \qquad \varphi_2}{\varphi_1 \wedge \varphi_2}$ $\mathsf{EC}_1\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_1}$ $\mathsf{EC}_2\ \dfrac{\varphi_1 \wedge \varphi_2}{\varphi_2}$
- Disjonction :

  $\mathsf{ID}_1\ \dfrac{\varphi_1}{\varphi_1 \vee \varphi_2}$ $\mathsf{ID}_2\ \dfrac{\varphi_2}{\varphi_1 \vee \varphi_2}$ $\mathsf{ED}\ \dfrac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \psi \qquad \varphi_2 \Rightarrow \psi}{\psi}$
- Négation : $\mathsf{IN}\ \dfrac{\varphi \Rightarrow \bot}{\neg\varphi}$ $\mathsf{EN}\ \dfrac{\neg\neg\varphi}{\varphi}$
- Contradiction (Faux) : $\mathsf{IF}\ \dfrac{\varphi \qquad \neg\varphi}{\bot}$ $\mathsf{EF}\ \dfrac{\bot}{\varphi}$
- Implication : $\mathsf{II}\ \dfrac{\varphi \vdash \psi}{\varphi \Rightarrow \psi}$ $\mathsf{EI}\ \dfrac{\varphi \Rightarrow \psi \qquad \varphi}{\psi}$
- Équivalence : (on peut voir $\varphi \Leftrightarrow \psi$ comme une abrév. de $(\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$)

  $\mathsf{IE}\ \dfrac{\varphi \Rightarrow \psi \qquad \psi \Rightarrow \varphi}{\varphi \Leftrightarrow \psi}$ $\mathsf{EE}_1\ \dfrac{\varphi \Leftrightarrow \psi}{\varphi \Rightarrow \psi}$ $\mathsf{EE}_2\ \dfrac{\varphi \Leftrightarrow \psi}{\psi \Rightarrow \varphi}$

Présentation d'une preuve dans le système de Fitch

On a toujours une suite d'étapes de raisonnement, mais :

- les sous-preuves sont indentées ;
- elles commencent par une hypothèse locale ;
- dans la sous-preuve, on peut utiliser l'hypothèse locale ;
- hors de la sous-preuve, on ne peut plus utiliser ni l'hypothèse locale ni les étapes intermédiaires de la sous-preuve.

Présentation d'une preuve dans le système de Fitch

On a toujours une suite d'étapes de raisonnement, mais :

- les sous-preuves sont indentées ;
- elles commencent par une hypothèse locale ;
- dans la sous-preuve, on peut utiliser l'hypothèse locale ;
- hors de la sous-preuve, on ne peut plus utiliser ni l'hypothèse locale ni les étapes intermédiaires de la sous-preuve.

Exemple : prouver $R \vee (P \wedge R \Rightarrow Q \vee P)$:

Présentation d'une preuve dans le système de Fitch

On a toujours une suite d'étapes de raisonnement, mais :

- les sous-preuves sont indentées ;
- elles commencent par une hypothèse locale ;
- dans la sous-preuve, on peut utiliser l'hypothèse locale ;
- hors de la sous-preuve, on ne peut plus utiliser ni l'hypothèse locale ni les étapes intermédiaires de la sous-preuve.

Exemple : prouver $R \vee (P \wedge R \Rightarrow Q \vee P)$:

$$1 \quad \Big| \quad \Big| \; P \wedge R$$

Présentation d'une preuve dans le système de Fitch

On a toujours une suite d'étapes de raisonnement, mais :

- les sous-preuves sont indentées ;
- elles commencent par une hypothèse locale ;
- dans la sous-preuve, on peut utiliser l'hypothèse locale ;
- hors de la sous-preuve, on ne peut plus utiliser ni l'hypothèse locale ni les étapes intermédiaires de la sous-preuve.

Exemple : prouver $R \vee (P \wedge R \Rightarrow Q \vee P)$:

1	|	$P \wedge R$	
2	|	P	EC, 1

Présentation d'une preuve dans le système de Fitch

On a toujours une suite d'étapes de raisonnement, mais :

- les sous-preuves sont indentées ;
- elles commencent par une hypothèse locale ;
- dans la sous-preuve, on peut utiliser l'hypothèse locale ;
- hors de la sous-preuve, on ne peut plus utiliser ni l'hypothèse locale ni les étapes intermédiaires de la sous-preuve.

Exemple : prouver $R \vee (P \wedge R \Rightarrow Q \vee P)$:

1	|	$P \wedge R$	
2	|	P	EC, 1
3	|	$Q \vee P$	ID, 2

Présentation d'une preuve dans le système de Fitch

On a toujours une suite d'étapes de raisonnement, mais :

- les sous-preuves sont indentées ;
- elles commencent par une hypothèse locale ;
- dans la sous-preuve, on peut utiliser l'hypothèse locale ;
- hors de la sous-preuve, on ne peut plus utiliser ni l'hypothèse locale ni les étapes intermédiaires de la sous-preuve.

Exemple : prouver $R \vee (P \wedge R \Rightarrow Q \vee P)$:

1	$P \wedge R$	
2	P	EC, 1
3	$Q \vee P$	ID, 2
4	$P \wedge R \Rightarrow Q \vee P$	II, 1–3

Présentation d'une preuve dans le système de Fitch

On a toujours une suite d'étapes de raisonnement, mais :

- les sous-preuves sont indentées ;
- elles commencent par une hypothèse locale ;
- dans la sous-preuve, on peut utiliser l'hypothèse locale ;
- hors de la sous-preuve, on ne peut plus utiliser ni l'hypothèse locale ni les étapes intermédiaires de la sous-preuve.

Exemple : prouver $R \vee (P \wedge R \Rightarrow Q \vee P)$:

1	$P \wedge R$	
2	P	EC, 1
3	$Q \vee P$	ID, 2
4	$P \wedge R \Rightarrow Q \vee P$	II, 1–3
5	$R \vee (P \wedge R \Rightarrow Q \vee P)$	ID, 5

Présentation d'une preuve dans le système de Fitch

Dans une sous-preuve, on a le droit d'utiliser les hypothèses ou les étapes précédentes de la preuve qui la contient.

Présentation d'une preuve dans le système de Fitch

Dans une sous-preuve, on a le droit d'utiliser les hypothèses ou les étapes précédentes de la preuve qui la contient.
Pour plus de lisibilité, dans ce cas on recopie les formules qu'on utilise dans la sous-preuve (règle R).

Présentation d'une preuve dans le système de Fitch

Dans une sous-preuve, on a le droit d'utiliser les hypothèses ou les étapes précédentes de la preuve qui la contient.
Pour plus de lisibilité, dans ce cas on recopie les formules qu'on utilise dans la sous-preuve (règle R).

Exemple : $Q \Rightarrow (P \Rightarrow Q)$:

Présentation d'une preuve dans le système de Fitch

Dans une sous-preuve, on a le droit d'utiliser les hypothèses ou les étapes précédentes de la preuve qui la contient.
Pour plus de lisibilité, dans ce cas on recopie les formules qu'on utilise dans la sous-preuve (règle R).

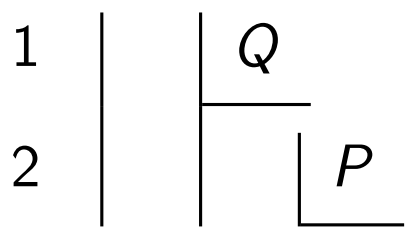
Exemple : $Q \Rightarrow (P \Rightarrow Q)$:

$$\begin{array}{l|l} 1 & \quad \vdots \\ & \quad \underline{Q} \end{array}$$

Présentation d'une preuve dans le système de Fitch

Dans une sous-preuve, on a le droit d'utiliser les hypothèses ou les étapes précédentes de la preuve qui la contient.
Pour plus de lisibilité, dans ce cas on recopie les formules qu'on utilise dans la sous-preuve (règle R).

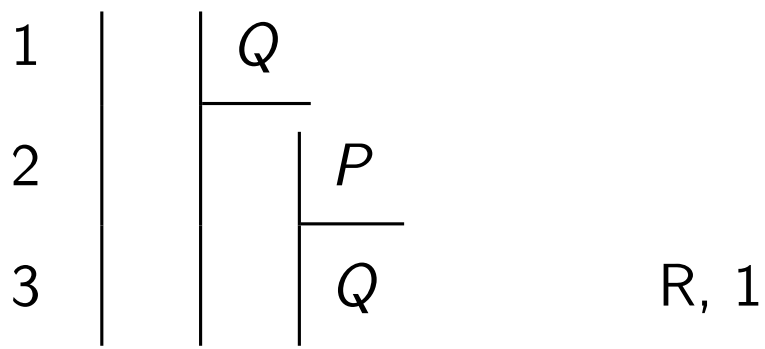
Exemple : $Q \Rightarrow (P \Rightarrow Q)$:



Présentation d'une preuve dans le système de Fitch

Dans une sous-preuve, on a le droit d'utiliser les hypothèses ou les étapes précédentes de la preuve qui la contient.
Pour plus de lisibilité, dans ce cas on recopie les formules qu'on utilise dans la sous-preuve (règle R).

Exemple : $Q \Rightarrow (P \Rightarrow Q)$:



Présentation d'une preuve dans le système de Fitch

Dans une sous-preuve, on a le droit d'utiliser les hypothèses ou les étapes précédentes de la preuve qui la contient.
Pour plus de lisibilité, dans ce cas on recopie les formules qu'on utilise dans la sous-preuve (règle R).

Exemple : $Q \Rightarrow (P \Rightarrow Q)$:

1	Q	
2	$\quad P$	
3	$\quad Q$	R, 1
4	$P \Rightarrow Q$	II, 2–3

Présentation d'une preuve dans le système de Fitch

Dans une sous-preuve, on a le droit d'utiliser les hypothèses ou les étapes précédentes de la preuve qui la contient.
Pour plus de lisibilité, dans ce cas on recopie les formules qu'on utilise dans la sous-preuve (règle R).

Exemple : $Q \Rightarrow (P \Rightarrow Q)$:

1	Q	
2	P	
3	Q	R, 1
4	$P \Rightarrow Q$	II, 2–3
5	$Q \Rightarrow (P \Rightarrow Q)$	II, 1–4

Règles dérivées

En pratique, pour éviter des preuves fastidieuses et répétitives, on utilise souvent des règles supplémentaires. Ces règles peuvent elles-mêmes être prouvées à partir des règles de base.

Règles dérivées

En pratique, pour éviter des preuves fastidieuses et répétitives, on utilise souvent des règles supplémentaires. Ces règles peuvent elles-mêmes être prouvées à partir des règles de base.

- Variantes de ED (élimination de la disjonction) :

$$\frac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \varphi_2}{\varphi_2}$$

Règles dérivées

En pratique, pour éviter des preuves fastidieuses et répétitives, on utilise souvent des règles supplémentaires. Ces règles peuvent elles-mêmes être prouvées à partir des règles de base.

- Variantes de ED (élimination de la disjonction) :

$$\frac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \varphi_2}{\varphi_2} \qquad \frac{\varphi_1 \vee \varphi_2 \qquad \neg\varphi_1}{\varphi_2}$$

Règles dérivées

En pratique, pour éviter des preuves fastidieuses et répétitives, on utilise souvent des règles supplémentaires. Ces règles peuvent elles-mêmes être prouvées à partir des règles de base.

- Variantes de ED (élimination de la disjonction) :

$$\frac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \varphi_2}{\varphi_2} \qquad \frac{\varphi_1 \vee \varphi_2 \qquad \neg\varphi_1}{\varphi_2}$$

- Variante de EI (contraposition) : $\dfrac{\varphi \Rightarrow \psi \qquad \neg\psi}{\neg\varphi}$

Règles dérivées

En pratique, pour éviter des preuves fastidieuses et répétitives, on utilise souvent des règles supplémentaires. Ces règles peuvent elles-mêmes être prouvées à partir des règles de base.

- Variantes de ED (élimination de la disjonction) :

$$\frac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \varphi_2}{\varphi_2} \qquad \frac{\varphi_1 \vee \varphi_2 \qquad \neg\varphi_1}{\varphi_2}$$

- Variante de EI (contraposition) : $\dfrac{\varphi \Rightarrow \psi \qquad \neg\psi}{\neg\varphi}$

- Tautologies de base :

$$\frac{}{\varphi \Rightarrow \varphi} \quad \text{(auto-implication)}$$

Règles dérivées

En pratique, pour éviter des preuves fastidieuses et répétitives, on utilise souvent des règles supplémentaires. Ces règles peuvent elles-mêmes être prouvées à partir des règles de base.

- Variantes de ED (élimination de la disjonction) :

$$\frac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \varphi_2}{\varphi_2} \qquad \frac{\varphi_1 \vee \varphi_2 \qquad \neg\varphi_1}{\varphi_2}$$

- Variante de EI (contraposition) : $\dfrac{\varphi \Rightarrow \psi \qquad \neg\psi}{\neg\varphi}$

- Tautologies de base :

$$\frac{}{\varphi \Rightarrow \varphi} \text{ (auto-implication)} \qquad \frac{}{\varphi \vee \neg\varphi} \text{ (tiers-exclu)}$$

Règles dérivées

En pratique, pour éviter des preuves fastidieuses et répétitives, on utilise souvent des règles supplémentaires. Ces règles peuvent elles-mêmes être prouvées à partir des règles de base.

- Variantes de ED (élimination de la disjonction) :

$$\frac{\varphi_1 \vee \varphi_2 \qquad \varphi_1 \Rightarrow \varphi_2}{\varphi_2} \qquad \frac{\varphi_1 \vee \varphi_2 \qquad \neg\varphi_1}{\varphi_2}$$

- Variante de EI (contraposition) : $\dfrac{\varphi \Rightarrow \psi \qquad \neg\psi}{\neg\varphi}$

- Tautologies de base :

$$\frac{}{\varphi \Rightarrow \varphi}\ \text{(auto-implication)} \qquad \frac{}{\varphi \vee \neg\varphi}\ \text{(tiers-exclu)}$$

- Règles d'équivalence :

$$\frac{\neg(\varphi \vee \psi)}{\neg\varphi \wedge \neg\psi} \qquad \frac{\neg(\varphi \wedge \psi)}{\neg\varphi \vee \neg\psi}\ \text{(De Morgan)} \qquad \frac{\neg\varphi \vee \psi}{\varphi \Rightarrow \psi}\ \text{(trad. impl.)}$$

Équivalence logique entre formules

Plusieurs manières équivalentes de définir sémantiquement l'équivalence logique entre formules :

$\varphi \equiv \psi$ si :

- $\varphi \models \psi$ et $\psi \models \varphi$
- $[\![\varphi]\!] = [\![\psi]\!]$
- $\models \varphi \Leftrightarrow \psi$
- φ et ψ ont la même table de vérité (mais attention, il se peut qu'une des formules comprenne plus de propositions atomiques que l'autre)

Équivalence logique entre formules

Divers cas d'équivalence classiques :

- Commutativité/Associativité de $\wedge$ et $\vee$
- Idempotence : $\varphi \vee \varphi \equiv \varphi$ et $\varphi \wedge \varphi \equiv \varphi$
- Distributivité de $\wedge$ sur $\vee$ et réciproquement :
  - $\varphi \wedge (\psi \vee \chi) \equiv (\varphi \wedge \psi) \vee (\varphi \wedge \chi)$
  - $\varphi \vee (\psi \wedge \chi) \equiv (\varphi \vee \psi) \wedge (\varphi \vee \chi)$
- Lois de De Morgan :
  - $\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$
  - $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$
- Contraposition : $\varphi \Rightarrow \psi \equiv \neg\psi \Rightarrow \neg\varphi$
- $\varphi \Rightarrow \psi \equiv \neg\varphi \vee \psi$
- $\varphi \Leftrightarrow \psi \equiv (\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi) \equiv (\neg\varphi \vee \psi) \wedge (\varphi \vee \neg\psi)$

Permettent de définir l'équivalence par des règles syntaxiques.

Équivalence logique entre formules

Divers cas d'équivalence classiques :

- Commutativité/Associativité de $\wedge$ et $\vee$
- Idempotence : $\varphi \vee \varphi \equiv \varphi$ et $\varphi \wedge \varphi \equiv \varphi$
- Distributivité de $\wedge$ sur $\vee$ et réciproquement :
  - $\varphi \wedge (\psi \vee \chi) \equiv (\varphi \wedge \psi) \vee (\varphi \wedge \chi)$
  - $\varphi \vee (\psi \wedge \chi) \equiv (\varphi \vee \psi) \wedge (\varphi \vee \chi)$
- Lois de De Morgan :
  - $\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$
  - $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$
- Contraposition : $\varphi \Rightarrow \psi \equiv \neg\psi \Rightarrow \neg\varphi$
- $\varphi \Rightarrow \psi \equiv \neg\varphi \vee \psi$
- $\varphi \Leftrightarrow \psi \equiv (\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi) \equiv (\neg\varphi \vee \psi) \wedge (\varphi \vee \neg\psi)$

Permettent de définir l'équivalence par des règles syntaxiques.

Règles communes à toute algèbre de Boole. L'ensemble des formules de la LP modulo équivalence est une algèbre de Boole.

Formes normales

- suppression de $\Rightarrow$ et $\Leftrightarrow$ obtenue par :
  - $A \Rightarrow B \rightarrow \neg A \vee B$
  - $A \Leftrightarrow B \rightarrow (A \wedge B) \vee (\neg A \wedge \neg B)$ ou bien $A \Leftrightarrow B \rightarrow (\neg A \vee B) \wedge (A \vee \neg B)$
- forme normale négative : formule où l'opérateur $\neg$ ne s'applique qu'à des propositions atomiques. Obtenue par :
  - $\neg\neg A \rightarrow A$
  - lois de De Morgan

Définitions : Un littéral est une formule de la forme P ou $\neg P$, avec P proposition atomique. Une clause est une disjonction de littéraux.

- forme normale conjonctive : conjonction de clauses, c'est-à-dire $\bigwedge_{i=1}^{n} \bigvee_{j=1}^{k_i} L_i^j$ où les L_i^j sont des littéraux.
- forme normale disjonctive : disjonction de conjonctions de littéraux, c'est-à-dire $\bigvee_{i=1}^{n} \bigwedge_{j=1}^{k_i} L_i^j$.

Obtenues par distributivité.

Forme clausale

Toute formule peut être écrite sous forme normale conjonctive, appelée aussi forme clausale.
Intérêt : du point de vue sémantique :

conjonction de clauses = ensemble des clauses.

(conjonction vraie si et seulement si toutes les clauses sont vraies, donc modèle de l'ensemble = modèle de la conjonction.)

Forme clausale

Toute formule peut être écrite sous forme normale conjonctive, appelée aussi forme clausale.
Intérêt : du point de vue sémantique :

conjonction de clauses = ensemble des clauses.

(conjonction vraie si et seulement si toutes les clauses sont vraies, donc modèle de l'ensemble = modèle de la conjonction.)
Les clauses elles-mêmes peuvent être vues comme des ensembles de littéraux

- inutile de représenter explicitement les opérateurs $\vee$ et $\wedge$.

Une clause est vraie si et seulement si l'un au moins de ses littéraux est vrai.

Forme clausale

Toute formule peut être écrite sous forme normale conjonctive, appelée aussi forme clausale.
Intérêt : du point de vue sémantique :

conjonction de clauses = ensemble des clauses.

(conjonction vraie si et seulement si toutes les clauses sont vraies, donc modèle de l'ensemble = modèle de la conjonction.)
Les clauses elles-mêmes peuvent être vues comme des ensembles de littéraux

- inutile de représenter explicitement les opérateurs $\vee$ et $\wedge$.

Une clause est vraie si et seulement si l'un au moins de ses littéraux est vrai.
Propriétés si C et D sont des clauses :

- C est valide si et seulement si elle contient P et $\neg P$ pour un certain P.
- si $C \subseteq D$ alors $C \models D$
- si $C \models D$ alors soit D est une tautologie soit $C \subseteq D$

Le problème SAT

On appelle SAT le pb de la satisfaisabilité d'un ensemble de clauses.
C'est un problème algorithmique NP-complet classique.

Le problème SAT

On appelle SAT le pb de la satisfaisabilité d'un ensemble de clauses. C'est un problème algorithmique NP-complet classique.

Problème NP (Non-déterministe Polynomial) : les meilleurs algos connus pour le résoudre ont des points de choix tels que :

- une machine capable d'explorer toutes les branches en même temps répondrait en temps polynomial, mais

Le problème SAT

On appelle SAT le pb de la satisfaisabilité d'un ensemble de clauses. C'est un problème algorithmique NP-complet classique.

Problème NP (Non-déterministe Polynomial) : les meilleurs algos connus pour le résoudre ont des points de choix tels que :

- une machine capable d'explorer toutes les branches en même temps répondrait en temps polynomial, mais
- une machine séquentielle répond en temps exponentiel (dans le pire cas).

Le problème SAT

On appelle SAT le pb de la satisfaisabilité d'un ensemble de clauses. C'est un problème algorithmique NP-complet classique.

Problème NP (Non-déterministe Polynomial) : les meilleurs algos connus pour le résoudre ont des points de choix tels que :

- une machine capable d'explorer toutes les branches en même temps répondrait en temps polynomial, mais
- une machine séquentielle répond en temps exponentiel (dans le pire cas).

Le problème SAT

On appelle SAT le pb de la satisfaisabilité d'un ensemble de clauses. C'est un problème algorithmique NP-complet classique.

Problème NP (Non-déterministe Polynomial) : les meilleurs algos connus pour le résoudre ont des points de choix tels que :

- une machine capable d'explorer toutes les branches en même temps répondrait en temps polynomial, mais
- une machine séquentielle répond en temps exponentiel (dans le pire cas).

Il n'est pas démontré qu'il n'existe pas d'algorithme séquentiel pour résoudre ce problème en temps polynomial (c'est le problème ouvert « P$\neq$NP »).

Le problème SAT

On appelle SAT le pb de la satisfaisabilité d'un ensemble de clauses. C'est un problème algorithmique NP-complet classique.

Problème NP (Non-déterministe Polynomial) : les meilleurs algos connus pour le résoudre ont des points de choix tels que :

- une machine capable d'explorer toutes les branches en même temps répondrait en temps polynomial, mais
- une machine séquentielle répond en temps exponentiel (dans le pire cas).

Il n'est pas démontré qu'il n'existe pas d'algorithme séquentiel pour résoudre ce problème en temps polynomial (c'est le problème ouvert « P$\neq$NP »).

Problème NP-complet : tout autre problème NP peut être ramené à ce problème via une *réduction polynomiale*.

Correction d'un raisonnement et problème SAT

Exemple : étant donné un raisonnement R, comprenant n hypothèses $\{h_1, \ldots, h_n\}$ et une conclusion c, on veut savoir s'il est correct.

Correction d'un raisonnement et problème SAT

Exemple : étant donné un raisonnement R, comprenant n hypothèses $\{h_1, \ldots, h_n\}$ et une conclusion c, on veut savoir s'il est correct.

1. On reformule : R est correct si et seulement si $\{h_1, h_2, \ldots h_n, \neg c\}$ est insatisfaisable.

Correction d'un raisonnement et problème SAT

Exemple : étant donné un raisonnement R, comprenant n hypothèses $\{h_1, \ldots, h_n\}$ et une conclusion c, on veut savoir s'il est correct.

1. On reformule : R est correct si et seulement si $\{h_1, h_2, \ldots h_n, \neg c\}$ est insatisfaisable.
2. On met sous forme clausale chacune des formules h_i et $\neg c$.

Correction d'un raisonnement et problème SAT

Exemple : étant donné un raisonnement R, comprenant n hypothèses $\{h_1, \ldots, h_n\}$ et une conclusion c, on veut savoir s'il est correct.

1. On reformule : R est correct si et seulement si $\{h_1, h_2, \ldots h_n, \neg c\}$ est insatisfaisable.
2. On met sous forme clausale chacune des formules h_i et $\neg c$.
3. On prend l'ensemble de toutes les clauses ainsi obtenues et on se pose le problème SAT.

Correction d'un raisonnement et problème SAT

Exemple : étant donné un raisonnement R, comprenant n hypothèses $\{h_1, \ldots, h_n\}$ et une conclusion c, on veut savoir s'il est correct.

1. On reformule : R est correct si et seulement si $\{h_1, h_2, \ldots h_n, \neg c\}$ est insatisfaisable.
2. On met sous forme clausale chacune des formules h_i et $\neg c$.
3. On prend l'ensemble de toutes les clauses ainsi obtenues et on se pose le problème SAT.

Remarque : cette traduction simple n'est pas toujours polynomiale (à cause de l'étape 2) !
Mais il existe des traductions polynomiales.

Correction d'un raisonnement et problème SAT

Soit : $h_1 = P \vee Q \Rightarrow R \vee S$, $h_2 = S \Rightarrow P$, $c_1 = Q \Rightarrow R$, $c_2 = \neg R \Rightarrow \neg S$.

Considérons $R_1 = \dfrac{\dfrac{h_2}{h_1}}{c_1}$ et $R_2 = \dfrac{\dfrac{h_2}{h_1}}{c_2}$.

Correction d'un raisonnement et problème SAT

Soit : $h_1 = P \vee Q \Rightarrow R \vee S$, $h_2 = S \Rightarrow P$, $c_1 = Q \Rightarrow R$, $c_2 = \neg R \Rightarrow \neg S$.

Considérons $R_1 = \dfrac{\begin{matrix} h_1 \\ h_2 \end{matrix}}{c_1}$ et $R_2 = \dfrac{\begin{matrix} h_1 \\ h_2 \end{matrix}}{c_2}$.

On a :

$h_1 \equiv \neg(P \vee Q) \vee R \vee S \equiv (\neg P \wedge \neg Q) \vee R \vee S \equiv (\neg P \vee R \vee S) \wedge (\neg Q \vee R \vee S)$;

$h_2 \equiv \neg S \vee P$; $\quad \neg c_1 \equiv Q \wedge \neg R$; $\quad \neg c_2 \equiv \neg R \wedge S$.

Correction d'un raisonnement et problème SAT

Soit : $h_1 = P \vee Q \Rightarrow R \vee S$, $h_2 = S \Rightarrow P$, $c_1 = Q \Rightarrow R$, $c_2 = \neg R \Rightarrow \neg S$.

Considérons $R_1 = \dfrac{\begin{matrix} h_1 \\ h_2 \end{matrix}}{c_1}$ et $R_2 = \dfrac{\begin{matrix} h_1 \\ h_2 \end{matrix}}{c_2}$.

On a :

$h_1 \equiv \neg(P \vee Q) \vee R \vee S \equiv (\neg P \wedge \neg Q) \vee R \vee S \equiv (\neg P \vee R \vee S) \wedge (\neg Q \vee R \vee S)$;

$h_2 \equiv \neg S \vee P$; $\quad \neg c_1 \equiv Q \wedge \neg R$; $\quad \neg c_2 \equiv \neg R \wedge S$.

Soit $\Gamma_1 = \{\{\neg P, R, S\}, \{\neg Q, R, S\}, \{\neg S, P\}, \{Q\}, \{\neg R\}\}$.

Soit $\Gamma_2 = \{\{\neg P, R, S\}, \{\neg Q, R, S\}, \{\neg S, P\}, \{\neg R\}, \{S\}\}$.

R_1 est correct ssi SAT(Γ_1) est faux, R_2 ssi SAT(Γ_2) est faux.

L'algorithme DPLL (Davis–Putnam–Logemann–Loveland)

Conventions :

- La clause vide (ne contenant aucun littéral) est assimilée à la formule toujours fausse : $\bot$
- Un ensemble de clauses ne contenant aucune clause est assimilé à la formule toujours vraie : $\top$

L'algorithme DPLL (Davis–Putnam–Logemann–Loveland)

Conventions :

- La clause vide (ne contenant aucun littéral) est assimilée à la formule toujours fausse : $\bot$
- Un ensemble de clauses ne contenant aucune clause est assimilé à la formule toujours vraie : $\top$

Soit Γ un ensemble de clauses et soit L un littéral, on note $\Gamma[L]$ l'ensemble obtenu à partir de Γ en :

- supprimant toutes les clauses comprenant L,
- supprimant le complémentaire de L de toutes les clauses où il apparaît (cela peut générer une clause vide, $\bot$).

L'algorithme DPLL (Davis–Putnam–Logemann–Loveland)

Conventions :

- La clause vide (ne contenant aucun littéral) est assimilée à la formule toujours fausse : $\bot$
- Un ensemble de clauses ne contenant aucune clause est assimilé à la formule toujours vraie : $\top$

Soit Γ un ensemble de clauses et soit L un littéral, on note $\Gamma[L]$ l'ensemble obtenu à partir de Γ en :

- supprimant toutes les clauses comprenant L,
- supprimant le complémentaire de L de toutes les clauses où il apparaît (cela peut générer une clause vide, $\bot$).

Exemple : si $\Gamma = \{\{P, Q\}, \{\neg P, \neg Q\}\}$, on a $\Gamma[\neg P] =$

L'algorithme DPLL (Davis–Putnam–Logemann–Loveland)

Conventions :

- La clause vide (ne contenant aucun littéral) est assimilée à la formule toujours fausse : $\bot$
- Un ensemble de clauses ne contenant aucune clause est assimilé à la formule toujours vraie : $\top$

Soit Γ un ensemble de clauses et soit L un littéral, on note $\Gamma[L]$ l'ensemble obtenu à partir de Γ en :

- supprimant toutes les clauses comprenant L,
- supprimant le complémentaire de L de toutes les clauses où il apparaît (cela peut générer une clause vide, $\bot$).

Exemple : si $\Gamma = \{\{P, Q\}, \{\neg P, \neg Q\}\}$, on a $\Gamma[\neg P] = \{\{Q\}\}$

L'algorithme DPLL (Davis–Putnam–Logemann–Loveland)

Conventions :

- La clause vide (ne contenant aucun littéral) est assimilée à la formule toujours fausse : $\bot$
- Un ensemble de clauses ne contenant aucune clause est assimilé à la formule toujours vraie : $\top$

Soit Γ un ensemble de clauses et soit L un littéral, on note $\Gamma[L]$ l'ensemble obtenu à partir de Γ en :

- supprimant toutes les clauses comprenant L,
- supprimant le complémentaire de L de toutes les clauses où il apparaît (cela peut générer une clause vide, $\bot$).

Exemple : si $\Gamma = \{\{P, Q\}, \{\neg P, \neg Q\}\}$, on a $\Gamma[\neg P] = \{\{Q\}\}$

Idée : $\Gamma[L]$ est la simplification de Γ qu'on peut faire si l'on suppose que L est vrai.

L'algorithme DPLL

Propriétés :

- $\Gamma[P]$ et $\Gamma[\neg P]$ ne contiennent pas le symbole P.

L'algorithme DPLL

Propriétés :

- $\Gamma[P]$ et $\Gamma[\neg P]$ ne contiennent pas le symbole P.
- Soit I un modèle de Γ, alors :
  - Si $\mathrm{I}(P) = \mathsf{V}$, I est un modèle de $\Gamma[P]$ et
  - Si $\mathrm{I}(P) = \mathsf{F}$, I est un modèle de $\Gamma[\neg P]$

L'algorithme DPLL

Propriétés :

- $\Gamma[P]$ et $\Gamma[\neg P]$ ne contiennent pas le symbole P.
- Soit I un modèle de Γ, alors :
  - Si $\mathrm{I}(P) = \mathsf{V}$, I est un modèle de $\Gamma[P]$ et
  - Si $\mathrm{I}(P) = \mathsf{F}$, I est un modèle de $\Gamma[\neg P]$
- Soit I un modèle de $\Gamma[P]$. Soit J l'interprétation obtenue en ajoutant à I : $\begin{matrix} P \\ \mathsf{V} \end{matrix}$. Alors J est un modèle de Γ.

L'algorithme DPLL

Propriétés :

- $\Gamma[P]$ et $\Gamma[\neg P]$ ne contiennent pas le symbole P.
- Soit I un modèle de Γ, alors :
  - Si $\mathrm{I}(P) = \mathsf{V}$, I est un modèle de $\Gamma[P]$ et
  - Si $\mathrm{I}(P) = \mathsf{F}$, I est un modèle de $\Gamma[\neg P]$
- Soit I un modèle de $\Gamma[P]$. Soit J l'interprétation obtenue en ajoutant à I : $\begin{matrix} P \\ \mathsf{V} \end{matrix}$. Alors J est un modèle de Γ.
- Soit I un modèle de $\Gamma[\neg P]$. Soit J l'interprétation obtenue en ajoutant à I : $\begin{matrix} P \\ \mathsf{F} \end{matrix}$. Alors J est un modèle de Γ.

L'algorithme DPLL

Propriétés :

- $\Gamma[P]$ et $\Gamma[\neg P]$ ne contiennent pas le symbole P.
- Soit I un modèle de Γ, alors :
  - Si $\mathrm{I}(P) = \mathsf{V}$, I est un modèle de $\Gamma[P]$ et
  - Si $\mathrm{I}(P) = \mathsf{F}$, I est un modèle de $\Gamma[\neg P]$
- Soit I un modèle de $\Gamma[P]$. Soit J l'interprétation obtenue en ajoutant à I : $\begin{matrix} P \\ \mathsf{V} \end{matrix}$. Alors J est un modèle de Γ.
- Soit I un modèle de $\Gamma[\neg P]$. Soit J l'interprétation obtenue en ajoutant à I : $\begin{matrix} P \\ \mathsf{F} \end{matrix}$. Alors J est un modèle de Γ.

Conséquence : Γ est satisfaisable si et seulement si l'un des ensembles $\Gamma[P]$ ou $\Gamma[\neg P]$ est satisfaisable.

L'algorithme DPLL

Algorithme récursif « diviser pour régner ».

Principe de base pour déterminer SAT(Γ) :

L'algorithme DPLL

Algorithme récursif « diviser pour régner ».

Principe de base pour déterminer SAT(Γ) :

- Si Γ est vide, on répond SAT ;

L'algorithme DPLL

Algorithme récursif « diviser pour régner ».

Principe de base pour déterminer SAT(Γ) :

- Si Γ est vide, on répond SAT ;
- Si Γ contient une clause vide, on répond INSAT ;

L'algorithme DPLL

Algorithme récursif « diviser pour régner ».

Principe de base pour déterminer SAT(Γ) :

- Si Γ est vide, on répond SAT ;
- Si Γ contient une clause vide, on répond INSAT ;
- Sinon, on choisit (non déterminisme) un symbole P dans Γ et :

L'algorithme DPLL

Algorithme récursif « diviser pour régner ».

Principe de base pour déterminer SAT(Γ) :

- Si Γ est vide, on répond SAT ;
- Si Γ contient une clause vide, on répond INSAT ;
- Sinon, on choisit (non déterminisme) un symbole P dans Γ et :
  1. On calcule $\Gamma[P]$ et on lui applique récursivement DPLL.
  2. Si la réponse est SAT , on répond SAT .

L'algorithme DPLL

Algorithme récursif « diviser pour régner ».

Principe de base pour déterminer SAT(Γ) :

- Si Γ est vide, on répond SAT ;
- Si Γ contient une clause vide, on répond INSAT ;
- Sinon, on choisit (non déterminisme) un symbole P dans Γ et :
  1. On calcule $\Gamma[P]$ et on lui applique récursivement DPLL.
  2. Si la réponse est SAT , on répond SAT .
  3. Sinon, on calcule $\Gamma[\neg P]$ et on lui applique récursivement DPLL.
  4. Si la réponse est SAT , on répond SAT .

L'algorithme DPLL

Algorithme récursif « diviser pour régner ».

Principe de base pour déterminer SAT(Γ) :

- Si Γ est vide, on répond SAT ;
- Si Γ contient une clause vide, on répond INSAT ;
- Sinon, on choisit (non déterminisme) un symbole P dans Γ et :
  1. On calcule $\Gamma[P]$ et on lui applique récursivement DPLL.
  2. Si la réponse est SAT , on répond SAT .
  3. Sinon, on calcule $\Gamma[\neg P]$ et on lui applique récursivement DPLL.
  4. Si la réponse est SAT , on répond SAT .
  5. Sinon, on répond INSAT.

L'algorithme DPLL

Algorithme récursif « diviser pour régner ».

Principe de base pour déterminer SAT(Γ) :

- Si Γ est vide, on répond SAT (et une interp. I vide) ;
- Si Γ contient une clause vide, on répond INSAT ;
- Sinon, on choisit (non déterminisme) un symbole P dans Γ et :
  1. On calcule $\Gamma[P]$ et on lui applique récursivement DPLL.
  2. Si la réponse est SAT (et I), on répond SAT (et $\mathrm{I} + \begin{smallmatrix} P \\ \mathsf{V} \end{smallmatrix}$).
  3. Sinon, on calcule $\Gamma[\neg P]$ et on lui applique récursivement DPLL.
  4. Si la réponse est SAT (et I), on répond SAT (et $\mathrm{I} + \begin{smallmatrix} P \\ \mathsf{F} \end{smallmatrix}$).
  5. Sinon, on répond INSAT.

Dans le cas où la réponse est SAT, on peut renvoyer un modèle.

L'algorithme DPLL

Remarque : le temps de calcul dans le pire cas augmente exponentiellement avec le nombre de symboles propositionnels présents.

L'algorithme DPLL

Remarque : le temps de calcul dans le pire cas augmente exponentiellement avec le nombre de symboles propositionnels présents... comme pour l'algorithme naïf consistant à calculer toute la table de vérité.

L'algorithme DPLL

Remarque : le temps de calcul dans le pire cas augmente exponentiellement avec le nombre de symboles propositionnels présents... comme pour l'algorithme naïf consistant à calculer toute la table de vérité.

Mais :

- En faisant les bons choix, on peut souvent éviter l'exponentielle
- Même dans le pire cas, l'algorithme est beaucoup plus efficace

L'algorithme DPLL

Remarque : le temps de calcul dans le pire cas augmente exponentiellement avec le nombre de symboles propositionnels présents... comme pour l'algorithme naïf consistant à calculer toute la table de vérité.

Mais :

- En faisant les bons choix, on peut souvent éviter l'exponentielle
- Même dans le pire cas, l'algorithme est beaucoup plus efficace

Les bons choix ?

L'algorithme DPLL

Remarque : le temps de calcul dans le pire cas augmente exponentiellement avec le nombre de symboles propositionnels présents... comme pour l'algorithme naïf consistant à calculer toute la table de vérité.

Mais :

- En faisant les bons choix, on peut souvent éviter l'exponentielle
- Même dans le pire cas, l'algorithme est beaucoup plus efficace

Les bons choix ?

- Propagation unitaire : lorsqu'on a une clause de taille 1 (un seul littéral L)

L'algorithme DPLL

Remarque : le temps de calcul dans le pire cas augmente exponentiellement avec le nombre de symboles propositionnels présents... comme pour l'algorithme naïf consistant à calculer toute la table de vérité.

Mais :

- En faisant les bons choix, on peut souvent éviter l'exponentielle
- Même dans le pire cas, l'algorithme est beaucoup plus efficace

Les bons choix ?

- Propagation unitaire : lorsqu'on a une clause de taille 1 (un seul littéral L), il suffit de calculer DPLL($\Gamma[L]$) (l'autre branche est forcément INSAT).

L'algorithme DPLL

Remarque : le temps de calcul dans le pire cas augmente exponentiellement avec le nombre de symboles propositionnels présents... comme pour l'algorithme naïf consistant à calculer toute la table de vérité.

Mais :

- En faisant les bons choix, on peut souvent éviter l'exponentielle
- Même dans le pire cas, l'algorithme est beaucoup plus efficace

Les bons choix ?

- Propagation unitaire : lorsqu'on a une clause de taille 1 (un seul littéral L), il suffit de calculer DPLL($\Gamma[L]$) (l'autre branche est forcément INSAT).
- Littéral pur : si Γ contient un littéral L dont la négation n'apparaît nulle part

L'algorithme DPLL

Remarque : le temps de calcul dans le pire cas augmente exponentiellement avec le nombre de symboles propositionnels présents... comme pour l'algorithme naïf consistant à calculer toute la table de vérité.

Mais :

- En faisant les bons choix, on peut souvent éviter l'exponentielle
- Même dans le pire cas, l'algorithme est beaucoup plus efficace

Les bons choix ?

- Propagation unitaire : lorsqu'on a une clause de taille 1 (un seul littéral L), il suffit de calculer DPLL($\Gamma[L]$) (l'autre branche est forcément INSAT).
- Littéral pur : si Γ contient un littéral L dont la négation n'apparaît nulle part, il suffit de calculer DPLL($\Gamma[L]$) (l'autre branche est un sur-ensemble de $\Gamma[L]$, donc si $\Gamma[L]$ est INSAT, l'autre branche aussi).

L'algorithme DPLL

Algorithme complet (version basique) :

1. Si Γ est vide (il ne contient aucune clause), répondre SAT.
2. Si Γ comprend une clause vide, répondre INSAT.
3. Propagation unitaire : Si Γ comprend une clause unitaire, réduite à un unique littéral L, répondre DPLL($\Gamma[L]$)
4. Subsomption : Si Γ comprend deux clauses C et D telles que $C \subseteq D$, répondre DPLL($\Gamma \setminus D$).
5. Littéral pur : On appelle *littéral pur* un littéral (positif ou négatif) qui apparaît dans certaines clauses mais dont la négation n'apparaît jamais. Si Γ comprend un littéral pur L, répondre DPLL($\Gamma[L]$)
6. Division : choisir un symbole propositionnel P dans Γ et calculer DPLL($\Gamma[P]$). Si la réponse est INSAT répondre DPLL($\Gamma[\neg P]$), sinon répondre SAT.

Deuxième partie

Logique du premier ordre

ou *calcul des prédicats*

Langage de la logique du premier ordre (calcul des prédicats)

Un langage de la logique du premier ordre est caractérisé par :

- Un ensemble $\mathcal{F}$ de symboles de fonction $f, g, h \ldots$ auxquels sont associés des arités (nombres d'arguments).
  Pour préciser l'arité d'un symbole on la note en exposant :
  $f^i, g^j, h^k \ldots \quad (i, j, k \in \mathbb{N})$
- Un ensemble $\mathcal{P}$ de symboles de prédicats (ou relations) également munis d'arités $P^i, Q^j, H^k \ldots$

Un symbole de « fonction » d'arité 0 (sans argument) représente une constante.
Le couple $\Sigma = (\mathcal{F}, \mathcal{P})$ est appelé signature du langage.

Langage de la logique du premier ordre (calcul des prédicats)

Un langage de la logique du premier ordre est caractérisé par :

- Un ensemble $\mathcal{F}$ de symboles de fonction $f, g, h \ldots$ auxquels sont associés des arités (nombres d'arguments).
  Pour préciser l'arité d'un symbole on la note en exposant :
  $f^i, g^j, h^k \ldots \quad (i, j, k \in \mathbb{N})$
- Un ensemble $\mathcal{P}$ de symboles de prédicats (ou relations) également munis d'arités $P^i, Q^j, H^k \ldots$

Un symbole de « fonction » d'arité 0 (sans argument) représente une constante.
Le couple $\Sigma = (\mathcal{F}, \mathcal{P})$ est appelé signature du langage.

En plus de ces symboles spécifiques, la syntaxe comprend également, pour tous les langages :

- Un ensemble infini $\mathcal{V}$ de variables $x, y, z \ldots$
- Les connecteurs logiques $(\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow)$
- Les quantificateurs universel $(\forall)$ et existentiel $(\exists)$.

Syntaxe de la logique du premier ordre

Un terme du premier ordre est :

- soit une variable,
- soit un symbole de fonction d'arité k suivi d'un k-uplet de termes.

Par exemple $f^3(g^2(x,y),y,h^1(g^2(z,x)))$ ou a^0 sont des termes.

Syntaxe de la logique du premier ordre

Un terme du premier ordre est :

- soit une variable,
- soit un symbole de fonction d'arité k suivi d'un k-uplet de termes.

Par exemple $f^3(g^2(x, y), y, h^1(g^2(z, x)))$ ou a^0 sont des termes.
T_Σ : ens. des termes sur la signature Σ.

Syntaxe de la logique du premier ordre

Un terme du premier ordre est :

- soit une variable,
- soit un symbole de fonction d'arité k suivi d'un k-uplet de termes.

Par exemple $f^3(g^2(x,y),y,h^1(g^2(z,x)))$ ou a^0 sont des termes.
T_Σ : ens. des termes sur la signature Σ.

Une formule atomique ou atome est un symbole de prédicat d'arité k suivi d'un k-uplet de termes.
Par exemple $P^2(f^3(g^2(x,y),y,h^1(g^2(z,x))),a^0)$ ou Q^0 sont des atomes.

Syntaxe de la logique du premier ordre

Un terme du premier ordre est :

- soit une variable,
- soit un symbole de fonction d'arité k suivi d'un k-uplet de termes.

Par exemple $f^3(g^2(x,y),y,h^1(g^2(z,x)))$ ou a^0 sont des termes.
T_Σ : ens. des termes sur la signature Σ.

Une formule atomique ou atome est un symbole de prédicat d'arité k suivi d'un k-uplet de termes.
Par exemple $P^2(f^3(g^2(x,y),y,h^1(g^2(z,x))),a^0)$ ou Q^0 sont des atomes.
A_Σ : ens. des atomes sur la signature Σ.

Syntaxe de la logique du premier ordre

Un terme du premier ordre est :

- soit une variable,
- soit un symbole de fonction d'arité k suivi d'un k-uplet de termes.

Par exemple $f^3(g^2(x,y),y,h^1(g^2(z,x)))$ ou a^0 sont des termes.
T_Σ : ens. des termes sur la signature Σ.

Un terme représente un objet dont on parle.

Une formule atomique ou atome est un symbole de prédicat d'arité k suivi d'un k-uplet de termes.
Par exemple $P^2(f^3(g^2(x,y),y,h^1(g^2(z,x))),a^0)$ ou Q^0 sont des atomes.
A_Σ : ens. des atomes sur la signature Σ.

Un atome représente une proposition qui peut être vraie ou fausse.

Syntaxe de la logique du premier ordre

Une formule du premier ordre est :

- soit une formule atomique,
- soit $\top$, soit $\bot$
- soit $\neg\varphi$ où φ est une formule,
- soit de la forme $\varphi * \psi$ avec $* \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$
- soit de la forme $\exists x\, \varphi$ ou $\forall x\, \varphi$ où x est une variable.

F_Σ : ens. des formules sur la signature Σ.
F_Σ est aussi appelé langage du premier ordre de signature Σ.

Syntaxe de la logique du premier ordre

Une formule du premier ordre est :

- soit une formule atomique,
- soit $\top$, soit $\bot$
- soit $\neg\varphi$ où φ est une formule,
- soit de la forme $\varphi * \psi$ avec $* \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$
- soit de la forme $\exists x\, \varphi$ ou $\forall x\, \varphi$ où x est une variable.

F_Σ : ens. des formules sur la signature Σ.
F_Σ est aussi appelé langage du premier ordre de signature Σ.

Convention : la portée d'un quantificateur va jusqu'à la fin de la parenthèse où il se trouve, par exemple :
$\forall x\, \varphi \wedge \psi$ signifie $\forall x\, (\varphi \wedge \psi)$ et non $(\forall x\, \varphi) \wedge \psi$.

Sémantique de la logique du premier ordre : réalisations

Une réalisation $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ d'un langage du premier ordre consiste en :

- un ensemble d'objets $\mathcal{D}$, non vide, appelé domaine ou univers du discours ;
- une interprétation I des différents symboles, c'est-à-dire :

Sémantique de la logique du premier ordre : réalisations

Une réalisation $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ d'un langage du premier ordre consiste en :

- un ensemble d'objets $\mathcal{D}$, non vide, appelé domaine ou univers du discours ;
- une interprétation I des différents symboles, c'est-à-dire :
  - l'affectation à chaque symbole de fonction f^k d'une application (fonction définie partout) $\mathrm{I}(f) : \mathcal{D}^k \to \mathcal{D}$;
  - l'affectation à chaque symbole de prédicat P^k d'une relation k-aire sur $\mathcal{D}$, c'est-à-dire d'un sous-ensemble $\mathrm{I}(P) \subseteq \mathcal{D}^k$.

Sémantique de la logique du premier ordre : réalisations

Une réalisation $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ d'un langage du premier ordre consiste en :

- un ensemble d'objets $\mathcal{D}$, non vide, appelé domaine ou univers du discours ;
- une interprétation I des différents symboles, c'est-à-dire :
  - l'affectation à chaque symbole de fonction f^k d'une application (fonction définie partout) $\mathrm{I}(f) : \mathcal{D}^k \to \mathcal{D}$;
  - l'affectation à chaque symbole de prédicat P^k d'une relation k-aire sur $\mathcal{D}$, c'est-à-dire d'un sous-ensemble $\mathrm{I}(P) \subseteq \mathcal{D}^k$. Remarque : c'est équivalent à une application de $\mathcal{D}^k$ dans $\{\mathsf{V}, \mathsf{F}\}$.

Sémantique de la logique du premier ordre : réalisations

Une réalisation $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ d'un langage du premier ordre consiste en :

- un ensemble d'objets $\mathcal{D}$, non vide, appelé domaine ou univers du discours ;
- une interprétation I des différents symboles, c'est-à-dire :
  - l'affectation à chaque symbole de fonction f^k d'une application (fonction définie partout) $\mathrm{I}(f) : \mathcal{D}^k \to \mathcal{D}$;
  - l'affectation à chaque symbole de prédicat P^k d'une relation k-aire sur $\mathcal{D}$, c'est-à-dire d'un sous-ensemble $\mathrm{I}(P) \subseteq \mathcal{D}^k$. Remarque : c'est équivalent à une application de $\mathcal{D}^k$ dans $\{\mathsf{V}, \mathsf{F}\}$.

Par exemple, soit le langage de signature $(\{f^2, g^1\}, \{P^2, Q^3\})$.
Une réalisation possible de ce langage est :

$$\left(\mathbb{R}, \quad \left\{ \begin{array}{ll} f \mapsto \text{multiplication}, & g \mapsto \text{valeur absolue}, \\ P \mapsto \leqslant, & Q \mapsto \{(x, y, z) \mid x + y + z = 0\} \end{array} \right\}\right).$$

Sémantique de la logique du premier ordre : réalisations

Une réalisation $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ d'un langage du premier ordre consiste en :

- un ensemble d'objets $\mathcal{D}$, non vide, appelé domaine ou univers du discours ;
- une interprétation I des différents symboles, c'est-à-dire :
  - l'affectation à chaque symbole de fonction f^k d'une application (fonction définie partout) $\mathrm{I}(f) : \mathcal{D}^k \to \mathcal{D}$;
  - l'affectation à chaque symbole de prédicat P^k d'une relation k-aire sur $\mathcal{D}$, c'est-à-dire d'un sous-ensemble $\mathrm{I}(P) \subseteq \mathcal{D}^k$. Remarque : c'est équivalent à une application de $\mathcal{D}^k$ dans $\{\mathrm{V}, \mathrm{F}\}$.

Par exemple, soit le langage de signature $(\{f^2, g^1\}, \{P^2, Q^3\})$.
Une réalisation possible de ce langage est :

$$\left(\mathbb{R}, \quad \left\{ \begin{array}{ll} f \mapsto \text{multiplication}, & g \mapsto \text{valeur absolue}, \\ P \mapsto \leqslant, & Q \mapsto \{(x, y, z) \mid x + y + z = 0\} \end{array} \right\}\right).$$

Une autre est :

$$\left(\left\{ \begin{array}{llll} \text{Tom}, & \text{Jerry}, \\ \text{Laurel}, & \text{Hardy} \end{array} \right\}, \quad \left\{ \begin{array}{ll} f \mapsto \text{le plus gros parmi}, & g \mapsto \text{le partenaire de}, \\ P \mapsto \text{souhaiterait manger}, & Q \mapsto \text{de même espèce} \end{array} \right\}\right).$$

Sémantique de la logique du premier ordre : affectations

Soient F_Σ un langage du premier ordre et $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ une réalisation de ce langage. Une affectation de valeurs aux variables (ou valuation) est une application $\theta : \mathcal{V} \to \mathcal{D}$.

Sémantique de la logique du premier ordre : affectations

Soient F_Σ un langage du premier ordre et $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ une réalisation de ce langage. Une affectation de valeurs aux variables (ou valuation) est une application $\theta : \mathcal{V} \to \mathcal{D}$.

Notation : On notera $\theta\{x \mapsto a\}$ la fonction qui affecte à x la valeur a et à toute autre variable la même valeur que θ :

$$(\theta\{x \mapsto a\})(y) = \begin{cases} a & \text{si } y = x \\ \theta(y) & \text{sinon.} \end{cases}$$

Sémantique de la logique du premier ordre : valeur d'un terme

À partir d'une réalisation et d'une affectation, on peut définir la valeur d'un terme : $\mathrm{Val}_{\mathcal{M}}(t, \theta)$ est l'élément de $\mathcal{D}$ obtenu en remplaçant, dans t, chaque variable x par $\theta(x)$ et chaque symbole de fonction f par $\mathrm{I}(f)$.

Exemple : Soit $\Sigma = (\{f^2, g^2\}, \varnothing)$ et soit t le terme $f(x, g(y, x))$.

Sémantique de la logique du premier ordre : valeur d'un terme

À partir d'une réalisation et d'une affectation, on peut définir la valeur d'un terme : $\mathrm{Val}_{\mathcal{M}}(t, \theta)$ est l'élément de $\mathcal{D}$ obtenu en remplaçant, dans t, chaque variable x par $\theta(x)$ et chaque symbole de fonction f par $\mathrm{I}(f)$.

Exemple : Soit $\Sigma = (\{f^2, g^2\}, \varnothing)$ et soit t le terme $f(x, g(y, x))$.

- si $\mathcal{M}$ est $(\mathbb{R}, \{f \mapsto +, g \mapsto \times\})$ et si θ est telle que $\theta(x) = 1$ et $\theta(y) = 2$, on a $\mathrm{Val}_{\mathcal{M}}(t, \theta) = 1 + 2 \times 1 = 3$.

Sémantique de la logique du premier ordre : valeur d'un terme

À partir d'une réalisation et d'une affectation, on peut définir la valeur d'un terme : $\mathrm{Val}_{\mathcal{M}}(t, \theta)$ est l'élément de $\mathcal{D}$ obtenu en remplaçant, dans t, chaque variable x par $\theta(x)$ et chaque symbole de fonction f par $\mathrm{I}(f)$.

Exemple : Soit $\Sigma = (\{f^2, g^2\}, \varnothing)$ et soit t le terme $f(x, g(y, x))$.

- si $\mathcal{M}$ est $(\mathbb{R}, \{f \mapsto +, g \mapsto \times\})$ et si θ est telle que $\theta(x) = 1$ et $\theta(y) = 2$, on a $\mathrm{Val}_{\mathcal{M}}(t, \theta) = 1 + 2 \times 1 = 3$.
- Si $\mathcal{M}$ est $(\{\text{Tom}, \text{Jerry}\}, \{f \mapsto \text{le plus grand}, g \mapsto \text{le plus petit}\})$ et si θ est telle que $\theta(x) = \text{Tom}$ et $\theta(y) = \text{Jerry}$, on a $\mathrm{Val}_{\mathcal{M}}(t, \theta) = \text{Tom}$.

Sémantique de la logique du premier ordre : évaluation

On définit la fonction d'évaluation d'une formule $\varphi \in F_\Sigma$ pour une réalisation $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ et une affectation θ, $\mathcal{E}(\varphi, \mathcal{M}, \theta)$:

Sémantique de la logique du premier ordre : évaluation

On définit la fonction d'évaluation d'une formule $\varphi \in F_\Sigma$ pour une réalisation $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ et une affectation θ, $\mathcal{E}(\varphi, \mathcal{M}, \theta)$:

- Pour un atome : $\mathcal{E}(P(t_1, \ldots, t_n), \mathcal{M}, \theta)$ vaut V si le n-uplet $(\mathrm{Val}_\mathcal{M}(t_1, \theta), \ldots, \mathrm{Val}_\mathcal{M}(t_n, \theta))$ appartient à la relation $\mathrm{I}(P)$, F s'il n'appartient pas à cette relation ;

Sémantique de la logique du premier ordre : évaluation

On définit la fonction d'évaluation d'une formule $\varphi \in F_\Sigma$ pour une réalisation $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ et une affectation θ, $\mathcal{E}(\varphi, \mathcal{M}, \theta)$:

- Pour un atome : $\mathcal{E}(P(t_1, \ldots, t_n), \mathcal{M}, \theta)$ vaut V si le n-uplet $(\mathrm{Val}_\mathcal{M}(t_1, \theta), \ldots, \mathrm{Val}_\mathcal{M}(t_n, \theta))$ appartient à la relation $\mathrm{I}(P)$, F s'il n'appartient pas à cette relation ;
- Pour les connecteurs : on utilise la table de vérité ;

Sémantique de la logique du premier ordre : évaluation

On définit la fonction d'évaluation d'une formule $\varphi \in F_\Sigma$ pour une réalisation $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ et une affectation θ, $\mathcal{E}(\varphi, \mathcal{M}, \theta)$:

- Pour un atome : $\mathcal{E}(P(t_1, \ldots, t_n), \mathcal{M}, \theta)$ vaut V si le n-uplet $(\mathrm{Val}_\mathcal{M}(t_1, \theta), \ldots, \mathrm{Val}_\mathcal{M}(t_n, \theta))$ appartient à la relation $\mathrm{I}(P)$, F s'il n'appartient pas à cette relation ;
- Pour les connecteurs : on utilise la table de vérité ;
- Quel que soit : $\mathcal{E}(\forall x\, \varphi, \mathcal{M}, \theta)$ vaut V si $\mathcal{E}(\varphi, \mathcal{M}, \theta\{x \mapsto a\})$ vaut V pour tout élément a de $\mathcal{D}$, F sinon ;

Sémantique de la logique du premier ordre : évaluation

On définit la fonction d'évaluation d'une formule $\varphi \in F_\Sigma$ pour une réalisation $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ et une affectation θ, $\mathcal{E}(\varphi, \mathcal{M}, \theta)$:

- Pour un atome : $\mathcal{E}(P(t_1, \ldots, t_n), \mathcal{M}, \theta)$ vaut V si le n-uplet $(\mathrm{Val}_\mathcal{M}(t_1, \theta), \ldots, \mathrm{Val}_\mathcal{M}(t_n, \theta))$ appartient à la relation $\mathrm{I}(P)$, F s'il n'appartient pas à cette relation ;
- Pour les connecteurs : on utilise la table de vérité ;
- Quel que soit : $\mathcal{E}(\forall x\, \varphi, \mathcal{M}, \theta)$ vaut V si $\mathcal{E}(\varphi, \mathcal{M}, \theta\{x \mapsto a\})$ vaut V pour tout élément a de $\mathcal{D}$, F sinon ;
- Il existe : $\mathcal{E}(\exists x\, \varphi, \mathcal{M}, \theta)$ vaut V si $\mathcal{E}(\varphi, \mathcal{M}, \theta\{x \mapsto a\})$ vaut V pour au moins un élément a de $\mathcal{D}$, F sinon.

Sémantique de la logique du premier ordre : évaluation

On définit la fonction d'évaluation d'une formule $\varphi \in F_\Sigma$ pour une réalisation $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ et une affectation θ, $\mathcal{E}(\varphi, \mathcal{M}, \theta)$:

- Pour un atome : $\mathcal{E}(P(t_1, \ldots, t_n), \mathcal{M}, \theta)$ vaut V si le n-uplet $(\mathrm{Val}_\mathcal{M}(t_1, \theta), \ldots, \mathrm{Val}_\mathcal{M}(t_n, \theta))$ appartient à la relation $\mathrm{I}(P)$, F s'il n'appartient pas à cette relation ;
- Pour les connecteurs : on utilise la table de vérité ;
- Quel que soit : $\mathcal{E}(\forall x\, \varphi, \mathcal{M}, \theta)$ vaut V si $\mathcal{E}(\varphi, \mathcal{M}, \theta\{x \mapsto a\})$ vaut V pour tout élément a de $\mathcal{D}$, F sinon ;
- Il existe : $\mathcal{E}(\exists x\, \varphi, \mathcal{M}, \theta)$ vaut V si $\mathcal{E}(\varphi, \mathcal{M}, \theta\{x \mapsto a\})$ vaut V pour au moins un élément a de $\mathcal{D}$, F sinon.

Exemple : Soit $\Sigma = (\{f^2\}, \{P^2\})$, soit $\varphi = \forall y\, P(f(x, y), y)$.

Sémantique de la logique du premier ordre : évaluation

On définit la fonction d'évaluation d'une formule $\varphi \in F_\Sigma$ pour une réalisation $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ et une affectation θ, $\mathcal{E}(\varphi, \mathcal{M}, \theta)$:

- Pour un atome : $\mathcal{E}(P(t_1, \ldots, t_n), \mathcal{M}, \theta)$ vaut V si le n-uplet $(\mathrm{Val}_\mathcal{M}(t_1, \theta), \ldots, \mathrm{Val}_\mathcal{M}(t_n, \theta))$ appartient à la relation $\mathrm{I}(P)$, F s'il n'appartient pas à cette relation ;
- Pour les connecteurs : on utilise la table de vérité ;
- Quel que soit : $\mathcal{E}(\forall x\, \varphi, \mathcal{M}, \theta)$ vaut V si $\mathcal{E}(\varphi, \mathcal{M}, \theta\{x \mapsto a\})$ vaut V pour tout élément a de $\mathcal{D}$, F sinon ;
- Il existe : $\mathcal{E}(\exists x\, \varphi, \mathcal{M}, \theta)$ vaut V si $\mathcal{E}(\varphi, \mathcal{M}, \theta\{x \mapsto a\})$ vaut V pour au moins un élément a de $\mathcal{D}$, F sinon.

Exemple : Soit $\Sigma = (\{f^2\}, \{P^2\})$, soit $\varphi = \forall y\, P(f(x, y), y)$.

- Si $\mathcal{M} = (\mathbb{R}, \{f \mapsto +, P \mapsto \leqslant\})$, alors $\mathcal{E}(\varphi, \mathcal{M}, \theta)$ est V si et seulement si $\theta(x)$ est négatif ou nul.

Sémantique de la logique du premier ordre : évaluation

On définit la fonction d'évaluation d'une formule $\varphi \in F_\Sigma$ pour une réalisation $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ et une affectation θ, $\mathcal{E}(\varphi, \mathcal{M}, \theta)$:

- Pour un atome : $\mathcal{E}(P(t_1, \ldots, t_n), \mathcal{M}, \theta)$ vaut V si le n-uplet $(\mathrm{Val}_\mathcal{M}(t_1, \theta), \ldots, \mathrm{Val}_\mathcal{M}(t_n, \theta))$ appartient à la relation $\mathrm{I}(P)$, F s'il n'appartient pas à cette relation ;
- Pour les connecteurs : on utilise la table de vérité ;
- Quel que soit : $\mathcal{E}(\forall x\, \varphi, \mathcal{M}, \theta)$ vaut V si $\mathcal{E}(\varphi, \mathcal{M}, \theta\{x \mapsto a\})$ vaut V pour tout élément a de $\mathcal{D}$, F sinon ;
- Il existe : $\mathcal{E}(\exists x\, \varphi, \mathcal{M}, \theta)$ vaut V si $\mathcal{E}(\varphi, \mathcal{M}, \theta\{x \mapsto a\})$ vaut V pour au moins un élément a de $\mathcal{D}$, F sinon.

Exemple : Soit $\Sigma = (\{f^2\}, \{P^2\})$, soit $\varphi = \forall y\, P(f(x, y), y)$.

- Si $\mathcal{M} = (\mathbb{R}, \{f \mapsto +, P \mapsto \leqslant\})$, alors $\mathcal{E}(\varphi, \mathcal{M}, \theta)$ est V si et seulement si $\theta(x)$ est négatif ou nul.
- Si $\mathcal{M} = (\mathbb{N}, \{f \mapsto \times, P \mapsto =\})$, alors $\mathcal{E}(\varphi, \mathcal{M}, \theta)$ est V si et seulement si $\theta(x) = 1$.

Sémantique de la logique du premier ordre : évaluation

On définit la fonction d'évaluation d'une formule $\varphi \in F_\Sigma$ pour une réalisation $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ et une affectation θ, $\mathcal{E}(\varphi, \mathcal{M}, \theta)$:

- Pour un atome : $\mathcal{E}(P(t_1, \ldots, t_n), \mathcal{M}, \theta)$ vaut V si le n-uplet $(\mathrm{Val}_\mathcal{M}(t_1, \theta), \ldots, \mathrm{Val}_\mathcal{M}(t_n, \theta))$ appartient à la relation $\mathrm{I}(P)$, F s'il n'appartient pas à cette relation ;
- Pour les connecteurs : on utilise la table de vérité ;
- Quel que soit : $\mathcal{E}(\forall x\, \varphi, \mathcal{M}, \theta)$ vaut V si $\mathcal{E}(\varphi, \mathcal{M}, \theta\{x \mapsto a\})$ vaut V pour tout élément a de $\mathcal{D}$, F sinon ;
- Il existe : $\mathcal{E}(\exists x\, \varphi, \mathcal{M}, \theta)$ vaut V si $\mathcal{E}(\varphi, \mathcal{M}, \theta\{x \mapsto a\})$ vaut V pour au moins un élément a de $\mathcal{D}$, F sinon.

Exemple : Soit $\Sigma = (\{f^2\}, \{P^2\})$, soit $\varphi = \forall y\, P(f(x, y), y)$.

- Si $\mathcal{M} = (\mathbb{R}, \{f \mapsto +, P \mapsto \leqslant\})$, alors $\mathcal{E}(\varphi, \mathcal{M}, \theta)$ est V si et seulement si $\theta(x)$ est négatif ou nul.
- Si $\mathcal{M} = (\mathbb{N}, \{f \mapsto \times, P \mapsto =\})$, alors $\mathcal{E}(\varphi, \mathcal{M}, \theta)$ est V si et seulement si $\theta(x) = 1$.

Remarque : pour cette φ, $\mathcal{E}(\varphi, \mathcal{M}, \theta)$ ne dépend jamais de $\theta(y)$.

Variables libres, liées

Une occurrence d'une variable x dans une formule est dite liée si elle se trouve dans une sous-formule commençant par $\exists x$ ou $\forall x$. On dit aussi qu'elle est quantifiée existentiellement ou universellement suivant le cas.

Variables libres, liées

Une occurrence d'une variable x dans une formule est dite liée si elle se trouve dans une sous-formule commençant par $\exists x$ ou $\forall x$. On dit aussi qu'elle est quantifiée existentiellement ou universellement suivant le cas.

Une occurrence de x est dite libre si elle n'est pas liée. Exemple :

$$P(x) \wedge (\forall x \exists y\, P(x) \Rightarrow Q(y)) \Rightarrow Q(y)$$

Les occurrences rouges sont liées, les vertes sont libres.

Variables libres, liées

Une occurrence d'une variable x dans une formule est dite liée si elle se trouve dans une sous-formule commençant par $\exists x$ ou $\forall x$. On dit aussi qu'elle est quantifiée existentiellement ou universellement suivant le cas.

Une occurrence de x est dite libre si elle n'est pas liée. Exemple :

$$P(x) \wedge (\forall x \exists y\, P(x) \Rightarrow Q(y)) \Rightarrow Q(y)$$

Les occurrences rouges sont liées, les vertes sont libres.

En renommant les variables liées, on peut toujours obtenir une formule équivalente où elles sont distinctes des variables libres.

Variables libres, liées

Une occurrence d'une variable x dans une formule est dite liée si elle se trouve dans une sous-formule commençant par $\exists x$ ou $\forall x$. On dit aussi qu'elle est quantifiée existentiellement ou universellement suivant le cas.

Une occurrence de x est dite libre si elle n'est pas liée. Exemple :

$$P(x) \wedge (\forall x \exists y\, P(x) \Rightarrow Q(y)) \Rightarrow Q(y)$$

Les occurrences rouges sont liées, les vertes sont libres.

En renommant les variables liées, on peut toujours obtenir une formule équivalente où elles sont distinctes des variables libres. Ici :

$$P(x) \wedge (\forall z \exists w\, P(z) \Rightarrow Q(w)) \Rightarrow Q(y)$$

est équivalente à la formule de départ.

Formules ouvertes et fermées

Une formule est dite ouverte si elle comprend au moins une variable libre ; sinon elle est dite close ou fermée.

Formules ouvertes et fermées

Une formule est dite ouverte si elle comprend au moins une variable libre ; sinon elle est dite close ou fermée.

Soit φ une formule ouverte. Soient $x_1, \ldots, x_n$ ses variables libres. La formule $\forall x_1 \ldots \forall x_n \, \varphi$ est appelée clôture universelle de φ.
On la note $\forall * \varphi$.
La formule $\exists x_1 \ldots \exists x_n \, \varphi$ est appelée clôture existentielle de φ.
On la note $\exists * \varphi$.

Formules ouvertes et fermées

Une formule est dite ouverte si elle comprend au moins une variable libre ; sinon elle est dite close ou fermée.

Soit φ une formule ouverte. Soient $x_1, \ldots, x_n$ ses variables libres. La formule $\forall x_1 \ldots \forall x_n \, \varphi$ est appelée clôture universelle de φ.
On la note $\forall * \varphi$.
La formule $\exists x_1 \ldots \exists x_n \, \varphi$ est appelée clôture existentielle de φ.
On la note $\exists * \varphi$.

Propriété :

- Si φ est une formule fermée, $\mathcal{E}(\varphi, \mathcal{M}, \theta)$ ne dépend pas de θ.

Formules ouvertes et fermées

Une formule est dite ouverte si elle comprend au moins une variable libre ; sinon elle est dite close ou fermée.

Soit φ une formule ouverte. Soient $x_1, \ldots, x_n$ ses variables libres. La formule $\forall x_1 \ldots \forall x_n \, \varphi$ est appelée clôture universelle de φ.
On la note $\forall * \varphi$.
La formule $\exists x_1 \ldots \exists x_n \, \varphi$ est appelée clôture existentielle de φ.
On la note $\exists * \varphi$.

Propriété :

- Si φ est une formule fermée, $\mathcal{E}(\varphi, \mathcal{M}, \theta)$ ne dépend pas de θ.
- On peut donc définir la valeur de vérité d'une formule fermée en fonction uniquement d'une réalisation du langage : $\mathcal{E}(\varphi, \mathcal{M})$.

Formules ouvertes et fermées

Une formule est dite ouverte si elle comprend au moins une variable libre ; sinon elle est dite close ou fermée.

Soit φ une formule ouverte. Soient $x_1, \ldots, x_n$ ses variables libres. La formule $\forall x_1 \ldots \forall x_n \varphi$ est appelée clôture universelle de φ.
On la note $\forall * \varphi$.
La formule $\exists x_1 \ldots \exists x_n \varphi$ est appelée clôture existentielle de φ.
On la note $\exists * \varphi$.

Propriété :

- Si φ est une formule fermée, $\mathcal{E}(\varphi, \mathcal{M}, \theta)$ ne dépend pas de θ.
- On peut donc définir la valeur de vérité d'une formule fermée en fonction uniquement d'une réalisation du langage : $\mathcal{E}(\varphi, \mathcal{M})$.

L'ensemble des formules fermées du langage de signature Σ est noté $\overline{F_\Sigma}$.

Remarques diverses

Les symboles de fonctions d'arité 0 correspondent à des constantes et représentent un élément fixé de $\mathcal{D}$.
On les notera $a, b, c, \ldots$ sans parenthèses.

Les symboles de prédicats d'arité 0 sont des propositions atomiques. On omettra également les parenthèses vides.

Les symboles de prédicats d'arité 1 représentent des propriétés.

Remarques diverses

Les symboles de fonctions d'arité 0 correspondent à des constantes et représentent un élément fixé de $\mathcal{D}$.
On les notera $a, b, c, \ldots$ sans parenthèses.

Les symboles de prédicats d'arité 0 sont des propositions atomiques. On omettra également les parenthèses vides.

Les symboles de prédicats d'arité 1 représentent des propriétés.
Ils peuvent aussi servir à classer les objets de $\mathcal{D}$ en différentes catégories (ex. points et droites)
Exemple : « Par deux points passe toujours une droite » :

Remarques diverses

Les symboles de fonctions d'arité 0 correspondent à des constantes et représentent un élément fixé de $\mathcal{D}$.
On les notera $a, b, c, \ldots$ sans parenthèses.

Les symboles de prédicats d'arité 0 sont des propositions atomiques. On omettra également les parenthèses vides.

Les symboles de prédicats d'arité 1 représentent des propriétés.
Ils peuvent aussi servir à classer les objets de $\mathcal{D}$ en différentes catégories (ex. points et droites)
Exemple : « Par deux points passe toujours une droite » :
$\forall x \forall y \, \mathrm{Point}(x) \wedge \mathrm{Point}(y) \Rightarrow$
$\exists z \, \mathrm{Droite}(z) \wedge \mathrm{Passepar}(z, x) \wedge \mathrm{Passepar}(z, y)$.

Remarques diverses

Les symboles de fonctions d'arité 0 correspondent à des constantes et représentent un élément fixé de $\mathcal{D}$.
On les notera $a, b, c, \ldots$ sans parenthèses.

Les symboles de prédicats d'arité 0 sont des propositions atomiques. On omettra également les parenthèses vides.

Les symboles de prédicats d'arité 1 représentent des propriétés.
Ils peuvent aussi servir à classer les objets de $\mathcal{D}$ en différentes catégories (ex. points et droites)
Exemple : « Par deux points passe toujours une droite » :
$\forall x \forall y \, \mathrm{Point}(x) \wedge \mathrm{Point}(y) \Rightarrow$
$\exists z \, \mathrm{Droite}(z) \wedge \mathrm{Passepar}(z, x) \wedge \mathrm{Passepar}(z, y)$.

On parle souvent de relations pour les prédicats en général (même d'arité 0 ou 1).

Conséquence, équivalence

On dit que ψ est une conséquence de φ, et on note $\varphi \models \psi$, si : pour toute réalisation $\mathcal{M}$ et toute affectation θ telles que $\mathcal{E}(\varphi, \mathcal{M}, \theta) = \mathsf{V}$, on a également $\mathcal{E}(\psi, \mathcal{M}, \theta) = \mathsf{V}$.

Conséquence, équivalence

On dit que ψ est une conséquence de φ, et on note $\varphi \models \psi$, si : pour toute réalisation $\mathcal{M}$ et toute affectation θ telles que $\mathcal{E}(\varphi, \mathcal{M}, \theta) = \mathsf{V}$, on a également $\mathcal{E}(\psi, \mathcal{M}, \theta) = \mathsf{V}$.

φ et ψ sont dites équivalentes, $\varphi \equiv \psi$, si $\varphi \models \psi$ et $\psi \models \varphi$.

Conséquence, équivalence

On dit que ψ est une conséquence de φ, et on note $\varphi \models \psi$, si : pour toute réalisation $\mathcal{M}$ et toute affectation θ telles que $\mathcal{E}(\varphi, \mathcal{M}, \theta) = \mathsf{V}$, on a également $\mathcal{E}(\psi, \mathcal{M}, \theta) = \mathsf{V}$.

φ et ψ sont dites équivalentes, $\varphi \equiv \psi$, si $\varphi \models \psi$ et $\psi \models \varphi$.
On a donc $\varphi \equiv \psi$ si et seulement si, pour toute réalisation $\mathcal{M}$ et toute affectation θ, $\mathcal{E}(\varphi, \mathcal{M}, \theta) = \mathcal{E}(\psi, \mathcal{M}, \theta)$.

Ainsi deux formules équivalentes ont exactement la même signification et sont donc interchangeables dans n'importe quel contexte.

Quelques cas d'équivalence importants

- De Morgan, associativité, commutativité...

Quelques cas d'équivalence importants

- De Morgan, associativité, commutativité...
- Interversion des quantificateurs du même type :
  - $\forall x \forall y \, \varphi \equiv \forall y \forall x \, \varphi$
  - $\exists x \exists y \, \varphi \equiv \exists y \exists x \, \varphi$

Quelques cas d'équivalence importants

- De Morgan, associativité, commutativité...
- Interversion des quantificateurs du même type :
  - $\forall x \forall y \, \varphi \equiv \forall y \forall x \, \varphi$
  - $\exists x \exists y \, \varphi \equiv \exists y \exists x \, \varphi$
- Négation d'un quantificateur :
  - $\neg \exists x \, \varphi \equiv \forall x \, \neg \varphi$
  - $\neg \forall x \, \varphi \equiv \exists x \, \neg \varphi$

Quelques cas d'équivalence importants

- De Morgan, associativité, commutativité...
- Interversion des quantificateurs du même type :
  - $\forall x \forall y \, \varphi \equiv \forall y \forall x \, \varphi$
  - $\exists x \exists y \, \varphi \equiv \exists y \exists x \, \varphi$
- Négation d'un quantificateur :
  - $\neg \exists x \, \varphi \equiv \forall x \, \neg \varphi$
  - $\neg \forall x \, \varphi \equiv \exists x \, \neg \varphi$
- Factorisation d'un quantificateur :
  - $(\forall x \, \varphi) \wedge (\forall x \, \psi) \equiv \forall x \, (\varphi \wedge \psi)$
  - $(\exists x \, \varphi) \vee (\exists x \, \psi) \equiv \exists x \, (\varphi \vee \psi)$

Quelques cas d'équivalence importants

- De Morgan, associativité, commutativité...
- Interversion des quantificateurs du même type :
  - $\forall x \forall y \, \varphi \equiv \forall y \forall x \, \varphi$
  - $\exists x \exists y \, \varphi \equiv \exists y \exists x \, \varphi$
- Négation d'un quantificateur :
  - $\neg \exists x \, \varphi \equiv \forall x \, \neg \varphi$
  - $\neg \forall x \, \varphi \equiv \exists x \, \neg \varphi$
- Factorisation d'un quantificateur :
  - $(\forall x \, \varphi) \wedge (\forall x \, \psi) \equiv \forall x \, (\varphi \wedge \psi)$
  - $(\exists x \, \varphi) \vee (\exists x \, \psi) \equiv \exists x \, (\varphi \vee \psi)$
- Indépendance d'une variable : Si x n'apparaît pas dans φ :
  - $\forall x \, \varphi \equiv \varphi \equiv \exists x \, \varphi$.

Quelques cas d'équivalence importants

- De Morgan, associativité, commutativité...
- Interversion des quantificateurs du même type :
  - $\forall x \forall y \, \varphi \equiv \forall y \forall x \, \varphi$
  - $\exists x \exists y \, \varphi \equiv \exists y \exists x \, \varphi$
- Négation d'un quantificateur :
  - $\neg \exists x \, \varphi \equiv \forall x \, \neg \varphi$
  - $\neg \forall x \, \varphi \equiv \exists x \, \neg \varphi$
- Factorisation d'un quantificateur :
  - $(\forall x \, \varphi) \wedge (\forall x \, \psi) \equiv \forall x \, (\varphi \wedge \psi)$
  - $(\exists x \, \varphi) \vee (\exists x \, \psi) \equiv \exists x \, (\varphi \vee \psi)$
- Indépendance d'une variable : Si x n'apparaît pas dans φ :
  - $\forall x \, \varphi \equiv \varphi \equiv \exists x \, \varphi$.
- Renommage d'une variable : Si y est une variable n'apparaissant pas dans φ, soit $\varphi[x \mapsto y]$ la formule obtenue en remplaçant dans φ toutes les occurrences de x par y. On a :

$$\forall x \, \varphi \equiv \forall y \, \varphi[x \mapsto y] \quad \text{et} \quad \exists x \, \varphi \equiv \exists y \, \varphi[x \mapsto y].$$

Modèles, satisfaisabilité, validité

Contrairement aux notions de conséquence et équivalence, les notions suivantes ne sont définies que pour les formules fermées :

- Soit $\varphi \in \overline{F_\Sigma}$ et soit $\mathcal{M}$ une réalisation de Σ. On dit que $\mathcal{M}$ satisfait φ, noté $\mathcal{M} \models \varphi$, si $\mathcal{E}(\varphi, \mathcal{M}) = \mathsf{V}$.

Modèles, satisfaisabilité, validité

Contrairement aux notions de conséquence et équivalence, les notions suivantes ne sont définies que pour les formules fermées :

- Soit $\varphi \in \overline{F_\Sigma}$ et soit $\mathcal{M}$ une réalisation de Σ. On dit que $\mathcal{M}$ satisfait φ, noté $\mathcal{M} \models \varphi$, si $\mathcal{E}(\varphi, \mathcal{M}) = \mathsf{V}$.
  La réalisation $\mathcal{M}$ est appelée modèle de la formule φ.

Modèles, satisfaisabilité, validité

Contrairement aux notions de conséquence et équivalence, les notions suivantes ne sont définies que pour les formules fermées :

- Soit $\varphi \in \overline{F_\Sigma}$ et soit $\mathcal{M}$ une réalisation de Σ. On dit que $\mathcal{M}$ satisfait φ, noté $\mathcal{M} \models \varphi$, si $\mathcal{E}(\varphi, \mathcal{M}) = \mathsf{V}$.
  La réalisation $\mathcal{M}$ est appelée modèle de la formule φ.

- Une formule qui possède au moins un modèle est dite satisfaisable. Sinon elle est insatisfaisable.

- Une formule $\varphi \in \overline{F_\Sigma}$ est dite valide si toute réalisation de Σ est un modèle de φ. On note $\models \varphi$. Une formule valide est aussi appelée tautologie.
  Dans le cas contraire elle est dite invalide (ou falsifiable).

Relations entre les différentes notions

Soient φ et ψ deux formules quelconques d'un même langage.
ψ est une conséquence de φ si et seulement si la clôture universelle de la formule $\varphi \Rightarrow \psi$ est valide.

Relations entre les différentes notions

Soient φ et ψ deux formules quelconques d'un même langage.
ψ est une conséquence de φ si et seulement si la clôture universelle de la formule $\varphi \Rightarrow \psi$ est valide.
En d'autres termes :

$$\varphi \models \psi \text{ si et seulement si } \models \forall*(\varphi \Rightarrow \psi).$$

Relations entre les différentes notions

Soient φ et ψ deux formules quelconques d'un même langage.
ψ est une conséquence de φ si et seulement si la clôture universelle de la formule $\varphi \Rightarrow \psi$ est valide.
En d'autres termes :

$$\varphi \models \psi \text{ si et seulement si } \models \forall*(\varphi \Rightarrow \psi).$$

De même, $\varphi \equiv \psi$ si et seulement si $\models \forall*(\varphi \Leftrightarrow \psi)$.

Relations entre les différentes notions

Soient φ et ψ deux formules quelconques d'un même langage.
ψ est une conséquence de φ si et seulement si la clôture universelle de la formule $\varphi \Rightarrow \psi$ est valide.
En d'autres termes :

$$\varphi \models \psi \text{ si et seulement si } \models \forall*(\varphi \Rightarrow \psi).$$

De même, $\varphi \equiv \psi$ si et seulement si $\models \forall*(\varphi \Leftrightarrow \psi)$.

Pour des formules fermées, on a $\varphi \models \psi$ si et seulement si tout modèle de φ est un modèle de ψ, et $\varphi \equiv \psi$ si et seulement si φ et ψ ont exactement les mêmes modèles.

Raisonnements en logique du premier ordre

Comme toujours en logique, un raisonnement doit faire abstraction de la réalité et de l'interprétation des symboles.

Raisonnements en logique du premier ordre

Comme toujours en logique, un raisonnement doit faire abstraction de la réalité et de l'interprétation des symboles.

En particulier, on ne connaît pas l'univers du discours.

Raisonnements en logique du premier ordre

Comme toujours en logique, un raisonnement doit faire abstraction de la réalité et de l'interprétation des symboles.

En particulier, on ne connaît pas l'univers du discours.

Pour raisonner sur les variables, on n'utilise donc pas des valuations θ, qui dépendent de $\mathcal{D}$.

Raisonnements en logique du premier ordre

Comme toujours en logique, un raisonnement doit faire abstraction de la réalité et de l'interprétation des symboles.

En particulier, on ne connaît pas l'univers du discours.

Pour raisonner sur les variables, on n'utilise donc pas des valuations θ, qui dépendent de $\mathcal{D}$.

On utilise plutôt des substitutions, qui remplacent les variables non par des valeurs mais par des termes.

Raisonnements en logique du premier ordre

Comme toujours en logique, un raisonnement doit faire abstraction de la réalité et de l'interprétation des symboles.

En particulier, on ne connaît pas l'univers du discours.

Pour raisonner sur les variables, on n'utilise donc pas des valuations θ, qui dépendent de $\mathcal{D}$.

On utilise plutôt des substitutions, qui remplacent les variables non par des valeurs mais par des termes.

Exemple : $P(f(x,y),g(y),x)[x \mapsto g(z); y \mapsto f(g(z),z)] =$

Raisonnements en logique du premier ordre

Comme toujours en logique, un raisonnement doit faire abstraction de la réalité et de l'interprétation des symboles.

En particulier, on ne connaît pas l'univers du discours.

Pour raisonner sur les variables, on n'utilise donc pas des valuations θ, qui dépendent de $\mathcal{D}$.

On utilise plutôt des substitutions, qui remplacent les variables non par des valeurs mais par des termes.

Exemple : $P(f(x,y),g(y),x)[x \mapsto g(z); y \mapsto f(g(z),z)] = P(f(g(z),f(g(z),z)),g(f(g(z),z)),g(z))$

Raisonnements en logique du premier ordre

Comme toujours en logique, un raisonnement doit faire abstraction de la réalité et de l'interprétation des symboles.

En particulier, on ne connaît pas l'univers du discours.

Pour raisonner sur les variables, on n'utilise donc pas des valuations θ, qui dépendent de $\mathcal{D}$.

On utilise plutôt des substitutions, qui remplacent les variables non par des valeurs mais par des termes.

Exemple : $P(f(x,y),g(y),x)[x \mapsto g(z); y \mapsto f(g(z),z)] = P(f(g(z),f(g(z),z)),g(f(g(z),z)),g(z))$

On peut faire un tel remplacement syntaxique sans connaître $\mathcal{D}$, ni $\mathrm{I}(f)$, ni $\mathrm{I}(g)$.

Substitutions

On appelle substitution une fonction de $\mathcal{V}$ dans T_Σ,
c'est-à-dire un remplacement de certaines variables par des termes.

Substitutions

On appelle substitution une fonction de $\mathcal{V}$ dans T_Σ,
c'est-à-dire un remplacement de certaines variables par des termes.

Soit σ une substitution, on note $\mathrm{Dom}(\sigma)$ l'ensemble des variables que σ remplace (domaine de σ).

Substitutions

On appelle substitution une fonction de $\mathcal{V}$ dans T_Σ,
c'est-à-dire un remplacement de certaines variables par des termes.

Soit σ une substitution, on note $\mathrm{Dom}(\sigma)$ l'ensemble des variables que σ remplace (domaine de σ).

Notation : on peut noter $[x_1 \mapsto t_1; x_2 \mapsto t_2; x_3 \mapsto t_3]$ la substitution de domaine $\{x_1, x_2, x_3\}$ qui remplace x_1 par t_1 etc.

Substitutions

On appelle substitution une fonction de $\mathcal{V}$ dans T_Σ,
c'est-à-dire un remplacement de certaines variables par des termes.

Soit σ une substitution, on note $\mathrm{Dom}(\sigma)$ l'ensemble des variables que σ remplace (domaine de σ).

Notation : on peut noter $[x_1 \mapsto t_1; x_2 \mapsto t_2; x_3 \mapsto t_3]$ la substitution de domaine $\{x_1, x_2, x_3\}$ qui remplace x_1 par t_1 etc.

Soit σ une substitution et t un terme, on note $t\sigma$ le terme obtenu en remplaçant dans t chaque variable x par $\sigma(x)$.

Substitutions

On appelle substitution une fonction de $\mathcal{V}$ dans T_Σ,
c'est-à-dire un remplacement de certaines variables par des termes.

Soit σ une substitution, on note $\mathrm{Dom}(\sigma)$ l'ensemble des variables que σ remplace (domaine de σ).

Notation : on peut noter $[x_1 \mapsto t_1; x_2 \mapsto t_2; x_3 \mapsto t_3]$ la substitution de domaine $\{x_1, x_2, x_3\}$ qui remplace x_1 par t_1 etc.

Soit σ une substitution et t un terme, on note $t\sigma$ le terme obtenu en remplaçant dans t chaque variable x par $\sigma(x)$.
Définition inductive :

- si $t = x \in \mathcal{V}$ (variable), alors $t\sigma = \begin{cases} \sigma(x) & \text{si } x \in \mathrm{Dom}(\sigma) \\ x & \text{si } x \notin \mathrm{Dom}(\sigma) \end{cases}$.
- si $t = f(t_1 \dots t_n)$ (terme composé), alors $t\sigma = f(t_1\sigma \dots t_n\sigma)$.

Substitutions

On définit de la même manière $\varphi\sigma$ pour une formule φ sans quantificateur.

Substitutions

On définit de la même manière $\varphi\sigma$ pour une formule φ sans quantificateur.

Si φ contient un quantificateur, $\varphi\sigma$ ne remplace que les occurrences libres des variables.

Substitutions

On définit de la même manière $\varphi\sigma$ pour une formule φ sans quantificateur.

Si φ contient un quantificateur, $\varphi\sigma$ ne remplace que les occurrences libres des variables.

Définition formelle (inductive) :

- si $\varphi = P(t_1, \ldots, t_n)$ (atome), alors $\varphi\sigma = P(t_1\sigma, \ldots, t_n\sigma)$;
- si $\varphi = \neg\psi$, alors $\varphi\sigma = \neg\psi\sigma$;
- si $\varphi = \psi_1 * \psi_2$ (pour $* \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$) alors $\varphi\sigma = \psi_1\sigma * \psi_2\sigma$;
- si $\varphi = \forall x\, \psi$, alors $\varphi\sigma = \forall x\, \psi(\sigma\{x \mapsto x\})$;
- si $\varphi = \exists x\, \psi$, alors $\varphi\sigma = \exists x\, \psi(\sigma\{x \mapsto x\})$.

Un système formel pour la logique du premier ordre

Un système formel pour la logique du premier ordre

On peut étendre le système de Fitch (transp. 26) avec les règles suivantes pour les quantificateurs :

Un système formel pour la logique du premier ordre

On peut étendre le système de Fitch (transp. 26) avec les règles suivantes pour les quantificateurs :

- Instanciation (élimination de $\forall$) :

$$\mathsf{E}\forall\ \frac{\forall x\,\varphi}{\varphi[x \mapsto t]} \qquad \textbf{où } t \textbf{ ne contient aucune variable liée dans } \varphi$$

Un système formel pour la logique du premier ordre

On peut étendre le système de Fitch (transp. 26) avec les règles suivantes pour les quantificateurs :

- Instanciation (élimination de $\forall$) :

$$\mathsf{E}\forall\ \frac{\forall x\,\varphi}{\varphi[x \mapsto t]}$$ **où t ne contient aucune variable liée dans φ**

- Introduction de $\exists$:

$$\mathsf{I}\exists\ \frac{\varphi[x \mapsto t]}{\exists x\,\varphi}$$ **où t ne contient aucune variable liée dans φ**

Un système formel pour la logique du premier ordre

On peut étendre le système de Fitch (transp. 26) avec les règles suivantes pour les quantificateurs :

- Instanciation (élimination de $\forall$) :

$$\mathsf{E}\forall\ \frac{\forall x\,\varphi}{\varphi[x \mapsto t]} \qquad \textbf{où } t \textbf{ ne contient aucune variable liée dans } \varphi$$

- Introduction de $\exists$:

$$\mathsf{I}\exists\ \frac{\varphi[x \mapsto t]}{\exists x\,\varphi} \qquad \textbf{où } t \textbf{ ne contient aucune variable liée dans } \varphi$$

- Généralisation (introduction de $\forall$) :

$$\mathsf{I}\forall\ \frac{\varphi \qquad x \text{ n'apparaît dans aucune hypothèse}}{\forall x\,\varphi}$$

Un système formel pour la logique du premier ordre

On peut étendre le système de Fitch (transp. 26) avec les règles suivantes pour les quantificateurs :

- Instanciation (élimination de $\forall$) :

$$\mathsf{E}\forall\ \frac{\forall x\,\varphi}{\varphi[x \mapsto t]} \qquad \textbf{où } t \textbf{ ne contient aucune variable liée dans } \varphi$$

- Introduction de $\exists$:

$$\mathsf{I}\exists\ \frac{\varphi[x \mapsto t]}{\exists x\,\varphi} \qquad \textbf{où } t \textbf{ ne contient aucune variable liée dans } \varphi$$

- Généralisation (introduction de $\forall$) :

$$\mathsf{I}\forall\ \frac{\varphi \qquad x \text{ n'apparaît dans aucune hypothèse}}{\forall x\,\varphi}$$

- Élimination de $\exists$:

$$\mathsf{E}\exists\ \frac{\exists x\,\varphi \qquad \forall x\,(\varphi \Rightarrow \psi) \qquad \psi \text{ ne contient pas } x}{\psi}$$

Un système formel pour la logique du premier ordre

On peut étendre le système de Fitch (transp. 26) avec les règles suivantes pour les quantificateurs :

- Instanciation (élimination de $\forall$) :

$$\mathsf{E}\forall\ \frac{\forall x\,\varphi}{\varphi[x \mapsto t]} \qquad \textbf{où } t \textbf{ ne contient aucune variable liée dans } \varphi$$

- Introduction de $\exists$:

$$\mathsf{I}\exists\ \frac{\varphi[x \mapsto t]}{\exists x\,\varphi} \qquad \textbf{où } t \textbf{ ne contient aucune variable liée dans } \varphi$$

- Généralisation (introduction de $\forall$) :

$$\mathsf{I}\forall\ \frac{\varphi \qquad x \text{ n'apparaît dans aucune hypothèse}}{\forall x\,\varphi}$$

- Élimination de $\exists$:

$$\mathsf{E}\exists\ \frac{\exists x\,\varphi \qquad \forall x\,(\varphi \Rightarrow \psi) \qquad \psi \text{ ne contient pas } x}{\psi}$$

Le système ainsi obtenu est correct et complet. On a donc une relation de déduction $\vdash$ identique à la relation de conséquence $\models$ (théorème de complétude de Gödel).

Système de Fitch au premier ordre

Utilisation de la règle de généralisation dans une preuve :

1. On démarre une sous-preuve en introduisant une nouvelle variable : « Soit x quelconque »
2. On généralise x quand on quitte la sous-preuve

Système de Fitch au premier ordre

Utilisation de la règle de généralisation dans une preuve :

1. On démarre une sous-preuve en introduisant une nouvelle variable : « Soit x quelconque »
2. On généralise x quand on quitte la sous-preuve

Exemple : $(\forall x\, P(x,x)) \Rightarrow \forall z \exists y\, P(z,y)$:

Système de Fitch au premier ordre

Utilisation de la règle de généralisation dans une preuve :

1. On démarre une sous-preuve en introduisant une nouvelle variable : « Soit x quelconque »
2. On généralise x quand on quitte la sous-preuve

Exemple : $(\forall x\, P(x,x)) \Rightarrow \forall z \exists y\, P(z,y)$:

1	$\forall x\, P(x,x)$	
2	Soit z	
3	$P(z,z)$	E$\forall$ $(x \mapsto z)$, 1
4	$\exists y\, P(z,y)$	I$\exists$ $(y \mapsto z)$, 3
5	$\forall z \exists y\, P(z,y)$	I$\forall$, 2–4
6	$(\forall x\, P(x,x)) \Rightarrow \forall z \exists y\, P(z,y)$	I$\Rightarrow$, 1–5

Système de Fitch au premier ordre

Utilisation de la règle d'élimination de $\exists$ dans une preuve :

Exemple : on veut prouver $(\exists y \forall x\, P(x, y)) \Rightarrow \exists z\, P(z, z)$

Système de Fitch au premier ordre

Utilisation de la règle d'élimination de $\exists$ dans une preuve :
Exemple : on veut prouver $(\exists y \forall x\, P(x,y)) \Rightarrow \exists z\, P(z,z)$

1	$\exists y \forall x\, P(x,y)$	
2	Soit y	
3	$\forall x\, P(x,y)$	
4	$P(y,y)$	E$\forall$ $(x \mapsto y)$, 3
5	$\exists z\, P(z,z)$	I$\exists$ $(y \mapsto z)$, 4
6	$(\forall x\, P(x,y)) \Rightarrow \exists z\, P(z,z)$	II, 3–5
7	$\forall y\, ((\forall x\, P(x,y)) \Rightarrow \exists z\, P(z,z))$	I$\forall$, 2–6
8	$\exists z\, P(z,z)$	E$\exists$, 1, 7
9	$(\exists y \forall x\, P(x,y)) \Rightarrow \exists z\, P(z,z)$	II, 1–8

Théories du premier ordre

Une théorie (du premier ordre) est un ensemble de formules closes (d'un langage du premier ordre).

Théories du premier ordre

Une théorie (du premier ordre) est un ensemble de formules closes (d'un langage du premier ordre).

La plupart des théories mathématiques sont (ou peuvent être vues comme) des théories du premier ordre : théorie des groupes, des anneaux etc., théorie des ensembles...

Théories du premier ordre

Une théorie (du premier ordre) est un ensemble de formules closes (d'un langage du premier ordre).

La plupart des théories mathématiques sont (ou peuvent être vues comme) des théories du premier ordre : théorie des groupes, des anneaux etc., théorie des ensembles...

Une théorie décrit une classe de structures mathématiques ayant des propriétés similaires ; ces structures sont les modèles de la théorie.

Théories du premier ordre

Une théorie (du premier ordre) est un ensemble de formules closes (d'un langage du premier ordre).

La plupart des théories mathématiques sont (ou peuvent être vues comme) des théories du premier ordre : théorie des groupes, des anneaux etc., théorie des ensembles...

Une théorie décrit une classe de structures mathématiques ayant des propriétés similaires ; ces structures sont les modèles de la théorie.

Étant donnée une théorie $\mathcal{T}$, un sous-ensemble $A \subseteq \mathcal{T}$ est une axiomatisation de $\mathcal{T}$ si toute conséquence de $\mathcal{T}$ est une conséquence de A : $\mathcal{T} \models \psi$ si et seulement si $A \models \psi$.

Théories du premier ordre

Une théorie (du premier ordre) est un ensemble de formules closes (d'un langage du premier ordre).

La plupart des théories mathématiques sont (ou peuvent être vues comme) des théories du premier ordre : théorie des groupes, des anneaux etc., théorie des ensembles...

Une théorie décrit une classe de structures mathématiques ayant des propriétés similaires ; ces structures sont les modèles de la théorie.

Étant donnée une théorie $\mathcal{T}$, un sous-ensemble $A \subseteq \mathcal{T}$ est une axiomatisation de $\mathcal{T}$ si toute conséquence de $\mathcal{T}$ est une conséquence de A : $\mathcal{T} \models \psi$ si et seulement si $A \models \psi$.

Une théorie $\mathcal{T}$ est finiment axiomatisable si elle possède une axiomatisation finie. Elle est récursivement axiomatisable s'il existe une axiomatisation A de $\mathcal{T}$ et un algorithme capable de dire pour toute formule ψ si $\psi \in A$.

Théories du premier ordre

Une théorie (du premier ordre) est un ensemble de formules closes (d'un langage du premier ordre).

La plupart des théories mathématiques sont (ou peuvent être vues comme) des théories du premier ordre : théorie des groupes, des anneaux etc., théorie des ensembles...

Une théorie décrit une classe de structures mathématiques ayant des propriétés similaires ; ces structures sont les modèles de la théorie.

Étant donnée une théorie $\mathcal{T}$, un sous-ensemble $A \subseteq \mathcal{T}$ est une axiomatisation de $\mathcal{T}$ si toute conséquence de $\mathcal{T}$ est une conséquence de A : $\mathcal{T} \models \psi$ si et seulement si $A \models \psi$.

Une théorie $\mathcal{T}$ est finiment axiomatisable si elle possède une axiomatisation finie. Elle est récursivement axiomatisable s'il existe une axiomatisation A de $\mathcal{T}$ et un algorithme capable de dire pour toute formule ψ si $\psi \in A$.

En pratique on ne considère que des théories récursivement axiomatisables, et on les décrit par leur ensemble d'axiomes.

Théories des relations binaires

Exemples de théories simples et très utilisées : le regroupement de plusieurs propriétés d'une relation binaire pour définir des relations d'ordre, d'équivalence, etc.

Théories des relations binaires

Exemples de théories simples et très utilisées : le regroupement de plusieurs propriétés d'une relation binaire pour définir des relations d'ordre, d'équivalence, etc.

Soit P un prédicat d'arité 2, les formules universelles suivantes décrivent des propriétés classiques de la relation représentée par P :

Théories des relations binaires

Exemples de théories simples et très utilisées : le regroupement de plusieurs propriétés d'une relation binaire pour définir des relations d'ordre, d'équivalence, etc.

Soit P un prédicat d'arité 2, les formules universelles suivantes décrivent des propriétés classiques de la relation représentée par P :

Réflexivité $\forall x\, P(x, x)$

Théories des relations binaires

Exemples de théories simples et très utilisées : le regroupement de plusieurs propriétés d'une relation binaire pour définir des relations d'ordre, d'équivalence, etc.

Soit P un prédicat d'arité 2, les formules universelles suivantes décrivent des propriétés classiques de la relation représentée par P :

Réflexivité $\forall x\, P(x, x)$

Irréflexivité $\forall x\, \neg P(x, x)$

Théories des relations binaires

Exemples de théories simples et très utilisées : le regroupement de plusieurs propriétés d'une relation binaire pour définir des relations d'ordre, d'équivalence, etc.

Soit P un prédicat d'arité 2, les formules universelles suivantes décrivent des propriétés classiques de la relation représentée par P :

Réflexivité $\forall x\, P(x, x)$

Irréflexivité $\forall x\, \neg P(x, x)$

Symétrie $\forall x \forall y\, P(x, y) \Rightarrow P(y, x)$

Théories des relations binaires

Exemples de théories simples et très utilisées : le regroupement de plusieurs propriétés d'une relation binaire pour définir des relations d'ordre, d'équivalence, etc.

Soit P un prédicat d'arité 2, les formules universelles suivantes décrivent des propriétés classiques de la relation représentée par P :

Réflexivité $\forall x\, P(x, x)$

Irréflexivité $\forall x\, \neg P(x, x)$

Symétrie $\forall x \forall y\, P(x, y) \Rightarrow P(y, x)$

Transitivité $\forall x \forall y \forall z\, P(x, y) \wedge P(y, z) \Rightarrow P(x, z)$

Théories des relations binaires

Exemples de théories simples et très utilisées : le regroupement de plusieurs propriétés d'une relation binaire pour définir des relations d'ordre, d'équivalence, etc.

Soit P un prédicat d'arité 2, les formules universelles suivantes décrivent des propriétés classiques de la relation représentée par P :

Réflexivité $\forall x\, P(x, x)$

Irréflexivité $\forall x\, \neg P(x, x)$

Symétrie $\forall x \forall y\, P(x, y) \Rightarrow P(y, x)$

Transitivité $\forall x \forall y \forall z\, P(x, y) \wedge P(y, z) \Rightarrow P(x, z)$

Les relations transitives sont particulièrement importantes en pratique. On définit les théories suivantes :

Théories des relations binaires

Exemples de théories simples et très utilisées : le regroupement de plusieurs propriétés d'une relation binaire pour définir des relations d'ordre, d'équivalence, etc.

Soit P un prédicat d'arité 2, les formules universelles suivantes décrivent des propriétés classiques de la relation représentée par P :

Réflexivité $\forall x\, P(x,x)$

Irréflexivité $\forall x\, \neg P(x,x)$

Symétrie $\forall x \forall y\, P(x,y) \Rightarrow P(y,x)$

Transitivité $\forall x \forall y \forall z\, P(x,y) \wedge P(y,z) \Rightarrow P(x,z)$

Les relations transitives sont particulièrement importantes en pratique. On définit les théories suivantes :

Relation de préordre = relation transitive et réflexive

Théories des relations binaires

Exemples de théories simples et très utilisées : le regroupement de plusieurs propriétés d'une relation binaire pour définir des relations d'ordre, d'équivalence, etc.

Soit P un prédicat d'arité 2, les formules universelles suivantes décrivent des propriétés classiques de la relation représentée par P :

Réflexivité $\forall x\, P(x,x)$

Irréflexivité $\forall x\, \neg P(x,x)$

Symétrie $\forall x \forall y\, P(x,y) \Rightarrow P(y,x)$

Transitivité $\forall x \forall y \forall z\, P(x,y) \wedge P(y,z) \Rightarrow P(x,z)$

Les relations transitives sont particulièrement importantes en pratique. On définit les théories suivantes :

Relation de préordre = relation transitive et réflexive

Relation d'ordre strict = relation transitive et irréflexive

Théories des relations binaires

Exemples de théories simples et très utilisées : le regroupement de plusieurs propriétés d'une relation binaire pour définir des relations d'ordre, d'équivalence, etc.

Soit P un prédicat d'arité 2, les formules universelles suivantes décrivent des propriétés classiques de la relation représentée par P :

Réflexivité $\forall x\, P(x,x)$

Irréflexivité $\forall x\, \neg P(x,x)$

Symétrie $\forall x \forall y\, P(x,y) \Rightarrow P(y,x)$

Transitivité $\forall x \forall y \forall z\, P(x,y) \wedge P(y,z) \Rightarrow P(x,z)$

Les relations transitives sont particulièrement importantes en pratique. On définit les théories suivantes :

Relation de préordre = relation transitive et réflexive

Relation d'ordre strict = relation transitive et irréflexive

Relation d'équivalence = relation transitive, réflexive et symétrique

Théories des relations binaires

Exemples de théories simples et très utilisées : le regroupement de plusieurs propriétés d'une relation binaire pour définir des relations d'ordre, d'équivalence, etc.

Soit P un prédicat d'arité 2, les formules universelles suivantes décrivent des propriétés classiques de la relation représentée par P :

Réflexivité $\forall x\, P(x,x)$

Irréflexivité $\forall x\, \neg P(x,x)$

Symétrie $\forall x \forall y\, P(x,y) \Rightarrow P(y,x)$

Transitivité $\forall x \forall y \forall z\, P(x,y) \wedge P(y,z) \Rightarrow P(x,z)$

Les relations transitives sont particulièrement importantes en pratique. On définit les théories suivantes :

Relation de préordre = relation transitive et réflexive

Relation d'ordre strict = relation transitive et irréflexive

Relation d'équivalence = relation transitive, réflexive et symétrique

Que dire d'une relation transitive, irréflexive et symétrique ?

Définition de nouvelles relations

Si P est un préordre, on peut construire :

Définition de nouvelles relations

Si P est un préordre, on peut construire :

- une relation d'ordre strict P_{ord} définie par $\forall x \forall y \, P_{ord}(x, y) \Leftrightarrow P(x, y) \wedge \neg P(y, x)$

Définition de nouvelles relations

Si P est un préordre, on peut construire :

- une relation d'ordre strict P_{ord} définie par $\forall x \forall y \, P_{ord}(x, y) \Leftrightarrow P(x, y) \wedge \neg P(y, x)$
- une relation d'équivalence P_{eq} définie par $\forall x \forall y \, P_{eq}(x, y) \Leftrightarrow P(x, y) \wedge P(y, x)$

Définition de nouvelles relations

Si P est un préordre, on peut construire :

- une relation d'ordre strict P_{ord} définie par $\forall x \forall y \, P_{ord}(x, y) \Leftrightarrow P(x, y) \wedge \neg P(y, x)$
- une relation d'équivalence P_{eq} définie par $\forall x \forall y \, P_{eq}(x, y) \Leftrightarrow P(x, y) \wedge P(y, x)$

Si P est transitive mais pas réflexive, P_{ord} est encore un ordre strict. Construire un préordre à partir de P nécessite l'égalité (transparent suivant).

Définition de nouvelles relations

Si P est un préordre, on peut construire :

- une relation d'ordre strict P_{ord} définie par $\forall x \forall y \, P_{ord}(x, y) \Leftrightarrow P(x, y) \wedge \neg P(y, x)$
- une relation d'équivalence P_{eq} définie par $\forall x \forall y \, P_{eq}(x, y) \Leftrightarrow P(x, y) \wedge P(y, x)$

Si P est transitive mais pas réflexive, P_{ord} est encore un ordre strict. Construire un préordre à partir de P nécessite l'égalité (transparent suivant).

Et si P n'est pas transitive ?

Définition de nouvelles relations

Si P est un préordre, on peut construire :

- une relation d'ordre strict P_{ord} définie par $\forall x \forall y \, P_{ord}(x, y) \Leftrightarrow P(x, y) \wedge \neg P(y, x)$
- une relation d'équivalence P_{eq} définie par $\forall x \forall y \, P_{eq}(x, y) \Leftrightarrow P(x, y) \wedge P(y, x)$

Si P est transitive mais pas réflexive, P_{ord} est encore un ordre strict. Construire un préordre à partir de P nécessite l'égalité (transparent suivant).

Et si P n'est pas transitive ? Limite de la logique du premier ordre :

- on peut définir mathématiquement la fermeture transitive de P (ex. la relation « parent » donne la relation « ancêtre »)
- cette définition mathématique ne peut pas être exprimée par des formules du premier ordre.

Un prédicat particulier : l'égalité

La plupart des théories utilisent un prédicat binaire dont l'interprétation est imposée : $=$.
Par convention, pour une telle théorie on ne considère comme des modèles que des réalisations où l'interprétation de $=$ est la relation d'égalité entre les éléments du domaine.

Un prédicat particulier : l'égalité

La plupart des théories utilisent un prédicat binaire dont l'interprétation est imposée : $=$.
Par convention, pour une telle théorie on ne considère comme des modèles que des réalisations où l'interprétation de $=$ est la relation d'égalité entre les éléments du domaine.

Utiliser ce prédicat nécessite d'ajouter des règles de raisonnement supplémentaires pour prendre en compte les propriétés de $=$:

Un prédicat particulier : l'égalité

La plupart des théories utilisent un prédicat binaire dont l'interprétation est imposée : $=$.
Par convention, pour une telle théorie on ne considère comme des modèles que des réalisations où l'interprétation de $=$ est la relation d'égalité entre les éléments du domaine.

Utiliser ce prédicat nécessite d'ajouter des règles de raisonnement supplémentaires pour prendre en compte les propriétés de $=$:

$$\text{RE}\ \frac{}{t = t}\ \text{(réflexivité de l'égalité)}$$

Un prédicat particulier : l'égalité

La plupart des théories utilisent un prédicat binaire dont l'interprétation est imposée : $=$.
Par convention, pour une telle théorie on ne considère comme des modèles que des réalisations où l'interprétation de $=$ est la relation d'égalité entre les éléments du domaine.

Utiliser ce prédicat nécessite d'ajouter des règles de raisonnement supplémentaires pour prendre en compte les propriétés de $=$:

RE $\dfrac{}{t = t}$ (réflexivité de l'égalité)

PS $\dfrac{\varphi[x \mapsto t_1] \qquad t_1 = t_2}{\varphi[x \mapsto t_2]}$ **où t_1, t_2 ne contiennent aucune variable liée dans φ**
(propriété de substitution)

Un prédicat particulier : l'égalité

La plupart des théories utilisent un prédicat binaire dont l'interprétation est imposée : $=$.
Par convention, pour une telle théorie on ne considère comme des modèles que des réalisations où l'interprétation de $=$ est la relation d'égalité entre les éléments du domaine.

Utiliser ce prédicat nécessite d'ajouter des règles de raisonnement supplémentaires pour prendre en compte les propriétés de $=$:

RE $\dfrac{}{t = t}$ (réflexivité de l'égalité)

PS $\dfrac{\varphi[x \mapsto t_1] \qquad t_1 = t_2}{\varphi[x \mapsto t_2]}$ **où t_1, t_2 ne contiennent aucune variable liée dans φ**
(propriété de substitution)

Ces axiomes ont notamment pour conséquence que $=$ est une relation d'équivalence (exercice)

Égalité en pratique : exemples

1	$P(x, f(y), g(z, x))$
2	$g(z, x) = f(x)$

Égalité en pratique : exemples

1	$P(x, f(y), g(z, x))$	
2	$g(z, x) = f(x)$	
3	$P(x, f(y), f(x))$	PS, 1, 2

Égalité en pratique : exemples

1	$P(x, f(y), g(z, x))$	
2	$g(z, x) = f(x)$	
3	$P(x, f(y), f(x))$	PS, 1, 2

On n'est pas obligé de remplacer toutes les occurrences :

1	$P(x, f(y), f(x))$
2	$x = f(y)$

Égalité en pratique : exemples

1	$P(x, f(y), g(z, x))$	
2	$g(z, x) = f(x)$	
3	$P(x, f(y), f(x))$	PS, 1, 2

On n'est pas obligé de remplacer toutes les occurrences :

1	$P(x, f(y), f(x))$	
2	$x = f(y)$	
3	$P(f(y), f(y), f(x))$	PS, 1, 2

Égalité en pratique : exemples

1	$P(x, f(y), g(z, x))$	
2	$g(z, x) = f(x)$	
3	$P(x, f(y), f(x))$	PS, 1, 2

On n'est pas obligé de remplacer toutes les occurrences :

1	$P(x, f(y), f(x))$	
2	$x = f(y)$	
3	$P(f(y), f(y), f(x))$	PS, 1, 2
4	$P(x, f(y), f(f(y)))$	PS, 1, 2

Égalité en pratique : exemples

On peut aussi « appliquer la même opération des deux côtés du signe = » (réflexivité + substitution).

Égalité en pratique : exemples

On peut aussi « appliquer la même opération des deux côtés du signe = » (réflexivité + substitution).
Par exemple montrer que si $z = f(x)$ alors $g(z, x) = g(f(x), x)$:

$$1 \quad \big| \; z = f(x)$$

Égalité en pratique : exemples

On peut aussi « appliquer la même opération des deux côtés du signe = » (réflexivité + substitution).
Par exemple montrer que si $z = f(x)$ alors $g(z, x) = g(f(x), x)$:

1	$z = f(x)$	
2	$g(z, x) = g(z, x)$	RE
3	$g(z, x) = g(f(x, g(y)), x)$	PS, 2, 1

Égalité en pratique : exemples

On peut aussi « appliquer la même opération des deux côtés du signe = » (réflexivité + substitution).
Par exemple montrer que si $z = f(x)$ alors $g(z, x) = g(f(x), x)$:

1	$z = f(x)$	
2	$g(z, x) = g(z, x)$	RE
3	$g(z, x) = g(f(x, g(y)), x)$	PS, 2, 1

En pratique on peut fusionner :

1	$z = f(x)$	
2	$g(z, x) = g(f(x, g(y)), x)$	RE+PS, 1

Unicité, antisymétrie et relation d'ordre

L'égalité est nécessaire pour exprimer l'unicité.
Il existe au plus un objet qui vérifie P^1 se dit : $\exists y \forall x\, P(x) \Rightarrow x = y$
Il existe exactement un objet qui vérifie P^1 se dit donc :

$$\exists y\, P(y) \wedge \forall x\, P(x) \Rightarrow x = y$$

Unicité, antisymétrie et relation d'ordre

L'égalité est nécessaire pour exprimer l'unicité.
Il existe au plus un objet qui vérifie P^1 se dit : $\exists y \forall x\, P(x) \Rightarrow x = y$
Il existe exactement un objet qui vérifie P^1 se dit donc :

$$\exists y\, P(y) \wedge \forall x\, P(x) \Rightarrow x = y$$

Avec l'égalité, on peut définir une autre propriété des relations binaires :

Antisymétrie $\forall x \forall y\, P(x, y) \wedge P(y, x) \Rightarrow x = y$

Unicité, antisymétrie et relation d'ordre

L'égalité est nécessaire pour exprimer l'unicité.
Il existe au plus un objet qui vérifie P^1 se dit : $\exists y \forall x\, P(x) \Rightarrow x = y$
Il existe exactement un objet qui vérifie P^1 se dit donc :

$$\exists y\, P(y) \wedge \forall x\, P(x) \Rightarrow x = y$$

Avec l'égalité, on peut définir une autre propriété des relations binaires :

Antisymétrie $\forall x \forall y\, P(x, y) \wedge P(y, x) \Rightarrow x = y$

Remarque : un ordre strict est automatiquement antisymétrique (la partie à gauche de $\Rightarrow$ est toujours fausse).

Unicité, antisymétrie et relation d'ordre

L'égalité est nécessaire pour exprimer l'unicité.
Il existe au plus un objet qui vérifie P^1 se dit : $\exists y \forall x\, P(x) \Rightarrow x = y$
Il existe exactement un objet qui vérifie P^1 se dit donc :

$$\exists y\, P(y) \wedge \forall x\, P(x) \Rightarrow x = y$$

Avec l'égalité, on peut définir une autre propriété des relations binaires :

Antisymétrie $\forall x \forall y\, P(x, y) \wedge P(y, x) \Rightarrow x = y$

Remarque : un ordre strict est automatiquement antisymétrique (la partie à gauche de $\Rightarrow$ est toujours fausse).

Avec cette propriété on peut définir une relation d'ordre (non strict) : réflexive, transitive et antisymétrique.

Exemple de théorie : l'arithmétique de Peano

L'axiomatique de Peano permet de décrire la théorie des nombres entiers naturels.

Exemple de théorie : l'arithmétique de Peano

L'axiomatique de Peano permet de décrire la théorie des nombres entiers naturels.

Signature :

- Symboles de fonctions : $0^0, s^1, +^2, \times^2$
- Symbole de relation : uniquement $=$ (on peut étendre le langage avec $\leqslant^2$)

Exemple de théorie : l'arithmétique de Peano

L'axiomatique de Peano permet de décrire la théorie des nombres entiers naturels.

Signature :

- Symboles de fonctions : $0^0, s^1, +^2, \times^2$
- Symbole de relation : uniquement $=$ (on peut étendre le langage avec $\leqslant^2$)

Axiomes :

1. $\forall x \neg(s(x) = 0)$
2. $\forall x \neg(x = 0) \Rightarrow \exists y\, s(y) = x$
3. $\forall x \forall y\, (s(x) = s(y) \Rightarrow x = y)$

Exemple de théorie : l'arithmétique de Peano

L'axiomatique de Peano permet de décrire la théorie des nombres entiers naturels.

Signature :

- Symboles de fonctions : $0^0, s^1, +^2, \times^2$
- Symbole de relation : uniquement $=$ (on peut étendre le langage avec $\leqslant^2$)

Axiomes :

1. $\forall x \neg(s(x) = 0)$
2. $\forall x \neg(x = 0) \Rightarrow \exists y\, s(y) = x$
3. $\forall x \forall y\, (s(x) = s(y) \Rightarrow x = y)$
4. $\forall x\, x + 0 = x$
5. $\forall x \forall y\, x + s(y) = s(x + y)$

Exemple de théorie : l'arithmétique de Peano

L'axiomatique de Peano permet de décrire la théorie des nombres entiers naturels.

Signature :

- Symboles de fonctions : $0^0, s^1, +^2, \times^2$
- Symbole de relation : uniquement $=$ (on peut étendre le langage avec $\leqslant^2$)

Axiomes :

1. $\forall x \neg(s(x) = 0)$
2. $\forall x \neg(x = 0) \Rightarrow \exists y\, s(y) = x$
3. $\forall x \forall y\, (s(x) = s(y) \Rightarrow x = y)$
4. $\forall x\, x + 0 = x$
5. $\forall x \forall y\, x + s(y) = s(x + y)$
6. $\forall x\, x \times 0 = 0$
7. $\forall x \forall y\, x \times s(y) = (x \times y) + x$

Exemple de théorie : l'arithmétique de Peano

L'axiomatique de Peano permet de décrire la théorie des nombres entiers naturels.

Signature :

- Symboles de fonctions : $0^0, s^1, +^2, \times^2$
- Symbole de relation : uniquement $=$ (on peut étendre le langage avec $\leqslant^2$)

Axiomes :

1. $\forall x \neg(s(x) = 0)$
2. $\forall x \neg(x = 0) \Rightarrow \exists y\, s(y) = x$
3. $\forall x \forall y\, (s(x) = s(y) \Rightarrow x = y)$
4. $\forall x\, x + 0 = x$
5. $\forall x \forall y\, x + s(y) = s(x + y)$
6. $\forall x\, x \times 0 = 0$
7. $\forall x \forall y\, x \times s(y) = (x \times y) + x$
8. Schéma : pour toute formule φ dont x est une variable libre :

$$\varphi[x \mapsto 0] \wedge (\forall x\, (\varphi \Rightarrow \varphi[x \mapsto s(x)])) \Rightarrow \forall x\, \varphi$$

Exemple de démonstration dans l'axiomatique de Peano

On veut montrer la commutativité : $\forall x \forall y \, (x + y = y + x)$

Exemple de démonstration dans l'axiomatique de Peano

On veut montrer la commutativité : $\forall x \forall y \, (x + y = y + x)$

On va faire un raisonnement par récurrence sur la variable x :

Exemple de démonstration dans l'axiomatique de Peano

On veut montrer la commutativité : $\forall x \forall y \, (x + y = y + x)$

On va faire un raisonnement par récurrence sur la variable x :

- On montre la propriété vraie pour $x = 0$ (initialisation)

Exemple de démonstration dans l'axiomatique de Peano

On veut montrer la commutativité : $\forall x \forall y \, (x + y = y + x)$

On va faire un raisonnement par récurrence sur la variable x :

- On montre la propriété vraie pour $x = 0$ (initialisation)
- On suppose la propriété vraie pour un certain x et on montre qu'elle est alors vraie aussi pour $s(x)$ (hérédité)

Exemple de démonstration dans l'axiomatique de Peano

On veut montrer la commutativité : $\forall x \forall y \, (x + y = y + x)$

On va faire un raisonnement par récurrence sur la variable x :

- On montre la propriété vraie pour $x = 0$ (initialisation)
- On suppose la propriété vraie pour un certain x et on montre qu'elle est alors vraie aussi pour $s(x)$ (hérédité)
- On conclut que la propriété est vraie pour tout x en utilisant l'instance appropriée de l'axiome 8 (ici $\varphi \leftarrow \forall y \, (x + y = y + x)$).

Exemple de démonstration dans l'axiomatique de Peano

On veut montrer la commutativité : $\forall x \forall y \, (x + y = y + x)$

On va faire un raisonnement par récurrence sur la variable x :

- On montre la propriété vraie pour $x = 0$ (initialisation)
- On suppose la propriété vraie pour un certain x et on montre qu'elle est alors vraie aussi pour $s(x)$ (hérédité)
- On conclut que la propriété est vraie pour tout x en utilisant l'instance appropriée de l'axiome 8 (ici $\varphi \leftarrow \forall y \, (x + y = y + x)$).

On démontre d'abord des propriétés plus simples qu'on utilisera dans la démonstration principale :
Lemme 1 : $\forall x (0 + x = x)$
Lemme 2 : $\forall x \forall y \, (y + s(x) = s(y) + x)$

Preuve détaillée du lemme 1 en Fitch

On fait un raisonnement par récurrence (utilisation de l'axiome 8) avec $\varphi \leftarrow 0 + x = x$.

Preuve détaillée du lemme 1 en Fitch

On fait un raisonnement par récurrence (utilisation de l'axiome 8) avec $\varphi \leftarrow 0 + x = x$. L'instance de l'axiome est donc :
$(0 + 0 = 0) \wedge (\forall x\,(0 + x = x) \Rightarrow (0 + s(x) = s(x))) \Rightarrow \forall x\,(0 + x = x)$

Preuve détaillée du lemme 1 en Fitch

On fait un raisonnement par récurrence (utilisation de l'axiome 8) avec $\varphi \leftarrow 0 + x = x$. L'instance de l'axiome est donc :
$(0 + 0 = 0) \wedge (\forall x\,(0 + x = x) \Rightarrow (0 + s(x) = s(x))) \Rightarrow \forall x\,(0 + x = x)$

1	$0 + 0 = 0$ (initialisation)	E$\forall$ $(x \mapsto 0)$, ax. 4

Preuve détaillée du lemme 1 en Fitch

On fait un raisonnement par récurrence (utilisation de l'axiome 8) avec $\varphi \leftarrow 0 + x = x$. L'instance de l'axiome est donc :
$(0 + 0 = 0) \wedge (\forall x\,(0 + x = x) \Rightarrow (0 + s(x) = s(x))) \Rightarrow \forall x\,(0 + x = x)$

1	$0 + 0 = 0$ (initialisation)	E$\forall$ $(x \mapsto 0)$, ax. 4
2	Soit x (hérédité)	
3	$0 + x = x$ (hyp. de récurrence)	

Preuve détaillée du lemme 1 en Fitch

On fait un raisonnement par récurrence (utilisation de l'axiome 8) avec $\varphi \leftarrow 0 + x = x$. L'instance de l'axiome est donc :
$(0 + 0 = 0) \wedge (\forall x\,(0 + x = x) \Rightarrow (0 + s(x) = s(x))) \Rightarrow \forall x\,(0 + x = x)$

1	$0 + 0 = 0$ (initialisation)	E$\forall$ $(x \mapsto 0)$, ax. 4
2	Soit x (hérédité)	
3	$0 + x = x$ (hyp. de récurrence)	
4	$s(0 + x) = s(x)$	RE+PS, 3

Preuve détaillée du lemme 1 en Fitch

On fait un raisonnement par récurrence (utilisation de l'axiome 8) avec $\varphi \leftarrow 0 + x = x$. L'instance de l'axiome est donc :
$(0 + 0 = 0) \wedge (\forall x\,(0 + x = x) \Rightarrow (0 + s(x) = s(x))) \Rightarrow \forall x\,(0 + x = x)$

1	$0 + 0 = 0$ (initialisation)	E$\forall$ $(x \mapsto 0)$, ax. 4
2	Soit x (hérédité)	
3	$0 + x = x$ (hyp. de récurrence)	
4	$s(0 + x) = s(x)$	RE+PS, 3
5	$0 + s(x) = s(0 + x)$	E$\forall$ $(x \mapsto 0, y \mapsto x)$, ax. 5

Preuve détaillée du lemme 1 en Fitch

On fait un raisonnement par récurrence (utilisation de l'axiome 8) avec $\varphi \leftarrow 0 + x = x$. L'instance de l'axiome est donc :
$(0 + 0 = 0) \wedge (\forall x\,(0 + x = x) \Rightarrow (0 + s(x) = s(x))) \Rightarrow \forall x\,(0 + x = x)$

1	$0 + 0 = 0$ (initialisation)	E$\forall$ $(x \mapsto 0)$, ax. 4
2	Soit x (hérédité)	
3	$0 + x = x$ (hyp. de récurrence)	
4	$s(0 + x) = s(x)$	RE+PS, 3
5	$0 + s(x) = s(0 + x)$	E$\forall$ $(x \mapsto 0, y \mapsto x)$, ax. 5
6	$0 + s(x) = s(x)$	PS, 5, 4

Preuve détaillée du lemme 1 en Fitch

On fait un raisonnement par récurrence (utilisation de l'axiome 8) avec $\varphi \leftarrow 0 + x = x$. L'instance de l'axiome est donc :
$(0 + 0 = 0) \wedge (\forall x\,(0 + x = x) \Rightarrow (0 + s(x) = s(x))) \Rightarrow \forall x\,(0 + x = x)$

1	$0 + 0 = 0$ (initialisation)	E$\forall$ $(x \mapsto 0)$, ax. 4
2	Soit x (hérédité)	
3	$0 + x = x$ (hyp. de récurrence)	
4	$s(0 + x) = s(x)$	RE+PS, 3
5	$0 + s(x) = s(0 + x)$	E$\forall$ $(x \mapsto 0, y \mapsto x)$, ax. 5
6	$0 + s(x) = s(x)$	PS, 5, 4
7	$(0 + x = x) \Rightarrow (0 + s(x) = s(x))$	I$\Rightarrow$, 3–6
8	$\forall x\,(0 + x = x) \Rightarrow (0 + s(x) = s(x))$	I$\forall$, 2–7

Preuve détaillée du lemme 1 en Fitch

On fait un raisonnement par récurrence (utilisation de l'axiome 8) avec $\varphi \leftarrow 0 + x = x$. L'instance de l'axiome est donc :
$(0 + 0 = 0) \wedge (\forall x\,(0 + x = x) \Rightarrow (0 + s(x) = s(x))) \Rightarrow \forall x\,(0 + x = x)$

1	$0 + 0 = 0$ (initialisation)	E$\forall$ $(x \mapsto 0)$, ax. 4
2	Soit x (hérédité)	
3	$0 + x = x$ (hyp. de récurrence)	
4	$s(0 + x) = s(x)$	RE+PS, 3
5	$0 + s(x) = s(0 + x)$	E$\forall$ $(x \mapsto 0, y \mapsto x)$, ax. 5
6	$0 + s(x) = s(x)$	PS, 5, 4
7	$(0 + x = x) \Rightarrow (0 + s(x) = s(x))$	II, 3–6
8	$\forall x\,(0 + x = x) \Rightarrow (0 + s(x) = s(x))$	I$\forall$, 2–7
9	$(1) \wedge (8)$ (init. + héréd.)	IC, 1, 8
10	$\forall x\,(0 + x = x)$ (concl. récurrence)	EI, ax. 8, 9

Preuve du lemme 2 (abrégée)

On veut montrer $\forall x \forall y\, y + s(x) = s(y) + x$.

Récurrence sur x (ax. 8, $\varphi \leftarrow \forall y\, y + s(x) = s(y) + x$) :

Preuve du lemme 2 (abrégée)

On veut montrer $\forall x \forall y\, y + s(x) = s(y) + x$.

Récurrence sur x (ax. 8, $\varphi \leftarrow \forall y\, y + s(x) = s(y) + x$) :

- Initialisation : pour tout y :
  - $y + s(0) = s(y + 0)$ (E$\forall$ $(x \mapsto y)$, ax. 5)

Preuve du lemme 2 (abrégée)

On veut montrer $\forall x \forall y\, y + s(x) = s(y) + x$.

Récurrence sur x (ax. 8, $\varphi \leftarrow \forall y\, y + s(x) = s(y) + x$) :

- Initialisation : pour tout y :
  - $y + s(0) = s(y + 0)$ (E$\forall$ ($x \mapsto y$), ax. 5)
  - $y + s(0) = s(y)$ (PS, E$\forall$ ax. 4 ($x \mapsto y$))

Preuve du lemme 2 (abrégée)

On veut montrer $\forall x \forall y\, y + s(x) = s(y) + x$.

Récurrence sur x (ax. 8, $\varphi \leftarrow \forall y\, y + s(x) = s(y) + x$) :

- Initialisation : pour tout y :
  - $y + s(0) = s(y + 0)$ (E$\forall$ ($x \mapsto y$), ax. 5)
  - $y + s(0) = s(y)$ (PS, E$\forall$ ax. 4 ($x \mapsto y$))
  - $y + s(0) = s(y) + 0$ (PS, E$\forall$ ax. 4 ($x \mapsto s(y)$))

Preuve du lemme 2 (abrégée)

On veut montrer $\forall x \forall y\, y + s(x) = s(y) + x$.

Récurrence sur x (ax. 8, $\varphi \leftarrow \forall y\, y + s(x) = s(y) + x$) :

- Initialisation : pour tout y :
  - $y + s(0) = s(y + 0)$ (E$\forall$ $(x \mapsto y)$, ax. 5)
  - $y + s(0) = s(y)$ (PS, E$\forall$ ax. 4 $(x \mapsto y)$)
  - $y + s(0) = s(y) + 0$ (PS, E$\forall$ ax. 4 $(x \mapsto s(y))$)
- Hérédité : supposons $y + s(x) = s(y) + x$. On a :

Preuve du lemme 2 (abrégée)

On veut montrer $\forall x \forall y\, y + s(x) = s(y) + x$.

Récurrence sur x (ax. 8, $\varphi \leftarrow \forall y\, y + s(x) = s(y) + x$) :

- Initialisation : pour tout y :
  - $y + s(0) = s(y + 0)$ (E$\forall$ $(x \mapsto y)$, ax. 5)
  - $y + s(0) = s(y)$ (PS, E$\forall$ ax. 4 $(x \mapsto y)$)
  - $y + s(0) = s(y) + 0$ (PS, E$\forall$ ax. 4 $(x \mapsto s(y))$)
- Hérédité : supposons $y + s(x) = s(y) + x$. On a :
  - $y + s(s(x)) = s(y + s(x))$ (E$\forall$ $(x \mapsto y, y \mapsto s(x))$, ax. 5)

Preuve du lemme 2 (abrégée)

On veut montrer $\forall x \forall y\, y + s(x) = s(y) + x$.

Récurrence sur x (ax. 8, $\varphi \leftarrow \forall y\, y + s(x) = s(y) + x$) :

- Initialisation : pour tout y :
  - $y + s(0) = s(y + 0)$ (E$\forall$ $(x \mapsto y)$, ax. 5)
  - $y + s(0) = s(y)$ (PS, E$\forall$ ax. 4 $(x \mapsto y)$)
  - $y + s(0) = s(y) + 0$ (PS, E$\forall$ ax. 4 $(x \mapsto s(y))$)

- Hérédité : supposons $y + s(x) = s(y) + x$. On a :
  - $y + s(s(x)) = s(y + s(x))$ (E$\forall$ $(x \mapsto y, y \mapsto s(x))$, ax. 5)
  - $y + s(s(x)) = s(s(y) + x)$ (PS, hyp. de récurrence)

Preuve du lemme 2 (abrégée)

On veut montrer $\forall x \forall y\, y + s(x) = s(y) + x$.

Récurrence sur x (ax. 8, $\varphi \leftarrow \forall y\, y + s(x) = s(y) + x$) :

- Initialisation : pour tout y :
  - $y + s(0) = s(y + 0)$ (E$\forall$ $(x \mapsto y)$, ax. 5)
  - $y + s(0) = s(y)$ (PS, E$\forall$ ax. 4 $(x \mapsto y)$)
  - $y + s(0) = s(y) + 0$ (PS, E$\forall$ ax. 4 $(x \mapsto s(y))$)

- Hérédité : supposons $y + s(x) = s(y) + x$. On a :
  - $y + s(s(x)) = s(y + s(x))$ (E$\forall$ $(x \mapsto y, y \mapsto s(x))$, ax. 5)
  - $y + s(s(x)) = s(s(y) + x)$ (PS, hyp. de récurrence)
  - $y + s(s(x)) = s(y) + s(x)$ (PS, E$\forall$ ax. 5 $(x \mapsto s(y), y \mapsto x)$)

Preuve de la commutativité (abrégée)

On montre maintenant $\forall x \forall y\, x + y = y + x$ par récurrence sur x :

Preuve de la commutativité (abrégée)

On montre maintenant $\forall x \forall y\, x + y = y + x$ par récurrence sur x :

- Initialisation : $0 + y = y = y + 0$ (lemme 1, ax. 4)

Preuve de la commutativité (abrégée)

On montre maintenant $\forall x \forall y\, x + y = y + x$ par récurrence sur x :

- Initialisation : $0 + y = y = y + 0$ (lemme 1, ax. 4)
- Hérédité : on suppose $x + y = y + x$. On a :

Preuve de la commutativité (abrégée)

On montre maintenant $\forall x \forall y\, x + y = y + x$ par récurrence sur x :

- Initialisation : $0 + y = y = y + 0$ (lemme 1, ax. 4)
- Hérédité : on suppose $x + y = y + x$. On a :
  - $s(x) + y = x + s(y)$ (E$\forall$ $(x \mapsto y, y \mapsto x)$, lemme 2)

Preuve de la commutativité (abrégée)

On montre maintenant $\forall x \forall y\, x + y = y + x$ par récurrence sur x :

- Initialisation : $0 + y = y = y + 0$ (lemme 1, ax. 4)
- Hérédité : on suppose $x + y = y + x$. On a :
  - $s(x) + y = x + s(y)$ (E$\forall$ $(x \mapsto y, y \mapsto x)$, lemme 2)
  - $s(x) + y = s(x + y)$ (PS, E$\forall$ ax. 5)

Preuve de la commutativité (abrégée)

On montre maintenant $\forall x \forall y\, x + y = y + x$ par récurrence sur x :

- Initialisation : $0 + y = y = y + 0$ (lemme 1, ax. 4)
- Hérédité : on suppose $x + y = y + x$. On a :
  - $s(x) + y = x + s(y)$ (E$\forall$ $(x \mapsto y, y \mapsto x)$, lemme 2)
  - $s(x) + y = s(x + y)$ (PS, E$\forall$ ax. 5)
  - $s(x) + y = s(y + x)$ (PS, hyp. de récurrence)

Preuve de la commutativité (abrégée)

On montre maintenant $\forall x \forall y\, x + y = y + x$ par récurrence sur x :

- Initialisation : $0 + y = y = y + 0$ (lemme 1, ax. 4)

- Hérédité : on suppose $x + y = y + x$. On a :
  - $s(x) + y = x + s(y)$ (E$\forall$ $(x \mapsto y, y \mapsto x)$, lemme 2)
  - $s(x) + y = s(x + y)$ (PS, E$\forall$ ax. 5)
  - $s(x) + y = s(y + x)$ (PS, hyp. de récurrence)
  - $s(x) + y = y + s(x)$ (PS, E$\forall$ ax. $5(y \mapsto x, x \mapsto y)$)

Cohérence d'une théorie

Une théorie est cohérente si on ne peut pas en déduire une contradiction.

Cohérence d'une théorie

Une théorie est cohérente si on ne peut pas en déduire une contradiction.

D'après le théorème de complétude, c'est équivalent à dire que la théorie possède un modèle. En effet :

Cohérence d'une théorie

Une théorie est cohérente si on ne peut pas en déduire une contradiction.

D'après le théorème de complétude, c'est équivalent à dire que la théorie possède un modèle. En effet :

- Supposons que $\mathcal{T}$ n'a pas de modèle (est insatisfaisable)
- Alors $\mathcal{T} \models \varphi$ pour toute formule φ.
- Par complétude, $\mathcal{T} \vdash \varphi$ pour toute φ.
- Vrai notamment pour φ contradictoire.

Cohérence d'une théorie

Une théorie est cohérente si on ne peut pas en déduire une contradiction.

D'après le théorème de complétude, c'est équivalent à dire que la théorie possède un modèle. En effet :

- Supposons que $\mathcal{T}$ n'a pas de modèle (est insatisfaisable)
- Alors $\mathcal{T} \models \varphi$ pour toute formule φ.
- Par complétude, $\mathcal{T} \vdash \varphi$ pour toute φ.
- Vrai notamment pour φ contradictoire.

Par contraposition, si $\mathcal{T}$ ne permet pas de prouver une contradiction, elle possède un modèle.

Le théorème de complétude peut donc s'énoncer :
« toute théorie cohérente possède un modèle. »

Indécidabilité logique, complétude d'une théorie

Attention : ne pas confondre avec la décidabilité algorithmique et la complétude d'un système formel...

Une formule close φ est indécidable dans une théorie $\mathcal{T}$ si on n'a ni $\mathcal{T} \vdash \varphi$ ni $\mathcal{T} \vdash \neg\varphi$.

Indécidabilité logique, complétude d'une théorie

Attention : ne pas confondre avec la décidabilité algorithmique et la complétude d'un système formel...

Une formule close φ est indécidable dans une théorie $\mathcal{T}$ si on n'a ni $\mathcal{T} \vdash \varphi$ ni $\mathcal{T} \vdash \neg\varphi$.
D'après le théorème de complétude, cela signifie que $\mathcal{T}$ possède des modèles où φ est vraie et des modèles où φ est fausse.

Une théorie $\mathcal{T}$ est dite complète si aucune formule n'est indécidable dans cette théorie.

Indécidabilité logique, complétude d'une théorie

Attention : ne pas confondre avec la décidabilité algorithmique et la complétude d'un système formel...

Une formule close φ est indécidable dans une théorie $\mathcal{T}$ si on n'a ni $\mathcal{T} \vdash \varphi$ ni $\mathcal{T} \vdash \neg\varphi$.
D'après le théorème de complétude, cela signifie que $\mathcal{T}$ possède des modèles où φ est vraie et des modèles où φ est fausse.

Une théorie $\mathcal{T}$ est dite complète si aucune formule n'est indécidable dans cette théorie.

Remarque : une théorie complète peut néanmoins avoir plusieurs modèles différents, mais leurs différences ne sont pas exprimables dans le langage.

Indécidabilité logique, complétude d'une théorie

Attention : ne pas confondre avec la décidabilité algorithmique et la complétude d'un système formel...

Une formule close φ est indécidable dans une théorie $\mathcal{T}$ si on n'a ni $\mathcal{T} \vdash \varphi$ ni $\mathcal{T} \vdash \neg\varphi$.
D'après le théorème de complétude, cela signifie que $\mathcal{T}$ possède des modèles où φ est vraie et des modèles où φ est fausse.

Une théorie $\mathcal{T}$ est dite complète si aucune formule n'est indécidable dans cette théorie.
Remarque : une théorie complète peut néanmoins avoir plusieurs modèles différents, mais leurs différences ne sont pas exprimables dans le langage.

Théorème d'incomplétude de Gödel : toute théorie $\mathcal{T}$ récursivement axiomatisable, cohérente, et contenant l'arithmétique de Peano, est incomplète.

Indécidabilité logique, complétude d'une théorie

Attention : ne pas confondre avec la décidabilité algorithmique et la complétude d'un système formel...

Une formule close φ est indécidable dans une théorie $\mathcal{T}$ si on n'a ni $\mathcal{T} \vdash \varphi$ ni $\mathcal{T} \vdash \neg\varphi$.
D'après le théorème de complétude, cela signifie que $\mathcal{T}$ possède des modèles où φ est vraie et des modèles où φ est fausse.

Une théorie $\mathcal{T}$ est dite complète si aucune formule n'est indécidable dans cette théorie.
Remarque : une théorie complète peut néanmoins avoir plusieurs modèles différents, mais leurs différences ne sont pas exprimables dans le langage.

Théorème d'incomplétude de Gödel : toute théorie $\mathcal{T}$ récursivement axiomatisable, cohérente, et contenant l'arithmétique de Peano, est incomplète.

Remarque : L'arithmétique de Presburger, définie comme celle de Peano mais sans la multiplication, est complète.

Idée de la preuve de l'incomplétude

Soit $\mathcal{T}$ une théorie sur le langage F_Σ, vérifiant les hypothèses.
Puisque $\mathcal{T}$ contient l'arithmétique, un de ses modèles est $(\mathbb{N}, +, \times, =)$.

Idée de la preuve de l'incomplétude

Soit $\mathcal{T}$ une théorie sur le langage F_Σ, vérifiant les hypothèses.
Puisque $\mathcal{T}$ contient l'arithmétique, un de ses modèles est $(\mathbb{N}, +, \times, =)$.
On construit une formule φ telle que $\mathbb{N} \models \varphi$ mais $\mathcal{T} \not\vdash \varphi$:

Idée de la preuve de l'incomplétude

Soit $\mathcal{T}$ une théorie sur le langage F_Σ, vérifiant les hypothèses.
Puisque $\mathcal{T}$ contient l'arithmétique, un de ses modèles est $(\mathbb{N}, +, \times, =)$.
On construit une formule φ telle que $\mathbb{N} \models \varphi$ mais $\mathcal{T} \not\vdash \varphi$:

- On code les formules de F_Σ par des nombres.

Idée de la preuve de l'incomplétude

Soit $\mathcal{T}$ une théorie sur le langage F_Σ, vérifiant les hypothèses.
Puisque $\mathcal{T}$ contient l'arithmétique, un de ses modèles est $(\mathbb{N}, +, \times, =)$.
On construit une formule φ telle que $\mathbb{N} \models \varphi$ mais $\mathcal{T} \not\vdash \varphi$:

- On code les formules de F_Σ par des nombres.
- On code également les déductions du système par des nombres.

Idée de la preuve de l'incomplétude

Soit $\mathcal{T}$ une théorie sur le langage F_Σ, vérifiant les hypothèses.
Puisque $\mathcal{T}$ contient l'arithmétique, un de ses modèles est $(\mathbb{N}, +, \times, =)$.
On construit une formule φ telle que $\mathbb{N} \models \varphi$ mais $\mathcal{T} \nvdash \varphi$:

- On code les formules de F_Σ par des nombres.
- On code également les déductions du système par des nombres.
- On exprime la propriété « tel nombre représente une preuve de telle formule à partir de $\mathcal{T}$ » par un énoncé arithmétique.

Idée de la preuve de l'incomplétude

Soit $\mathcal{T}$ une théorie sur le langage F_Σ, vérifiant les hypothèses.
Puisque $\mathcal{T}$ contient l'arithmétique, un de ses modèles est $(\mathbb{N}, +, \times, =)$.
On construit une formule φ telle que $\mathbb{N} \models \varphi$ mais $\mathcal{T} \nvdash \varphi$:

- On code les formules de F_Σ par des nombres.
- On code également les déductions du système par des nombres.
- On exprime la propriété « tel nombre représente une preuve de telle formule à partir de $\mathcal{T}$ » par un énoncé arithmétique.
- Cet énoncé peut lui-même être traduit par une formule de F_Σ, et donc par un nombre.

Idée de la preuve de l'incomplétude

Soit $\mathcal{T}$ une théorie sur le langage F_Σ, vérifiant les hypothèses.
Puisque $\mathcal{T}$ contient l'arithmétique, un de ses modèles est $(\mathbb{N}, +, \times, =)$.
On construit une formule φ telle que $\mathbb{N} \models \varphi$ mais $\mathcal{T} \nvdash \varphi$:

- On code les formules de F_Σ par des nombres.
- On code également les déductions du système par des nombres.
- On exprime la propriété « tel nombre représente une preuve de telle formule à partir de $\mathcal{T}$ » par un énoncé arithmétique.
- Cet énoncé peut lui-même être traduit par une formule de F_Σ, et donc par un nombre.
- On construit un énoncé auto-référent de la forme « aucun nombre ne représente une preuve de cet énoncé. »

Idée de la preuve de l'incomplétude

Soit $\mathcal{T}$ une théorie sur le langage F_Σ, vérifiant les hypothèses.
Puisque $\mathcal{T}$ contient l'arithmétique, un de ses modèles est $(\mathbb{N}, +, \times, =)$.
On construit une formule φ telle que $\mathbb{N} \models \varphi$ mais $\mathcal{T} \nvdash \varphi$:

- On code les formules de F_Σ par des nombres.
- On code également les déductions du système par des nombres.
- On exprime la propriété « tel nombre représente une preuve de telle formule à partir de $\mathcal{T}$ » par un énoncé arithmétique.
- Cet énoncé peut lui-même être traduit par une formule de F_Σ, et donc par un nombre.
- On construit un énoncé auto-référent de la forme « aucun nombre ne représente une preuve de cet énoncé. »
- Si cet énoncé était déductible à partir de $\mathcal{T}$, il existerait un nombre qui code sa preuve, mais alors l'énoncé serait faux dans $\mathbb{N}$. Or $\mathbb{N}$ est un modèle de $\mathcal{T}$: impossible.

Idée de la preuve de l'incomplétude

Soit $\mathcal{T}$ une théorie sur le langage F_Σ, vérifiant les hypothèses.
Puisque $\mathcal{T}$ contient l'arithmétique, un de ses modèles est $(\mathbb{N}, +, \times, =)$.
On construit une formule φ telle que $\mathbb{N} \models \varphi$ mais $\mathcal{T} \nvdash \varphi$:

- On code les formules de F_Σ par des nombres.
- On code également les déductions du système par des nombres.
- On exprime la propriété « tel nombre représente une preuve de telle formule à partir de $\mathcal{T}$ » par un énoncé arithmétique.
- Cet énoncé peut lui-même être traduit par une formule de F_Σ, et donc par un nombre.
- On construit un énoncé auto-référent de la forme « aucun nombre ne représente une preuve de cet énoncé. »
- Si cet énoncé était déductible à partir de $\mathcal{T}$, il existerait un nombre qui code sa preuve, mais alors l'énoncé serait faux dans $\mathbb{N}$. Or $\mathbb{N}$ est un modèle de $\mathcal{T}$: impossible.
- Donc cet énoncé n'est pas démontrable dans $\mathcal{T}$, mais alors il est vrai dans $\mathbb{N}$.

Satisfaisabilité dans la logique du premier ordre

Validité ou satisfaisabilité d'un ensemble de formules :
problème central, comme dans la logique propositionnelle.

Nous allons maintenant voir comment il est possible d'adapter la méthode de résolution à la logique du premier ordre, dans un premier temps pour les formules de la forme $\forall * \varphi$, puis pour toutes les formules.

Formes normales

- Une formule est en forme normale négative si $\neg$ n'apparaît que devant des formules atomiques.
  - Lois de De Morgan + changement de quantificateur
- Un littéral est une formule atomique éventuellement précédée de $\neg$.
- On définit comme pour la LP formes conjonctive et disjonctive.

Formes normales

- Une formule est en forme normale négative si $\neg$ n'apparaît que devant des formules atomiques.
  - Lois de De Morgan + changement de quantificateur
- Un littéral est une formule atomique éventuellement précédée de $\neg$.
- On définit comme pour la LP formes conjonctive et disjonctive.
- L'équivalence permet aussi de déplacer les quantificateurs.
  Une formule est sous forme normale prénexe si tous les quantificateurs sont au début : $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \varphi$ où les Q_i sont $\exists$ ou $\forall$ et φ ne contient pas de quantificateur.

Formes normales

- Une formule est en forme normale négative si $\neg$ n'apparaît que devant des formules atomiques.
  - Lois de De Morgan + changement de quantificateur
- Un littéral est une formule atomique éventuellement précédée de $\neg$.
- On définit comme pour la LP formes conjonctive et disjonctive.
- L'équivalence permet aussi de déplacer les quantificateurs.
  Une formule est sous forme normale prénexe si tous les quantificateurs sont au début : $Q_1 x_1 Q_2 x_2 \dots Q_n x_n \varphi$ où les Q_i sont $\exists$ ou $\forall$ et φ ne contient pas de quantificateur.
  On utilise l'indépendance et le renommage :
  - Si x n'apparaît pas dans ψ, on a $(\forall x \varphi) \vee \psi \equiv \forall x \varphi \vee \psi$.

Formes normales

- Une formule est en forme normale négative si $\neg$ n'apparaît que devant des formules atomiques.
  - Lois de De Morgan + changement de quantificateur
- Un littéral est une formule atomique éventuellement précédée de $\neg$.
- On définit comme pour la LP formes conjonctive et disjonctive.
- L'équivalence permet aussi de déplacer les quantificateurs.
  Une formule est sous forme normale prénexe si tous les quantificateurs sont au début : $Q_1 x_1 Q_2 x_2 \dots Q_n x_n \varphi$ où les Q_i sont $\exists$ ou $\forall$ et φ ne contient pas de quantificateur.
  On utilise l'indépendance et le renommage :
  - Si x n'apparaît pas dans ψ, on a $(\forall x \varphi) \vee \psi \equiv \forall x \varphi \vee \psi$.
    (De même avec $\forall$ et $\wedge$, avec $\exists$ et $\vee$, avec $\exists$ et $\wedge$)

Formes normales

- Une formule est en forme normale négative si $\neg$ n'apparaît que devant des formules atomiques.
  - Lois de De Morgan + changement de quantificateur
- Un littéral est une formule atomique éventuellement précédée de $\neg$.
- On définit comme pour la LP formes conjonctive et disjonctive.
- L'équivalence permet aussi de déplacer les quantificateurs.
  Une formule est sous forme normale prénexe si tous les quantificateurs sont au début : $Q_1 x_1 Q_2 x_2 \dots Q_n x_n \varphi$ où les Q_i sont $\exists$ ou $\forall$ et φ ne contient pas de quantificateur.
  On utilise l'indépendance et le renommage :
  - Si x n'apparaît pas dans ψ, on a $(\forall x \varphi) \vee \psi \equiv \forall x \varphi \vee \psi$.
    (De même avec $\forall$ et $\wedge$, avec $\exists$ et $\vee$, avec $\exists$ et $\wedge$)
  - Si x apparaît dans ψ, il suffit de choisir une variable y n'apparaissant ni dans ψ ni dans φ et de renommer x en y dans $\forall x \varphi$.

Formes normales

- Une formule est en forme normale négative si $\neg$ n'apparaît que devant des formules atomiques.
  - Lois de De Morgan + changement de quantificateur
- Un littéral est une formule atomique éventuellement précédée de $\neg$.
- On définit comme pour la LP formes conjonctive et disjonctive.
- L'équivalence permet aussi de déplacer les quantificateurs.
  Une formule est sous forme normale prénexe si tous les quantificateurs sont au début : $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \varphi$ où les Q_i sont $\exists$ ou $\forall$ et φ ne contient pas de quantificateur.
  On utilise l'indépendance et le renommage :
  - Si x n'apparaît pas dans ψ, on a $(\forall x \varphi) \vee \psi \equiv \forall x \varphi \vee \psi$.
    (De même avec $\forall$ et $\wedge$, avec $\exists$ et $\vee$, avec $\exists$ et $\wedge$)
  - Si x apparaît dans ψ, il suffit de choisir une variable y n'apparaissant ni dans ψ ni dans φ et de renommer x en y dans $\forall x \varphi$.
- Une formule universelle est une formule sous forme prénexe dont toutes les variables sont quantifiées universellement.
  Elle est donc de la forme $\forall * \varphi$ avec φ sans quantificateur.

Unification des termes du premier ordre

On cherche maintenant à détecter quand deux littéraux sont contradictoires, toutes les variables étant quantifiées universellement.

En d'autres termes, étant donnés L_1 et L_2, on cherche à savoir si l'ensemble $\{\forall * L_1, \forall * L_2\}$ est insatisfaisable. Pour qu'il le soit :

Unification des termes du premier ordre

On cherche maintenant à détecter quand deux littéraux sont contradictoires, toutes les variables étant quantifiées universellement.

En d'autres termes, étant donnés L_1 et L_2, on cherche à savoir si l'ensemble $\{\forall * L_1, \forall * L_2\}$ est insatisfaisable. Pour qu'il le soit :

- L_1 et L_2 doivent être de la forme $P(t_1, \ldots, t_n)$ et $\neg P(t'_1, \ldots, t'_n)$ pour un certain prédicat P.

Unification des termes du premier ordre

On cherche maintenant à détecter quand deux littéraux sont contradictoires, toutes les variables étant quantifiées universellement.

En d'autres termes, étant donnés L_1 et L_2, on cherche à savoir si l'ensemble $\{\forall * L_1, \forall * L_2\}$ est insatisfaisable. Pour qu'il le soit :

- L_1 et L_2 doivent être de la forme $P(t_1, \ldots, t_n)$ et $\neg P(t'_1, \ldots, t'_n)$ pour un certain prédicat P.
- Pour toute réalisation $\mathcal{M}$ du langage, il doit exister une affectation θ telle que $\mathrm{Val}_{\mathcal{M}}(t_i, \theta) = \mathrm{Val}_{\mathcal{M}}(t'_i, \theta)$ pour tout i. (Sinon, il serait possible de trouver un modèle de l'ensemble en modifiant l'interprétation de P.)

Unification des termes du premier ordre

On cherche maintenant à détecter quand deux littéraux sont contradictoires, toutes les variables étant quantifiées universellement.

En d'autres termes, étant donnés L_1 et L_2, on cherche à savoir si l'ensemble $\{\forall * L_1, \forall * L_2\}$ est insatisfaisable. Pour qu'il le soit :

- L_1 et L_2 doivent être de la forme $P(t_1, \ldots, t_n)$ et $\neg P(t'_1, \ldots, t'_n)$ pour un certain prédicat P.
- Pour toute réalisation $\mathcal{M}$ du langage, il doit exister une affectation θ telle que $\mathrm{Val}_{\mathcal{M}}(t_i, \theta) = \mathrm{Val}_{\mathcal{M}}(t'_i, \theta)$ pour tout i. (Sinon, il serait possible de trouver un modèle de l'ensemble en modifiant l'interprétation de P.)

Pour résoudre le deuxième point de façon purement syntaxique (c'est-à-dire indépendamment de $\mathcal{M}$), on va chercher une substitution (non nécessairement fermée) des variables.

Unification des termes du premier ordre

Soient t_1 et t_2 deux termes, on dit qu'une substitution σ unifie ces deux termes si $t_1\sigma = t_2\sigma$.

Unification des termes du premier ordre

Soient t_1 et t_2 deux termes, on dit qu'une substitution σ unifie ces deux termes si $t_1\sigma = t_2\sigma$.

Exemple : si $t = g(x, f(y))$ et $t' = g(f(z), x)$, la substitution $\sigma : x \mapsto f(z), y \mapsto z$ unifie t et t'.

On peut voir σ comme une solution de l'équation $t \sim t'$ dans l'ensemble des termes.

Plus généralement, on peut considérer des systèmes d'équations : un problème d'unification est un ensemble d'équations $t_i \sim t'_i$. Une solution est une substitution qui unifie t_i à t'_i pour tout i.

Unification des termes du premier ordre

Soient t_1 et t_2 deux termes, on dit qu'une substitution σ unifie ces deux termes si $t_1\sigma = t_2\sigma$.

Exemple : si $t = g(x, f(y))$ et $t' = g(f(z), x)$, la substitution $\sigma : x \mapsto f(z), y \mapsto z$ unifie t et t'.

On peut voir σ comme une solution de l'équation $t \sim t'$ dans l'ensemble des termes.

Plus généralement, on peut considérer des systèmes d'équations : un problème d'unification est un ensemble d'équations $t_i \sim t'_i$. Une solution est une substitution qui unifie t_i à t'_i pour tout i.

Étant donné un problème d'unification, une solution σ est appelée l'unificateur le plus général (upg) du problème si pour toute autre solution ρ il existe une substitution π telle que $\rho = \sigma\pi$.

Unification des termes du premier ordre

Soient t_1 et t_2 deux termes, on dit qu'une substitution σ unifie ces deux termes si $t_1\sigma = t_2\sigma$.

Exemple : si $t = g(x, f(y))$ et $t' = g(f(z), x)$, la substitution $\sigma : x \mapsto f(z), y \mapsto z$ unifie t et t'.

On peut voir σ comme une solution de l'équation $t \sim t'$ dans l'ensemble des termes.

Plus généralement, on peut considérer des systèmes d'équations : un problème d'unification est un ensemble d'équations $t_i \sim t'_i$. Une solution est une substitution qui unifie t_i à t'_i pour tout i.

Étant donné un problème d'unification, une solution σ est appelée l'unificateur le plus général (upg) du problème si pour toute autre solution ρ il existe une substitution π telle que $\rho = \sigma\pi$.

Propriété : si un problème d'unification possède une solution, il possède un upg, unique au nom des variables près.

Algorithme d'unification

L'algorithme suivant permet de calculer l'upg d'un problème d'unification, s'il existe, et d'indiquer que le système est insoluble sinon. P est un système d'équations entre termes. On lui applique itérativement les transformations suivantes, le but étant d'aboutir uniquement à des équations de la forme $x \sim t$:

Effacement Éliminer les équations de la forme $t \sim t$.

Retournement Remplacer $t \sim x$ par $x \sim t$ (si t n'est pas une variable).

Décomposition Remplacer $f(t_1, \ldots, t_n) \sim f(t'_1, \ldots, t'_n)$ par la liste d'équations $t_1 \sim t'_1, \ldots t_n \sim t'_n$.

Conflit Si P contient une équation de la forme $f(\ldots) \sim g(\ldots)$ avec $f \neq g$, il n'y a pas de solution : terminer. (Vrai aussi pour les constantes.)

Remplacement Si P contient $x \sim t$ et x n'apparaît pas dans t, remplacer x par t dans toutes les autres équations.

Occurrence Si P contient $x \sim t$, x apparaît dans t et $x \neq t$, il n'y a pas de solution : terminer.

L'algorithme se termine quand plus aucune règle ne s'applique, ou quand on a déterminé qu'il n'y avait pas de solution.

Formes de Skolem

On considère des formules sous forme prénexe (tous les quantificateurs sont au début).
Pour simplifier leur traitement dans des algorithmes, on cherche à éliminer les quantificateurs existentiels.

Formes de Skolem

On considère des formules sous forme prénexe (tous les quantificateurs sont au début).
Pour simplifier leur traitement dans des algorithmes, on cherche à éliminer les quantificateurs existentiels.

Idée :

- si $\exists x\, \varphi$ est satisfaisable, la valeur de x qui rend φ vraie peut être représentée par une constante c qu'on ajoute au langage.

Formes de Skolem

On considère des formules sous forme prénexe (tous les quantificateurs sont au début).
Pour simplifier leur traitement dans des algorithmes, on cherche à éliminer les quantificateurs existentiels.

Idée :

- si $\exists x\, \varphi$ est satisfaisable, la valeur de x qui rend φ vraie peut être représentée par une constante c qu'on ajoute au langage.
- plus généralement, si $\forall x_1 \forall x_2 \ldots \forall x_n \exists x\, \varphi$ est satisfaisable, on peut associer à toutes valeurs de $x_1 \ldots x_n$ une valeur de x qui rend φ vraie.

Formes de Skolem

On considère des formules sous forme prénexe (tous les quantificateurs sont au début).
Pour simplifier leur traitement dans des algorithmes, on cherche à éliminer les quantificateurs existentiels.

Idée :

- si $\exists x\, \varphi$ est satisfaisable, la valeur de x qui rend φ vraie peut être représentée par une constante c qu'on ajoute au langage.
- plus généralement, si $\forall x_1 \forall x_2 \ldots \forall x_n \exists x\, \varphi$ est satisfaisable, on peut associer à toutes valeurs de $x_1 \ldots x_n$ une valeur de x qui rend φ vraie.
  Cette valeur peut être représentée par une fonction $f(x_1 \ldots x_n)$.

Formes de Skolem

Soit $\varphi = \forall x_1 \dots \forall x_{i_1} \exists y_1 \forall x_{i_1+1} \dots \forall x_{i_2} \exists y_2 \ \cdots \ \forall x_{i_{n-1}+1} \dots \forall x_{i_n} \exists y_n \psi$ une formule close sous forme prénexe, dans un langage du premier ordre de signature Σ, telle que ψ ne contient pas de quantificateur existentiel.

On suppose toutes les variables différentes (sinon on renomme).

Soient $f_1^{i_1} \dots f_n^{i_n}$ des symboles de fonction n'appartenant pas à Σ.

Soit σ la substitution $\{y_k \mapsto f_k(x_1, \dots, x_{i_k}) \mid 1 \leqslant k \leqslant n\}$.

La forme de Skolem $\mathrm{sk}(\varphi)$ de φ est une formule sur le langage de signature $\Sigma' = \Sigma \cup \{f_1^{i_1} \dots f_k^{i_k}\}$, définie par $\mathrm{sk}(\varphi) = \forall x_1 \dots \forall x_{i_n} \psi\sigma$.

On appelle le passage de φ à $\mathrm{sk}(\varphi)$ « skolémisation ».

Formes de Skolem

Soit $\varphi = \forall x_1 \ldots \forall x_{i_1} \exists y_1 \forall x_{i_1+1} \ldots \forall x_{i_2} \exists y_2 \ \cdots \ \forall x_{i_{n-1}+1} \ldots \forall x_{i_n} \exists y_n \psi$ une formule close sous forme prénexe, dans un langage du premier ordre de signature Σ, telle que ψ ne contient pas de quantificateur existentiel.

On suppose toutes les variables différentes (sinon on renomme).

Soient $f_1^{i_1} \ldots f_n^{i_n}$ des symboles de fonction n'appartenant pas à Σ.

Soit σ la substitution $\{y_k \mapsto f_k(x_1, \ldots, x_{i_k}) \mid 1 \leqslant k \leqslant n\}$.

La forme de Skolem $\mathrm{sk}(\varphi)$ de φ est une formule sur le langage de signature $\Sigma' = \Sigma \cup \{f_1^{i_1} \ldots f_k^{i_k}\}$, définie par $\mathrm{sk}(\varphi) = \forall x_1 \ldots \forall x_{i_n} \psi\sigma$.

On appelle le passage de φ à $\mathrm{sk}(\varphi)$ « skolémisation ».

Théorème : Une formule est satisfaisable si et seulement si sa forme de Skolem l'est.

Formes de Skolem

Soit $\varphi = \forall x_1 \ldots \forall x_{i_1} \exists y_1 \forall x_{i_1+1} \ldots \forall x_{i_2} \exists y_2 \ \cdots \ \forall x_{i_{n-1}+1} \ldots \forall x_{i_n} \exists y_n \psi$ une formule close sous forme prénexe, dans un langage du premier ordre de signature Σ, telle que ψ ne contient pas de quantificateur existentiel.

On suppose toutes les variables différentes (sinon on renomme).

Soient $f_1^{i_1} \ldots f_n^{i_n}$ des symboles de fonction n'appartenant pas à Σ.

Soit σ la substitution $\{y_k \mapsto f_k(x_1, \ldots, x_{i_k}) \mid 1 \leqslant k \leqslant n\}$.

La forme de Skolem $\mathrm{sk}(\varphi)$ de φ est une formule sur le langage de signature $\Sigma' = \Sigma \cup \{f_1^{i_1} \ldots f_k^{i_k}\}$, définie par $\mathrm{sk}(\varphi) = \forall x_1 \ldots \forall x_{i_n} \psi\sigma$.

On appelle le passage de φ à $\mathrm{sk}(\varphi)$ « skolémisation ».

Théorème : Une formule est satisfaisable si et seulement si sa forme de Skolem l'est.

Attention : les deux formules ne sont pas équivalentes
(mais on a la relation $\mathrm{sk}(\varphi) \models \varphi$).

Formes de Skolem

Soit $\varphi = \forall x_1 \ldots \forall x_{i_1} \exists y_1 \forall x_{i_1+1} \ldots \forall x_{i_2} \exists y_2 \ \cdots \ \forall x_{i_{n-1}+1} \ldots \forall x_{i_n} \exists y_n \psi$ une formule close sous forme prénexe, dans un langage du premier ordre de signature Σ, telle que ψ ne contient pas de quantificateur existentiel.

On suppose toutes les variables différentes (sinon on renomme).

Soient $f_1^{i_1} \ldots f_n^{i_n}$ des symboles de fonction n'appartenant pas à Σ.

Soit σ la substitution $\{y_k \mapsto f_k(x_1, \ldots, x_{i_k}) \mid 1 \leqslant k \leqslant n\}$.

La forme de Skolem $\mathrm{sk}(\varphi)$ de φ est une formule sur le langage de signature $\Sigma' = \Sigma \cup \{f_1^{i_1} \ldots f_k^{i_k}\}$, définie par $\mathrm{sk}(\varphi) = \forall x_1 \ldots \forall x_{i_n} \psi\sigma$.

On appelle le passage de φ à $\mathrm{sk}(\varphi)$ « skolémisation ».

Théorème : Une formule est satisfaisable si et seulement si sa forme de Skolem l'est.

Attention : les deux formules ne sont pas équivalentes
(mais on a la relation $\mathrm{sk}(\varphi) \models \varphi$).

Une formule peut en particulier être *valide* sans que sa forme de Skolem le soit. Exemple :
$\forall x \exists y\, P(x,x) \Rightarrow P(x,y)$ est valide mais pas $\forall x\, P(x,x) \Rightarrow P(x, f(x))$.

Formes de Skolem

Preuve du théorème dans le cas $n = 1$ et $i_1 = 1$:
Soit $\varphi = \forall x \exists y \, \psi$. La forme de Skolem de φ est
$\varphi' = \forall x \, \psi[y \mapsto f(x)]$.

Formes de Skolem

Preuve du théorème dans le cas $n = 1$ et $i_1 = 1$:
Soit $\varphi = \forall x \exists y \, \psi$. La forme de Skolem de φ est
$\varphi' = \forall x \, \psi[y \mapsto f(x)]$.
Si φ est satisfaisable elle possède un modèle. Soit $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ un tel modèle. Puisque $\mathcal{M} \models \varphi$, pour tout $d \in \mathcal{D}$ on peut trouver un $d' \in \mathcal{D}$ tel que $\mathcal{E}(\psi, \mathcal{M}, \{x \mapsto d, y \mapsto d'\}) = \mathrm{V}$.

Formes de Skolem

Preuve du théorème dans le cas $n = 1$ et $i_1 = 1$:
Soit $\varphi = \forall x \exists y \, \psi$. La forme de Skolem de φ est
$\varphi' = \forall x \, \psi[y \mapsto f(x)]$.
Si φ est satisfaisable elle possède un modèle. Soit $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ un tel modèle. Puisque $\mathcal{M} \models \varphi$, pour tout $d \in \mathcal{D}$ on peut trouver un $d' \in \mathcal{D}$ tel que $\mathcal{E}(\psi, \mathcal{M}, \{x \mapsto d, y \mapsto d'\}) = \mathrm{V}$.
On peut donc définir une fonction $\gamma : \mathcal{D} \to \mathcal{D}$ qui à chaque d associe le d' correspondant.

Formes de Skolem

Preuve du théorème dans le cas $n = 1$ et $i_1 = 1$:
Soit $\varphi = \forall x \exists y \, \psi$. La forme de Skolem de φ est
$\varphi' = \forall x \, \psi[y \mapsto f(x)]$.
Si φ est satisfaisable elle possède un modèle. Soit $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ un tel modèle. Puisque $\mathcal{M} \models \varphi$, pour tout $d \in \mathcal{D}$ on peut trouver un $d' \in \mathcal{D}$ tel que $\mathcal{E}(\psi, \mathcal{M}, \{x \mapsto d, y \mapsto d'\}) = \mathrm{V}$.
On peut donc définir une fonction $\gamma : \mathcal{D} \to \mathcal{D}$ qui à chaque d associe le d' correspondant.
Soit I' l'interprétation de Σ' obtenue en ajoutant à I l'interprétation $\mathrm{I}'(f) = \gamma$. Soit $\mathcal{M}' = (\mathcal{D}, \mathrm{I}')$. On va montrer que $\mathcal{M}' \models \varphi'$.

Formes de Skolem

Preuve du théorème dans le cas $n = 1$ et $i_1 = 1$:
Soit $\varphi = \forall x \exists y \, \psi$. La forme de Skolem de φ est
$\varphi' = \forall x \, \psi[y \mapsto f(x)]$.
Si φ est satisfaisable elle possède un modèle. Soit $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ un tel modèle. Puisque $\mathcal{M} \models \varphi$, pour tout $d \in \mathcal{D}$ on peut trouver un $d' \in \mathcal{D}$ tel que $\mathcal{E}(\psi, \mathcal{M}, \{x \mapsto d, y \mapsto d'\}) = \mathrm{V}$.
On peut donc définir une fonction $\gamma : \mathcal{D} \to \mathcal{D}$ qui à chaque d associe le d' correspondant.
Soit I' l'interprétation de Σ' obtenue en ajoutant à I l'interprétation $\mathrm{I}'(f) = \gamma$. Soit $\mathcal{M}' = (\mathcal{D}, \mathrm{I}')$. On va montrer que $\mathcal{M}' \models \varphi'$.
Soit une affectation $\theta : x \mapsto d$.

Formes de Skolem

Preuve du théorème dans le cas $n = 1$ et $i_1 = 1$:
Soit $\varphi = \forall x \exists y \, \psi$. La forme de Skolem de φ est $\varphi' = \forall x \, \psi[y \mapsto f(x)]$.
Si φ est satisfaisable elle possède un modèle. Soit $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ un tel modèle. Puisque $\mathcal{M} \models \varphi$, pour tout $d \in \mathcal{D}$ on peut trouver un $d' \in \mathcal{D}$ tel que $\mathcal{E}(\psi, \mathcal{M}, \{x \mapsto d, y \mapsto d'\}) = \mathrm{V}$.
On peut donc définir une fonction $\gamma : \mathcal{D} \to \mathcal{D}$ qui à chaque d associe le d' correspondant.
Soit I' l'interprétation de Σ' obtenue en ajoutant à I l'interprétation $\mathrm{I}'(f) = \gamma$. Soit $\mathcal{M}' = (\mathcal{D}, \mathrm{I}')$. On va montrer que $\mathcal{M}' \models \varphi'$.
Soit une affectation $\theta : x \mapsto d$. Quand on calcule $\mathcal{E}(\psi[y \mapsto f(x)], \mathcal{M}', \theta)$, x est remplacé par d et $f(x)$ est remplacé par $\gamma(d)$, or on sait que ψ est vraie avec $x \mapsto d$ et $y \mapsto \gamma(d)$. Donc $\psi[y \mapsto f(x)]$ est vraie pour l'affectation θ, et comme c'est une affectation quelconque, $\forall x \psi[y \mapsto f(x)]$ est vraie dans l'interprétation $\mathcal{M}'$: on a bien $\mathcal{M}' \models \varphi'$.

Formes de Skolem

Preuve du théorème dans le cas $n = 1$ et $i_1 = 1$:
Soit $\varphi = \forall x \exists y \, \psi$. La forme de Skolem de φ est $\varphi' = \forall x \, \psi[y \mapsto f(x)]$.
Si φ est satisfaisable elle possède un modèle. Soit $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ un tel modèle. Puisque $\mathcal{M} \models \varphi$, pour tout $d \in \mathcal{D}$ on peut trouver un $d' \in \mathcal{D}$ tel que $\mathcal{E}(\psi, \mathcal{M}, \{x \mapsto d, y \mapsto d'\}) = \mathrm{V}$.
On peut donc définir une fonction $\gamma : \mathcal{D} \to \mathcal{D}$ qui à chaque d associe le d' correspondant.
Soit I' l'interprétation de Σ' obtenue en ajoutant à I l'interprétation $\mathrm{I}'(f) = \gamma$. Soit $\mathcal{M}' = (\mathcal{D}, \mathrm{I}')$. On va montrer que $\mathcal{M}' \models \varphi'$.
Soit une affectation $\theta : x \mapsto d$. Quand on calcule $\mathcal{E}(\psi[y \mapsto f(x)], \mathcal{M}', \theta)$, x est remplacé par d et $f(x)$ est remplacé par $\gamma(d)$, or on sait que ψ est vraie avec $x \mapsto d$ et $y \mapsto \gamma(d)$. Donc $\psi[y \mapsto f(x)]$ est vraie pour l'affectation θ, et comme c'est une affectation quelconque, $\forall x \psi[y \mapsto f(x)]$ est vraie dans l'interprétation $\mathcal{M}'$: on a bien $\mathcal{M}' \models \varphi'$.
Réciproquement, soit $\mathcal{M} = (\mathcal{D}, \mathrm{I})$ un modèle quelconque de φ', alors $\mathcal{M}$ est aussi un modèle de φ : pour toute affectation $x \mapsto d$, il

Skolémisation d'un ensemble

Soit E un ensemble de formules sur un langage de signature Σ.
On peut skolémiser les formules de E une par une :

- on choisit un ordre pour les formules : $E = \{\varphi_0, \varphi_1, \varphi_2, \ldots\}$
- on skolémise φ_0. Cela donne une formule ψ_0 sur une signature $\Sigma_0 \supseteq \Sigma$.
  - Soit $E_0 = \{\psi_0, \varphi_1, \varphi_2, \ldots\}$. On a $E_0 \models E$ car $\psi_0 \models \varphi_0$ et les autres formules sont identiques.
  - D'autre part, supposons $\mathcal{M} \models E$, on a donc $\mathcal{M} \models \varphi_0$, donc il existe une extension $\mathcal{M}_0$ de $\mathcal{M}$ telle que $\mathcal{M}_0 \models \psi_0$.
  - Comme $\mathcal{M}_0$ est identique à $\mathcal{M}$ sauf pour les nouveaux symboles ajoutés dans ψ_0, et comme les autres φ_i ne contiennent pas ces symboles, on a aussi $\mathcal{M}_0 \models \varphi_i$ pour $i \neq 0$. Donc au total $\mathcal{M}_0 \models E_0$.
- On skol. φ_1 avec des symboles hors de Σ_0. On obtient ψ_1, sur $\Sigma_1 \supseteq \Sigma_0$.
  - On définit $E_1 = \{\psi_0, \psi_1, \varphi_2, \ldots\}$. On a $E_1 \models E_0$ donc $E_1 \models E$.
  - Supposons $\mathcal{M}_0 \models E_0$, on a donc $\mathcal{M}_0 \models \varphi_1$ et on a une extension $\mathcal{M}_1$ de $\mathcal{M}_0$ telle que $\mathcal{M}_1 \models \psi_1$, et on a donc $\mathcal{M}_1 \models E_1$.
  - Donc si on a un modèle $\mathcal{M} \models E$, il existe un modèle $\mathcal{M}_1 \models E_1$ (construit en deux étapes).
- Et ainsi de suite. À chaque étape on garde la propriété : $E_n \models E$ et si $\mathcal{M} \models E$ alors on peut construire $\mathcal{M}_n$ tel que $\mathcal{M}_n \models E_n$.

Skolémisation d'un ensemble

Au bout du compte on obtient un ensemble $\mathrm{sk}(E)$ ne contenant que des formules universelles et qui est satisfaisable si et seulement si E est satisfaisable.

Remarque : c'est possible même pour E infini (on peut par exemple numéroter les symboles de skolémisation pour s'assurer qu'ils sont tous différents). (N.B. : cela fonctionne car l'ensemble des formules est dénombrable.)

Forme clausale

En logique du premier ordre, on appelle clause une disjonction de littéraux précédée d'une quantification universelle de toutes les variables intervenant dans la clause.
Une formule sous forme clausale est une conjonction de clauses.

Forme clausale

En logique du premier ordre, on appelle clause une disjonction de littéraux précédée d'une quantification universelle de toutes les variables intervenant dans la clause.
Une formule sous forme clausale est une conjonction de clauses.
Mise d'une formule sous forme clausale (en conservant la satisfaisabilité mais pas la validité) :

1. On traduit les $\Rightarrow$ et $\Leftrightarrow$

Forme clausale

En logique du premier ordre, on appelle clause une disjonction de littéraux précédée d'une quantification universelle de toutes les variables intervenant dans la clause.
Une formule sous forme clausale est une conjonction de clauses.
Mise d'une formule sous forme clausale (en conservant la satisfaisabilité mais pas la validité) :

1. On traduit les $\Rightarrow$ et $\Leftrightarrow$
2. On la met sous forme normale négative

Forme clausale

En logique du premier ordre, on appelle clause une disjonction de littéraux précédée d'une quantification universelle de toutes les variables intervenant dans la clause.
Une formule sous forme clausale est une conjonction de clauses.
Mise d'une formule sous forme clausale (en conservant la satisfaisabilité mais pas la validité) :

1. On traduit les $\Rightarrow$ et $\Leftrightarrow$
2. On la met sous forme normale négative
3. On la met sous forme prénexe

Forme clausale

En logique du premier ordre, on appelle clause une disjonction de littéraux précédée d'une quantification universelle de toutes les variables intervenant dans la clause.
Une formule sous forme clausale est une conjonction de clauses.
Mise d'une formule sous forme clausale (en conservant la satisfaisabilité mais pas la validité) :

1. On traduit les $\Rightarrow$ et $\Leftrightarrow$
2. On la met sous forme normale négative
3. On la met sous forme prénexe
4. On la skolémise

Forme clausale

En logique du premier ordre, on appelle clause une disjonction de littéraux précédée d'une quantification universelle de toutes les variables intervenant dans la clause.
Une formule sous forme clausale est une conjonction de clauses.
Mise d'une formule sous forme clausale (en conservant la satisfaisabilité mais pas la validité) :

1. On traduit les $\Rightarrow$ et $\Leftrightarrow$
2. On la met sous forme normale négative
3. On la met sous forme prénexe
4. On la skolémise
5. On met sous forme normale conjonctive ce qui est après les quantificateurs

Forme clausale

En logique du premier ordre, on appelle clause une disjonction de littéraux précédée d'une quantification universelle de toutes les variables intervenant dans la clause.
Une formule sous forme clausale est une conjonction de clauses.
Mise d'une formule sous forme clausale (en conservant la satisfaisabilité mais pas la validité) :

1. On traduit les $\Rightarrow$ et $\Leftrightarrow$
2. On la met sous forme normale négative
3. On la met sous forme prénexe
4. On la skolémise
5. On met sous forme normale conjonctive ce qui est après les quantificateurs
6. On utilise l'équivalence $\forall x\,(C \wedge C') \equiv (\forall x\,C) \wedge (\forall x\,C')$.

Forme clausale

En logique du premier ordre, on appelle clause une disjonction de littéraux précédée d'une quantification universelle de toutes les variables intervenant dans la clause.
Une formule sous forme clausale est une conjonction de clauses.
Mise d'une formule sous forme clausale (en conservant la satisfaisabilité mais pas la validité) :

1. On traduit les $\Rightarrow$ et $\Leftrightarrow$
2. On la met sous forme normale négative
3. On la met sous forme prénexe
4. On la skolémise
5. On met sous forme normale conjonctive ce qui est après les quantificateurs
6. On utilise l'équivalence $\forall x\,(C \wedge C') \equiv (\forall x\, C) \wedge (\forall x\, C')$.

On peut ensuite renommer les variables de façon que chaque clause en utilise des différentes.

Forme clausale

En logique du premier ordre, on appelle clause une disjonction de littéraux précédée d'une quantification universelle de toutes les variables intervenant dans la clause.
Une formule sous forme clausale est une conjonction de clauses.
Mise d'une formule sous forme clausale (en conservant la satisfaisabilité mais pas la validité) :

1. On traduit les $\Rightarrow$ et $\Leftrightarrow$
2. On la met sous forme normale négative
3. On la met sous forme prénexe
4. On la skolémise
5. On met sous forme normale conjonctive ce qui est après les quantificateurs
6. On utilise l'équivalence $\forall x\,(C \wedge C') \equiv (\forall x\, C) \wedge (\forall x\, C')$.

On peut ensuite renommer les variables de façon que chaque clause en utilise des différentes.
On réduit ainsi le problème de la satisfaisabilité d'une formule quelconque à celui de la satisfaisabilité d'un ensemble de clauses.

Forme clausale

En logique du premier ordre, on appelle clause une disjonction de littéraux précédée d'une quantification universelle de toutes les variables intervenant dans la clause.
Une formule sous forme clausale est une conjonction de clauses.
Mise d'une formule sous forme clausale (en conservant la satisfaisabilité mais pas la validité) :

1. On traduit les $\Rightarrow$ et $\Leftrightarrow$
2. On la met sous forme normale négative
3. On la met sous forme prénexe
4. On la skolémise
5. On met sous forme normale conjonctive ce qui est après les quantificateurs
6. On utilise l'équivalence $\forall x\,(C \wedge C') \equiv (\forall x\,C) \wedge (\forall x\,C')$.

On peut ensuite renommer les variables de façon que chaque clause en utilise des différentes.
On réduit ainsi le problème de la satisfaisabilité d'une formule quelconque à celui de la satisfaisabilité d'un ensemble de clauses.
(Mais le problème reste indécidable au sens algorithmique.)

La règle de résolution

On raisonne maintenant sur des ensembles de clauses.

On suppose que chaque clause a des variables différentes et on laisse implicites les quantificateurs universels.

La règle de résolution

On raisonne maintenant sur des ensembles de clauses.

On suppose que chaque clause a des variables différentes et on laisse implicites les quantificateurs universels.

Pour raisonner sur les clauses, on avait en logique propositionnelle la règle de résolution : $\dfrac{P \vee C \qquad \neg P \vee C'}{C \vee C'}$

formalisant le raisonnement :

« soit P est faux et alors C doit être vrai, soit P est vrai et alors C' doit être vrai. Dans tous les cas, $C \vee C'$ est vrai. »

La règle de résolution

On raisonne maintenant sur des ensembles de clauses.
On suppose que chaque clause a des variables différentes et on laisse implicites les quantificateurs universels.
Pour raisonner sur les clauses, on avait en logique propositionnelle la règle de résolution : $\dfrac{P \vee C \qquad \neg P \vee C'}{C \vee C'}$
formalisant le raisonnement :
« soit P est faux et alors C doit être vrai, soit P est vrai et alors C' doit être vrai. Dans tous les cas, $C \vee C'$ est vrai. »
En logique du premier ordre, on va utiliser le fait que $\forall x_1 \ldots \forall x_n \, \varphi$ implique $\forall x_1 \ldots \forall x_n \, \varphi\sigma$ pour toute substitution σ.

La règle de résolution

On raisonne maintenant sur des ensembles de clauses.
On suppose que chaque clause a des variables différentes et on laisse implicites les quantificateurs universels.
Pour raisonner sur les clauses, on avait en logique propositionnelle la règle de résolution : $\dfrac{P \vee C \qquad \neg P \vee C'}{C \vee C'}$
formalisant le raisonnement :
« soit P est faux et alors C doit être vrai, soit P est vrai et alors C' doit être vrai. Dans tous les cas, $C \vee C'$ est vrai. »
En logique du premier ordre, on va utiliser le fait que $\forall x_1 \ldots \forall x_n \, \varphi$ implique $\forall x_1 \ldots \forall x_n \, \varphi\sigma$ pour toute substitution σ.
Cela signifie que d'une clause C on peut déduire $C\sigma$ pour toute σ.

La règle de résolution

On raisonne maintenant sur des ensembles de clauses.
On suppose que chaque clause a des variables différentes et on laisse implicites les quantificateurs universels.
Pour raisonner sur les clauses, on avait en logique propositionnelle la règle de résolution : $\dfrac{P \vee C \qquad \neg P \vee C'}{C \vee C'}$
formalisant le raisonnement :
« soit P est faux et alors C doit être vrai, soit P est vrai et alors C' doit être vrai. Dans tous les cas, $C \vee C'$ est vrai. »
En logique du premier ordre, on va utiliser le fait que $\forall x_1 \ldots \forall x_n \, \varphi$ implique $\forall x_1 \ldots \forall x_n \, \varphi\sigma$ pour toute substitution σ.
Cela signifie que d'une clause C on peut déduire $C\sigma$ pour toute σ.
Soient les clauses $P(t_1, \ldots, t_n) \vee C$ et $\neg P(t'_1, \ldots, t'_n) \vee C'$. Soit σ l'upg du système $\{t_1 \sim t'_1; \cdots; t_n \sim t'_n\}$ (s'il existe).

La règle de résolution

On raisonne maintenant sur des ensembles de clauses.
On suppose que chaque clause a des variables différentes et on laisse implicites les quantificateurs universels.
Pour raisonner sur les clauses, on avait en logique propositionnelle la règle de résolution : $\dfrac{P \vee C \qquad \neg P \vee C'}{C \vee C'}$
formalisant le raisonnement :
« soit P est faux et alors C doit être vrai, soit P est vrai et alors C' doit être vrai. Dans tous les cas, $C \vee C'$ est vrai. »
En logique du premier ordre, on va utiliser le fait que $\forall x_1 \ldots \forall x_n \, \varphi$ implique $\forall x_1 \ldots \forall x_n \, \varphi\sigma$ pour toute substitution σ.
Cela signifie que d'une clause C on peut déduire $C\sigma$ pour toute σ.
Soient les clauses $P(t_1, \ldots, t_n) \vee C$ et $\neg P(t'_1, \ldots, t'_n) \vee C'$. Soit σ l'upg du système $\{t_1 \sim t'_1; \cdots; t_n \sim t'_n\}$ (s'il existe).
On peut déduire : $P(t_1\sigma, \ldots, t_n\sigma) \vee C\sigma$ et $\neg P(t'_1\sigma, \ldots, t'_n\sigma) \vee C'\sigma$.

La règle de résolution

On raisonne maintenant sur des ensembles de clauses.
On suppose que chaque clause a des variables différentes et on laisse implicites les quantificateurs universels.
Pour raisonner sur les clauses, on avait en logique propositionnelle la règle de résolution : $\dfrac{P \vee C \qquad \neg P \vee C'}{C \vee C'}$
formalisant le raisonnement :
« soit P est faux et alors C doit être vrai, soit P est vrai et alors C' doit être vrai. Dans tous les cas, $C \vee C'$ est vrai. »
En logique du premier ordre, on va utiliser le fait que $\forall x_1 \ldots \forall x_n \, \varphi$ implique $\forall x_1 \ldots \forall x_n \, \varphi\sigma$ pour toute substitution σ.
Cela signifie que d'une clause C on peut déduire $C\sigma$ pour toute σ.
Soient les clauses $P(t_1, \ldots, t_n) \vee C$ et $\neg P(t'_1, \ldots, t'_n) \vee C'$. Soit σ l'upg du système $\{t_1 \sim t'_1; \cdots; t_n \sim t'_n\}$ (s'il existe).
On peut déduire : $P(t_1\sigma, \ldots, t_n\sigma) \vee C\sigma$ et $\neg P(t'_1\sigma, \ldots, t'_n\sigma) \vee C'\sigma$.
Comme σ est l'upg du système, on a $t_1\sigma = t'_1\sigma$ etc. Les deux littéraux $P(...)$ sont donc maintenant complémentaires : on peut appliquer la résolution et conclure $C\sigma \vee C'\sigma$.

Résolution dans la logique du premier ordre

L'unification permet de construire les deux littéraux complémentaires nécessaires pour appliquer la résolution.

Résolution dans la logique du premier ordre

L'unification permet de construire les deux littéraux complémentaires nécessaires pour appliquer la résolution.

Cela donne la règle :

$$\frac{P(t_1, \ldots, t_n) \vee C \qquad \neg P(t'_1, \ldots, t'_,) \vee C'}{C\sigma \vee C'\sigma} \; \sigma = \mathsf{upg}(\{t_1 \sim t'_1; \cdots; t_n \sim t'_n\})$$

Résolution dans la logique du premier ordre

L'unification permet de construire les deux littéraux complémentaires nécessaires pour appliquer la résolution.
Cela donne la règle :

$$\frac{P(t_1, \ldots, t_n) \vee C \qquad \neg P(t'_1, \ldots, t'_,) \vee C'}{C\sigma \vee C'\sigma} \; \sigma = \mathsf{upg}(\{t_1 \sim t'_1; \cdots; t_n \sim t'_n\})$$

- On utilise l'associativité et la commutativité (les clauses sont vues comme des ensembles non ordonnés, comme en logique propositionnelle).

Résolution dans la logique du premier ordre

L'unification permet de construire les deux littéraux complémentaires nécessaires pour appliquer la résolution.
Cela donne la règle :

$$\frac{P(t_1, \ldots, t_n) \vee C \qquad \neg P(t'_1, \ldots, t'_,) \vee C'}{C\sigma \vee C'\sigma} \; \sigma = \mathsf{upg}(\{t_1 \sim t'_1; \cdots; t_n \sim t'_n\})$$

- On utilise l'associativité et la commutativité (les clauses sont vues comme des ensembles non ordonnés, comme en logique propositionnelle).
- Comme la clause conclusion doit être ajoutée à l'ensemble des clauses, on renomme ses variables pour éviter les conflits.

Résolution dans la logique du premier ordre

L'unification permet de construire les deux littéraux complémentaires nécessaires pour appliquer la résolution.
Cela donne la règle :

$$\frac{P(t_1, \ldots, t_n) \vee C \qquad \neg P(t'_1, \ldots, t'_,) \vee C'}{C\sigma \vee C'\sigma} \; \sigma = \mathsf{upg}(\{t_1 \sim t'_1; \cdots; t_n \sim t'_n\})$$

- On utilise l'associativité et la commutativité (les clauses sont vues comme des ensembles non ordonnés, comme en logique propositionnelle).
- Comme la clause conclusion doit être ajoutée à l'ensemble des clauses, on renomme ses variables pour éviter les conflits.
- Cas particulier : Les deux clauses dans les prémisses peuvent être deux copies de la même. Dans ce cas on doit renommer les variables d'une des copies.

Résolution dans la logique du premier ordre

L'unification permet de construire les deux littéraux complémentaires nécessaires pour appliquer la résolution.
Cela donne la règle :

$$\frac{P(t_1, \ldots, t_n) \vee C \qquad \neg P(t'_1, \ldots, t'_,) \vee C'}{C\sigma \vee C'\sigma} \; \sigma = \mathsf{upg}(\{t_1 \sim t'_1; \cdots; t_n \sim t'_n\})$$

- On utilise l'associativité et la commutativité (les clauses sont vues comme des ensembles non ordonnés, comme en logique propositionnelle).
- Comme la clause conclusion doit être ajoutée à l'ensemble des clauses, on renomme ses variables pour éviter les conflits.
- Cas particulier : Les deux clauses dans les prémisses peuvent être deux copies de la même. Dans ce cas on doit renommer les variables d'une des copies. Ex. : $P(f(x), y) \vee \neg P(x, g(y))$.

Résolution dans la logique du premier ordre

L'unification permet de construire les deux littéraux complémentaires nécessaires pour appliquer la résolution.
Cela donne la règle :

$$\frac{P(t_1, \ldots, t_n) \vee C \qquad \neg P(t'_1, \ldots, t'_,) \vee C'}{C\sigma \vee C'\sigma} \; \sigma = \mathsf{upg}(\{t_1 \sim t'_1; \cdots; t_n \sim t'_n\})$$

- On utilise l'associativité et la commutativité (les clauses sont vues comme des ensembles non ordonnés, comme en logique propositionnelle).
- Comme la clause conclusion doit être ajoutée à l'ensemble des clauses, on renomme ses variables pour éviter les conflits.
- Cas particulier : Les deux clauses dans les prémisses peuvent être deux copies de la même. Dans ce cas on doit renommer les variables d'une des copies. Ex. : $P(f(x), y) \vee \neg P(x, g(y))$.

$$\frac{P(f(x), y) \vee \neg P(x, g(y)) \qquad P(f(z), w) \vee \neg P(z, g(w))}{P(f(f(z)), y) \vee \neg P(z, g(g(y)))} \; \sigma = \begin{cases} x \mapsto f(z) \\ w \mapsto g(y) \end{cases}$$

Sans renommage on n'aurait pas pu unifier ($x \sim f(x)$ et $y \sim g(y)$).

La règle de factorisation

En logique propositionnelle on éliminait les littéraux redondants ($P \vee P$). Au premier ordre on peut avoir dans une même clause des littéraux non identiques mais unifiables, par ex. $P(x) \vee P(f(y))$.
Il est parfois nécessaire de fusionner ces littéraux.

La règle de factorisation

En logique propositionnelle on éliminait les littéraux redondants ($P \vee P$). Au premier ordre on peut avoir dans une même clause des littéraux non identiques mais unifiables, par ex. $P(x) \vee P(f(y))$.
Il est parfois nécessaire de fusionner ces littéraux.
On ajoute donc la règle :

$$\frac{P(t_1, \ldots, t_n) \vee P(t'_1, \ldots, t'_n) \vee C}{P(t_1\sigma, \ldots, t_n\sigma) \vee C\sigma} \; \sigma = \mathsf{upg}(\{t_1 \sim t'_1; \cdots; t_n \sim t'_n\})$$

La règle de factorisation

En logique propositionnelle on éliminait les littéraux redondants ($P \vee P$). Au premier ordre on peut avoir dans une même clause des littéraux non identiques mais unifiables, par ex. $P(x) \vee P(f(y))$.
Il est parfois nécessaire de fusionner ces littéraux.
On ajoute donc la règle :

$$\frac{P(t_1, \ldots, t_n) \vee P(t'_1, \ldots, t'_n) \vee C}{P(t_1\sigma, \ldots, t_n\sigma) \vee C\sigma} \; \sigma = \mathsf{upg}(\{t_1 \sim t'_1; \cdots; t_n \sim t'_n\})$$

Le système formé par ces règles (résolution et factorisation) est correct et complet pour la réfutation.

Semi-décidabilité de la logique du premier ordre

On peut écrire un algorithme qui applique indéfiniment la résolution à un ensemble de clauses, tant qu'il arrive à en générer de nouvelles. Le théorème de Herbrand permet d'être certain que, si l'ensemble de clauses est insatisfaisable, l'algorithme finira par le démontrer.

Semi-décidabilité de la logique du premier ordre

On peut écrire un algorithme qui applique indéfiniment la résolution à un ensemble de clauses, tant qu'il arrive à en générer de nouvelles. Le théorème de Herbrand permet d'être certain que, si l'ensemble de clauses est insatisfaisable, l'algorithme finira par le démontrer.

Mais si l'ensemble de clauses est *satisfaisable*, l'algorithme peut ne jamais terminer.

Semi-décidabilité de la logique du premier ordre

On peut écrire un algorithme qui applique indéfiniment la résolution à un ensemble de clauses, tant qu'il arrive à en générer de nouvelles. Le théorème de Herbrand permet d'être certain que, si l'ensemble de clauses est insatisfaisable, l'algorithme finira par le démontrer.

Mais si l'ensemble de clauses est *satisfaisable*, l'algorithme peut ne jamais terminer.

On dit que le problème de l'insatisfaisabilité (ou de la validité) dans la logique du premier ordre est semi-décidable :

- il n'existe pas d'algorithme qui donne toujours une réponse en temps fini (donc il n'est pas décidable), mais
- il existe un algorithme qui, dans le cas où la réponse est positive, la donne toujours en temps fini.