Système formels.

Un **système formel à la Hilbert** correspond à la donnée de $L = (A, P, F, R)$:
- Un alphabet A
- Un ensemble P de **règles de formation**, qu'on peut modéliser par une grammaire formelle, de symboles terminaux A, avec par exemple "," un symbole spécial utilisé pour séparer des inputs et exprimer des règles de formations sur plusieurs mots de A à la fois.
- Un **langage $\boldsymbol{F} \subseteq A^*$** de **formules bien formées** : c'est le langage engendré par P.
- Un ensemble R de **règles d'inférence**, qu'on peut modéliser par une grammaire formelle sur A, dont les règles doivent s'exprimer sur des formules bien formées de F.

Comme F est déterminé par P, on a pas vraiment besoin de spécifier F dans la définition.
Une règle d'inférence associe à n formules de F appelées **prémisses**, une formule de F appelée **conclusion**.
Un **axiome** correspond à une règle d'inférence sans prémisse, qui peut donc s'appliquer inconditionnellement.
Par abus de langage, une formule est un axiome ssi c'est la conclusion d'un axiome.
Une preuve d'une formule $\boldsymbol{f \in F}$ dans un système formel à la Hilbert $\boldsymbol{L = (A, P, F, R)}$ correspond à une suite finie de formules $(f_i)_{1 \leq i \leq n} \in F^n$ telle que à chaque étape i, la formule f_i résulte soit d'un axiome, soit d'une règle d'inférence appliquée à des formules précédentes, et telle que f est la dernière formule obtenue càd $f = f_n$.
Une formule $\boldsymbol{f \in F}$ est prouvable dans un système formel $\boldsymbol{L = (A, P, F, R)}$ ssi il en existe une preuve.
On note cela $\vdash_{\boldsymbol{L}} \boldsymbol{f}$
Une preuve d'une formule $\boldsymbol{f \in F}$ dans un système formel $\boldsymbol{L = (A, P, F, R)}$ à partir d'hypothèses $\boldsymbol{\Gamma \subseteq F}$ correspond à une suite finie de formules $(f_i)_{1 \leq i \leq n} \in F^n$ telle que à chaque étape i, la formule f_i résulte soit d'un axiome, soit d'une règle d'inférence appliquée à des formules précédentes ou à des hypothèses, et telle que f est la dernière formule obtenue càd $f = f_n$.
Une formule $\boldsymbol{f \in F}$ est prouvable dans un système formel $\boldsymbol{L = (A, P, F, R)}$ à partir d'hypothèses $\boldsymbol{\Gamma \subseteq F}$ ssi il en existe une preuve à partir de ces hypothèses.
On note cela $\boldsymbol{\Gamma \vdash_{L} f}$
Même sans fixer d'hypothèses, la notion de preuve, est toujours relative au système formel considéré, donc dépend toujours des axiomes et règles d'inférences choisies.
Une formule axiome, est une preuve d'elle-même, en une seule étape.
Si $f \in \Gamma$ alors $\Gamma \vdash_L f$
Pour $\Gamma \subseteq F$, et $\Sigma \subseteq F$, on dit que $\boldsymbol{\Gamma \vdash_{L} \Sigma}$ ssi $\forall f \in \Sigma \ \Gamma \vdash_{\mathrm{L}} f$
Un langage du premier ordre spécifié par sa signature L, correspond à un système formel, dont les règles de formation sont fournies par la définition inductive des formules, et les règles d'inférence sont fournies par les règles de la déduction naturelle.

Un système formel plus généralement, n'a pas de définition précise, mais correspond dans l'idée à un alphabet, des règles de formations, et des règles d'inférences, les règles pouvant être très particulières et complexes à formuler.
La déduction naturelle de Fitch est un système formel proche d'un système à la Hilbert.
La déduction naturelle de Gentzen est un système formel arborescent ou les preuves sont des arbres et pas des listes.
La déduction naturelle de Gentzen sans séquent a des règles complexes s'appliquant à des sous arbres de taille variable avec un concept d'hypothèses déchargeables.

1. Syntaxe de la logique

1.2. Langage du premier ordre

1.2.1. Signature

Une signature (d'un langage) du premier ordre notée L correspond à la donnée des ensembles distincts suivants.

- De la **partie non-logique** spécifique au langage:

Pour chaque $n \in \mathbb{N}^*$, d'un ensemble F_n de symboles de **fonctions d'arité** $\boldsymbol{n}$.

Pour chaque $n \in \mathbb{N}^*$, d'un ensemble R_n de symboles de **relations d'arité** $\boldsymbol{n}$.

D'un ensemble C de symboles **constantes**, qui peut se comprendre comme F_0.

D'un ensemble P de symboles **constantes propositionnelles**, qui peut se comprendre comme R_0.

D'un ensemble V de symboles **variables objets**.

- De la **partie logique**:

Des **quantificateurs logiques** en général au nombre de 2 $\{\forall, \exists\}$.

Un ensemble **d'opérateurs logiques binaires** pouvant inclure $\{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$ ($\{\wedge\}$ suffit).

Généralement la négation comme seul **opérateur logique unaire** $\{\neg\}$

Parfois aussi l'ensemble de **constantes logiques** $\{\bot, \top\}$

Tous ces ensembles et ces symboles sont supposés distincts, et aucun symbole n'est une sous-chaine d'un autre.

L'union de tous ces ensembles fournit **l'alphabet de la signature** $\boldsymbol{L}$: $\boldsymbol{\Sigma_L}$.

L'ensemble des variables V est généralement supposé fixé dans le contexte. Les opérateurs logiques et quantificateurs logiques sont les mêmes quelle que soit la signature choisie. Une signature se résume donc souvent à la donnée de $((F_n)_{n\in\mathbb{N}}, (R_n)_{n\in\mathbb{N}})$

On montre qu'une signature engendre un langage sur son alphabet de formules bien formées sur l'alphabet. Puisque fixer une signature L, fixe son langage de formules bien formées, on dit aussi par abus qu'une signature L est **un langage du premier ordre.**

1.2.2. Termes

L'ensemble des termes de $\boldsymbol{L}$ noté $\boldsymbol{\tau(L)}$ est défini inductivement par

$V \subseteq \tau$: Une variable est un terme.

$F_0 \subseteq \tau$: Une constante est un terme.

Si $f \in F_n, t_1, \dots, t_n \in \tau$ alors $f t_1 \dots t_n \in \tau$: Un opérateur n-aire appliqué à n termes, forme un terme.

Autrement dit $\tau = \bigcup_{k\in\mathbb{N}} \tau_k$ avec $\tau_{k+1} = \tau_k \cup \{f t_1 \dots t_n : n \in \mathbb{N}^*, f \in F_n, t_1, \dots, t_n \in \tau_k\}$ et $\tau_0 = V \cup F_0$.

Un **terme de** $\boldsymbol{L}$ est un élément de $\tau(L)$

Par soucis de lisibilité on écrit aussi $\boldsymbol{f(t_1, \dots, t_n)}$ le mot $f t_1 \dots t_n$

La hauteur d'un terme $\boldsymbol{t}$ est $\min\{k \in \mathbb{N} \mid t \in \tau_k\}$

Un **terme est clos** ssi il ne contient pas de variables, càd il n'est généré que par des constantes.

L'ensemble des termes (le type terme) peut se comprendre comme une grammaire ou un ADT (algebraic data type) : $\tau = V | F_0 | f(\tau, \dots, \tau)$

Un terme correspond à un arbre de nœud non feuille une fonction, et de nœud feuille une variable ou une constante.

Théorème de lecture unique pour les termes. Un terme est **non-ambigu** : Pour $t \in \tau$ alors soit $t \in F_0$, soit $t \in V$, soit $t = f t_1 \dots t_n$ avec d'<u>uniques</u> $n \in \mathbb{N}^*, t_1, \dots, t_n \in \tau, f \in F_n$, où

Unicité à prouver soigneusement pour le cas 3 en utilisant le lemme : Si t est un terme alors aucun préfixe propre de t n'est un terme.

Pour prouver le lemme, on considère la notion de **poids d'un mot** quelconque m de lettres $s_1 \dots s_m$, $p(s_1) + \cdots p(s_m)$ où $p(v) = -1$ si $v \in V$, $p(f) = n - 1$ si $f \in F_n$.

Le poids d'un terme est toujours -1 et le poids d'un préfixe propre d'un terme doit être ≥ 0.

Le théorème de lecture unique permet de définir la notion de **sous-terme** et d'**arbre de**

décomposition, de **hauteur d'un terme**.

La taille/longueur d'un terme $\boldsymbol{t}$ est le nombre $\boldsymbol{\tau(t)}$ de symboles de fonctions dans t.

Inductivement, si $t \in F_0 \cup V \ \ \tau(t) = 0,$ sinon $\tau\big(t = f(t_1, \dots, t_n)\big) = 1 + \sum_{i=1}^{n} \tau(t_i)$

Autrement dit la taille d'un terme est le nombre de nœuds non feuilles de son arbre.

La taille fournit un nouvel outil, autre que la hauteur, pour faire des récurrences.

Notation de dépendance. Les variables d'un terme sont celles ayant au moins une occurrence dedans, on peut en donner une définition inductive. Pour indiquer que les variables d'un terme sont exclusivement parmi $\{v_1, \dots, v_n\}$, on écrira parfois ce terme $\boldsymbol{t(v_1, \dots, v_n)}$

Substitution dans un terme. Soit $u \in \tau(L), v \in V, t \in \tau(L)$

On définit $\boldsymbol{u[v \coloneqq t]}$ inductivement par:

Si $u \in C \cup V \setminus \{v\}$ alors $\boldsymbol{u[v \coloneqq t]} = u$

Si $u = v$ alors $\boldsymbol{u[v \coloneqq t]} = t$

Si $u = f u_1 \dots u_m$ alors $\boldsymbol{u[v \coloneqq t]} = f u_1[v \coloneqq t] \dots u_m[v \coloneqq t]$

Substitution simultanée dans un terme. Soit $u \in \tau(L), v_1, \dots, v_n \in V, t_1, \dots, t_n \in \tau(L)$

Si $u \in C \cup V \setminus \{v_1, \dots, v_n\}$ alors $\boldsymbol{u[v_1 \coloneqq t_1, \dots, v_n \coloneqq t_n]} = u$

Si $u = v_i$ alors $\boldsymbol{u[v_1 \coloneqq t_1, \dots, v_n \coloneqq t_n]} = t_i$

Si $u = f u_1 \dots u_m$ alors $\boldsymbol{u[v_1 \coloneqq t_1, \dots, v_n \coloneqq t_n]} = f u_1[v_1 \coloneqq t_1, \dots, v_n \coloneqq t_n] \dots u_m[v_1 \coloneqq t_1, \dots, v_n \coloneqq t_n]$

1.2.3. Formules

Une **formule atomique de** $\boldsymbol{L}$, est

- Soit $f = \bot$
- Soit $f = \top$
- Soit $f \in R_0$
- Soit un mot $f = R t_1 \dots t_n$ avec $R \in R_n, t_1, \dots, t_n \in \tau(L)$.

On note $\boldsymbol{Atom(L)}$ l'ensemble des formules atomiques de L.

Le langage des formules sur $\boldsymbol{L}$ **= ensemble des formules sur** $\boldsymbol{L}$ noté $\boldsymbol{F(L)}$ est défini par

Si $f \in Atom$ alors $f \in F$

Si $f, g \in F$ alors $(f \wedge g) \in F, \ (f \vee g) \in F, (f \Rightarrow g) \in F$

Si $f \in F$ alors $\neg f \in F$

Si $f \in F$ et $x \in V$ alors $\forall x \ f \in F, \ \exists x \ f \in F$

Une formule de $\boldsymbol{L}$, est un élément de $F(L)$

$F(L)$ est un langage sur Σ_L. $F(L) \subseteq {\Sigma_L}^*$

Le type formule correspond à l'ADT : $F = Atom \mid F \wedge F \mid F \vee F \mid F \Rightarrow F \mid \neg F \mid \exists x \ F \mid \forall x \ F, (x \in V)$

Une formule correspond à un arbre de nœud non feuille un opérateur logique/un quantificateur logique, et de nœud feuille une formule atomique.

Une formule est **non-ambiguë** : pour $f \in F(L)$, on est exactement dans un des 4 cas de la définition inductive, de plus il y a unicité des sous-formules. Une formule dérive d'un unique arbre de formule.

Une formule n'utilise qu'un nombre fini de symboles non-logiques.

La signature d'une formule $\boldsymbol{f}$ **de** $\boldsymbol{L}$ noté $\boldsymbol{L(f)}$ est la signature du premier ordre dont les symboles non-logiques sont exactement ceux apparaissant dans f.

Le langage engendré par une formule $\boldsymbol{f}$ **de** $\boldsymbol{L}$ est celui engendré par sa signature $\boldsymbol{F(f) = F\big(L(f)\big)}$

L'ensemble des sous-formules $\boldsymbol{SF(f)}$ **d'une formule** $\boldsymbol{f}$ **de** $\boldsymbol{L}$ est défini inductivement par :

Si $f \in Atom, \ SF(f) = \{f\}$

Si $f = f_1 \circ f_2$ avec $\circ \in \{\wedge, \vee, \Rightarrow\}, SF(f_1 \circ f_2) = \{f_1 \circ f_2\} \cup SF(f_1) \cup SF(f_2)$

Si $f = \neg f_1, \ SF(\neg f_1) = \{\neg f_1\} \cup SF(f_1)$

Si $f = \square x\ f_1,$ avec $\square \in \{\forall, \exists\}$ et $x \in V,\ SF(\square x\ f_1) = \{\square x\ f_1\} \cup SF(f_1)$

Une sous-formule d'une formule $\boldsymbol{f}$ est un élément de $SF(f)$.

Une sous-formule d'une formule vue comme un arbre correspond à un sous-arbre.

La taille/longueur d'une formule $\boldsymbol{f} \in F(L)$ est le nombre $\boldsymbol{\tau(f)}$ de symboles d'opérateurs/quantificateurs logiques dans f. Inductivement,

Si $f \in Atom,\ \tau(f) = 0$

Si $f = f_1 \circ f_2$ avec $\circ \in \{\wedge, \vee, \Rightarrow\}$, $\tau(f_1 \circ f_2) = 1 + \tau(f_1) + \tau(f_2)$

Si $f = \neg f_1,\ \tau(\neg f_1) = 1 + \tau(f_1)$

Si $f = \square x\ f_1,$ avec $\square \in \{\forall, \exists\}$ et $x \in V,\ \tau(\square x\ f_1) = 1 + \tau(f_1)$

Autrement dit la taille d'une formule est le nombre de nœuds non feuilles de son arbre.

L'opérateur principal d'une formule $\boldsymbol{f} \in F(L)$ est la racine de son arbre.

1.2.4. Variables libres/liées, substitutions.

Une occurrence d'une variable $v \in V$ dans une formule $f \in F$ est une **occurrence libre** ssi elle n'est pas sous la portée d'un quantificateur, ssi dans la branche qui aboutit à la feuille de cette occurrence on ne trouve ni $\forall v$ ni $\exists v$. Dans le cas contraire **l'occurrence est liée**.

Une même variable peut avoir à la fois des occurrences libres et liées dans une même formule.

Une variable $\boldsymbol{v} \in \boldsymbol{V}$ est libre dans une formule $\boldsymbol{f} \in \boldsymbol{F}$ ssi v admet <u>au - une</u> occurrence libre dans f.

Dans le cas contraire **$\boldsymbol{v}$ est une variable liée** dans f. (ssi <u>toutes</u> ses occurrences sont liées dans f)

On peut définir l'ensemble des variables/variables libres/liées d'une formule f inductivement.

Une **formule est close** ssi elle n'a pas de variables libres.

Notation de dépendance. Pour indiquer que les variables <u>libres</u> d'une formule f sont exclusivement <u>parmi</u> $\{v_1, \dots, v_n\}$, on écrira parfois cette formule $\boldsymbol{f(v_1, \dots, v_n)}$

La clôture universelle d'une formule $\boldsymbol{f(v_1, \dots, v_n)}$ est la formule $\forall v_1 \dots \forall v_n\ f$

Une clôture universelle est toujours close.

Substitution d'une variable dans une formule.

Soit $t(x_1, \dots, x_n)$ terme, $f(v, x_1, \dots, x_n, u_1, \dots, u_p)$ une formule, (et $v \in V$ une variable).

Une substitution $\boldsymbol{f[v \coloneqq t]}$ est licite ssi <u>chaque</u> occurrence de la variable remplacée v est libre et n'est pas sous un quantificateur $\forall x_i$ ou $\exists x_i$ d'une des variables libres x_i du terme t introduit.

Pour généraliser la substitution au cas illicite, il faut prendre des précautions en renommant les $\forall x_i$ / $\exists x_i$

On définit $\boldsymbol{f[v \coloneqq t]}$ inductivement par:

Si $f = Rt_1 \dots t_n \in Atom$ alors $\boldsymbol{f[v \coloneqq t]} = Rt_1[v \coloneqq t] \dots t_n[v \coloneqq t]$

si $f \in R_0 \cup \{\bot, \top\} \in Atom$ alors $\boldsymbol{f[v \coloneqq t]} = f$

Si $f = f_1 \circ f_2$ avec $\circ \in \{\wedge, \vee, \Rightarrow\}$ alors $\boldsymbol{f[v \coloneqq t]} = f_1[v \coloneqq t] \circ f_2[v \coloneqq t]$

Si $f = \neg f_1$ alors $\boldsymbol{f[v \coloneqq t]} = \neg f_1[v \coloneqq t]$

Si $f = \square x\ f_1,$ avec $\square \in \{\forall, \exists\}$ et $x \in V \setminus \{v, x_1, \dots, x_n\},\ \boldsymbol{f[v \coloneqq t]} = \square x\ f_1[v \coloneqq t]$

Généralisation au cas illicite :

Si $f = \square x\ f_1,$ avec $\square \in \{\forall, \exists\}$ et $x = x_i$ alors on renomme d'abord x_i en une nouvelle variable y (sans occurrence dans f), afin de pouvoir substituer sans que x_i soit capturé par le quantificateur. $\boldsymbol{f[v \coloneqq t]} = \square y\ f_1[x_i \coloneqq y][v \coloneqq t]$.

Si $f = \square x\ f_1,$ avec $\square \in \{\forall, \exists\}$ et $x = v$ alors on ne substitue pas. $\boldsymbol{f[v \coloneqq t]} = \square x\ f_1$ (on peut aussi renommer x pour faire disparaitre v).

Remarque : On peut aussi définir la substitution simultanée $f\left[\frac{t_1}{v_1}, \dots, \frac{t_n}{v_n}\right]$, en général ce n'est pas la même chose qu'une substitution consécutive $f\left[\frac{t_1}{v_1}\right] \dots \left[\frac{t_n}{v_n}\right]$, qui dépend de l'ordre.

On pourrait aussi définir :
Substitution simultanée de variables dans une formule.
Substitution d'une ou plusieurs constantes propositionnelles dans une formule.

Déduction naturelle (arborescente (Gentzen) avec séquents).
Calcul propositionnel :

ax
$\Gamma, A \vdash A$

⇒intro
$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$$

⇒elim
$$\frac{\Gamma \vdash A \quad \Gamma \vdash A \Rightarrow B}{\Gamma \vdash B}$$

∧intro
$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$$

∧elim1
$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}$$

∧elim2
$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$$

∨intro1
$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}$$

∨intro2
$$\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$$

∨elim
$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}$$

⊥elim
$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}$$

¬intro
$$\frac{\Gamma, A \vdash \neg B \quad \Gamma, A \vdash B}{\Gamma \vdash \neg A}$$

¬elim
$$\frac{\Gamma \vdash \neg A \quad \Gamma \vdash A}{\Gamma \vdash \bot}$$

absurde
$$\frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}$$

¬¬elim
$$\frac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}$$

tiers exclus
$\Gamma \vdash A \vee \neg A$

La logique minimale est $\boldsymbol{NM}$ = {ax, ⇒intro, ⇒elim, ∧intro, ∧elim1, ∧elim2, ∨intro1, ∨intro2, ∨elim}
La logique intuitionniste est $\boldsymbol{NJ}$ $= NM \cup$ {⊥elim, ¬intro, ¬elim}
La logique classique est $\boldsymbol{NK}$ $= NJ \cup$ {absurde} $= NJ \cup$ {¬¬elim } $= NJ \cup$ {tiers exclus}
Calcul des prédicats :

∀intro
$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x\, A}$$
Préconditions :
x n'est libre dans aucune $f \in \Gamma$

∀elim
$$\frac{\Gamma \vdash \forall x\, A}{\Gamma \vdash A[x \coloneqq t]}$$
Préconditions :
$A[x \coloneqq t]$ substitution licite
(Ou pas de précondition et renommage automatique)

∃intro
$$\frac{\Gamma \vdash A[x \coloneqq t]}{\Gamma \vdash \exists x\, A}$$
Préconditions :

∃elim
$$\frac{\Gamma \vdash \exists x\, A \quad \Gamma, A \vdash C}{\Gamma \vdash C}$$
Préconditions :
x n'est libre dans aucune $f \in \Gamma \cup C$

Égalité

=intro
$\Gamma \vdash t = t$

=elim
$$\frac{\Gamma \vdash A[x \coloneqq t]}{}$$
Préconditions :
substitutions licites (ou pas de

$$\frac{\Gamma \vdash t = u}{\Gamma \vdash A[x \coloneqq u]}$$ précondition et renommage automatique)

Logique du premier ordre = logique classique ∪ {∀intro, ∀elim, ∃intro, ∃elim}

Logique du premier ordre avec égalité = Logique du premier ordre ∪ {=intro, =elim}

Déduction naturelle à la Fitch. Système plus proche de la rédaction humaine.
La rédaction d'une preuve est une liste multi-niveaux.
Chaque ligne est soit une entête soit une formule. Seuls les entêtes peuvent avoir des lignes indentées en dessous.
Le contexte de chaque formule est spécifié par les entêtes sous lesquelles elle se trouve.
Ce qui est affirmé dans un contexte peut être invalide dans un autre contexte. Il n'y a que deux types de contextes possibles, un contexte de supposition (Supposons que), et un contexte de quantification universelle (Soit une variable x).
Les capitales A, B, C représentent des formules de L.
On utilise … pour indiquer un nombre quelconque de lignes au même niveau d'indentation (dans le même contexte). Les règles énoncent parfois une formule de façon redondante pour que le système puisse rigoureusement s'appliquer. En pratique on évite de répéter ces informations redondantes dans une rédaction humaine, on marque entre crochets ces formules jugées trop encombrantes pour la rédaction.

restate Si on prouve quelque chose on peut le réaffirmer dans le même contexte.

A
…
$[A]$ En pratique un humain ne réécrit pas ce qu'il a déjà écrit d'où les [].

Calcul propositionnel :

⇒sub

Supposons A:
  $[A]$

⇒restate

B
…
Supposons A:
  …
  $[B]$

⇒intro

Supposons A:
  …
  B
  …
…
$[A \Rightarrow B]$

⇒elim

$A \Rightarrow B$
A
B

∧intro

A
B
$A \wedge B$

∧elim

$A \wedge B$
$[A]$
$[B]$

∨intro

A
$[A \vee B]$
$[B \vee A]$

∨elim

$A \vee B$
$A \Rightarrow C$
$B \Rightarrow C$
C

¬intro

$A \Rightarrow \bot$
$\neg A$

¬elim

A
$\neg A$
$\bot$

¬¬elim

$\neg\neg A$
A

absurd

⇔intro

⇔elim

$\neg A \Rightarrow \bot$	$A \Rightarrow B$	$A \Leftrightarrow B$
A	$B \Rightarrow A$	$[A \Rightarrow B]$
	$A \Leftrightarrow B$	$[B \Rightarrow A]$

Calcul des prédicats avec =, et $\in$

Soit un type S et une formule A and et une variable x inutilisée qui n'apparait pas dans S.
On est libre d'ignorer $\in S$, si on le souhaite.

∀sub	**∀restate**	Préconditions:
	A	x n'apparait pas dans A
Soit $x \in S$:	…	
$[x \in S]$	Soit $x \in S$:	
	…	
	$[A]$	

∀intro	**∀elim**	Préconditions:
Soit $x \in S$:	$\forall x \in S\ A$	Substitution licite
…	…	
A	$t \in S$	
…	…	
…	$A[x \coloneqq t]$	
$\forall x \in S\ A$		

∃intro	**∃elim**	Préconditions
$t \in S$	$\exists x \in S\ A$	y variable inutilisée
…	On choisit $y \in S$ t.q. $A[x \coloneqq y]$	Substitution licite
$A[x \coloneqq t]$	$[y \in S]$	
…	$[A[x \coloneqq y]]$	
$\exists x \in S\ A$		

=intro	**=elim**	Préconditions
	$t = u$	Substitutions licites
$[t = t]$	$A[x \coloneqq t]$	
	$A[x \coloneqq u]$	

On peut ajouter des règles de renommage, redondantes certes mais raccourcissant la longueur des preuves.

∀rename	**∃rename**	Préconditions
$\forall x \in S\ A$	$\exists x \in S\ A$	y variable inutilisée
$[\forall y \in S\ A[x \coloneqq y]]$	$[\exists y \in S\ A[x \coloneqq y]]$	Substitution licite

Formes courtes.

Par commodité on écrit "$\forall x, y \in S\ (P(x, y))$" pour "$\forall x \in S\ (\forall y \in S\ (P(x, y)))$", et similairement pour davantage de variables ou pour "$\exists$".
"$\exists!\, x \in S\ (P(x))$" signifie "$\exists x \in S\ (P(x) \wedge \forall y \in S\ (P(y) \Rightarrow x = y))$".

Exemple de l'écriture d'une preuve a la Fitch.
Soit S, T deux types quelconques et $P(x, y)$ une formule (a 2 variables libres x, y).
Prouvons : $\exists x \in S\ (\forall y \in T\ (P(x, y))) \Rightarrow \forall y \in T\ (\exists x \in S\ (P(x, y)))$

Premièrement en écrivant toutes les conclusions (avec crochets):

Supposons $\exists x \in S\ (\forall y \in T\ (P(x, y)))$: [⇒sub]
  $\exists x \in S\ (\forall y \in T\ (P(x, y)))$. [⇒sub]
  On choisit $a \in S$ tel que $\forall y \in T\ (P(a, y))$. [∃elim]
  $a \in S$. [∃elim]
  $\forall y \in T\ (P(a, y))$. [∃elim]
  $\forall z \in T\ (P(a, z))$. [∀rename]
  Soit $y \in T$: [∀sub]
    $y \in T$. [∀sub]
    $\forall z \in T\ (P(a, z))$. [∀restate]
    $y \in T$. [restate]
    $P(a, y)$. [∀elim]
    $a \in S$. [∀restate]
    $\exists x \in S\ (P(x, y))$. [∃intro]
  $\forall y \in T\ (\exists x \in S\ (P(x, y)))$. [∀intro]
$\exists x \in S\ (\forall y \in T\ (P(x, y))) \Rightarrow \forall y \in T\ (\exists x \in S\ (P(x, y)))$. [⇒intro]

Finalement en enlevant tous les formules entre crochets jugées encombrantes:

Supposons $\exists x \in S\ (\forall y \in T\ (P(x, y)))$: [⇒sub]
  On choisit $a \in S$ tel que $\forall y \in T\ (P(a, y))$. [∃elim]
  Soit $y \in T$: [∀sub]
    $P(a, y)$. [∀elim]
    $\exists x \in S\ (P(x, y))$. [∃intro]
  $\forall y \in T\ (\exists x \in S\ (P(x, y)))$. [∀intro]
$\exists x \in S\ (\forall y \in T\ (P(x, y))) \Rightarrow \forall y \in T\ (\exists x \in S\ (P(x, y)))$. [⇒intro]

Cette preuve finale est beaucoup plus claire, intuitive et proche de la rédaction humaine, et fait encore apparaitre à chaque ligne exactement une règle d'inférence.

Système de déduction de Hilbert. (beaucoup d'axiomes, peu de règles)
Une unique règle d'inférence : le modus ponens : $A, A \to B \vdash B$
Schémas d'axiomes du calcul propositionnel (Lukasiewicz) :
$A \to (B \to A)$
$\big(A \to (B \to C)\big) \to \big((A \to B) \to (A \to C)\big)$
$(\neg A \to \neg B) \to \big((\neg A \to B) \to A\big)$
Schémas d'axiomes du calcul des prédicats :
$\exists v\ \varphi \leftrightarrow \neg \forall v \neg \varphi$
$\forall v\ (\varphi \to \psi) \to (\varphi \to \forall v\ \psi)$ pourvu que v n'a pas d'occurrence libre dans φ
$\forall v\ \varphi \to \varphi[v \coloneqq t]$ pourvu que la substitution $\varphi[v \coloneqq t]$ soit licite
Schémas d'axiomes de l'égalité :
Pour $x, y, z, x_1, \dots, x_n, y_1, \dots, y_n$ variables, f symbole de fonction, R symbole de relation alors on a les axiomes :
$x = x$
$x = y \to y = x$
$(x = y \wedge y = z) \to x = z$
$(x_1 = y_1 \wedge \dots \wedge x_n = y_n) \to f(x_1, \dots, x_n) = f(y_1, \dots, y_n)$
$(x_1 = y_1 \wedge \dots \wedge x_n = y_n) \to \big(R(x_1, \dots, x_n) \to R(y_1, \dots, y_n)\big)$

Autres systèmes formels en logique.
Déduction naturelle sans séquents (avec déchargement d'hypothèses). (1.3.6. livre de David et al.)
Calcul des séquents. (permet de chercher une preuve automatiquement et donc facilite la preuve du théorème de complétude du calcul propositionnel).

2. Sémantique de la logique.

Divagations : jusqu'à présent on n'a fait que décrire la syntaxe de la logique sans s'occuper du sens. La syntaxe montre comment on pourrait physiquement construire une machine qui par le pur calcul, pourrait vérifier mécaniquement la correction d'une théorie mathématique fondée sur la logique, sans se préoccuper aucunement de l'intuition ou du sens de ces formules.
Ainsi, après avoir formalisé les mathématiques et disposant d'une machine (ou d'un cerveau, d'un crayon et d'un papier) comme moyen de vérification mécanique, on peut à présent formaliser l'étude de la logique puis des théories mathématiques, dans un langage mathématique « primaire » vérifiable. Il faut ainsi distinguer deux niveaux de langage : le langage de la théorie étudiée, et le langage mathématique primaire qui sert à décrire et à prouver des théorèmes sur cette théorie étudiée. Le langage primaire est aussi appelé méta langage.
Les définitions et théorèmes en syntaxe et sémantique sont donc des théorèmes en méta langage.
La sémantique cherche à exprimer une correspondance entre une formule du langage étudié, et le sens souhaité de cette formule exprimé en méta langage. Par exemple « $\forall x\ P(x)$ » est jugé vrai si pour tout $x\ \ P(x)$ est vrai. Mais ce lien n'a en fait lieu que strictement formellement entre le langage et le métalangage mathématique. Il semble qu'on n'a jamais vraiment de garantie que les concepts telles qu'on les définit, correspondent bel et bien au sens que notre intuition humaine leur donne. Cependant cette correspondance formelle permet quand même de faire émerger une théorie intéressante.

Une structure = un modèle = une interprétation $\boldsymbol{M}$ d'un langage du 1er ordre de signature $\boldsymbol{L}$, correspond à la donnée de :
1) - un ensemble X non vide.
2) - pour chaque $R \in R_n, n \geq 0$ une application $R^M : X^n \to \{0,1\}$, càd une partie $R^M \subseteq X^n$.
Remarque : pour $n = 0$ cela revient à fixer une valuation logique $\phi^M : P \to \{0,1\}$.
3) - pour chaque $f \in F_n, n \geq 0$, une application $f^M : X^n \to X$
Remarque : pour $n = 0$ cela revient à fixer pour chaque constante $c \in F_0$, un élément $c^M \in X$.
4) - une fonction $\rho^M : V \to X$ appelée valuation.
En logique propositionnelle, M se réduit à la donnée d'une valuation logique $\phi^M : P \to \{0,1\}$
Cardinal d'un modèle = cardinal de son ensemble.
Cardinal d'un signature = nombre de formules du langage engendré par la signature
Intuitivement, le modèle donne un sens à la syntaxe, et permet de définir un concept de vérité comme correspondance entre la syntaxe et son sens.
Une notation de la forme $\boldsymbol{M[v := d]}$ désigne le modèle M où on remplace sa valuation par $\rho' : V \to X$ qui coincide avec ρ sauf en v ou elle vaut d. On généralise ce genre de notations.
La **valeur d'un terme $\boldsymbol{t}$ dans un modèle $\boldsymbol{M}$** est :
Si $t = c \in C$ alors $[\![\boldsymbol{t}]\!]_{\boldsymbol{M}} = c^M$
Si $t = v \in V$ alors $[\![\boldsymbol{t}]\!]_{\boldsymbol{M}} = \rho^M(v)$
Si $t = f t_1 \dots t_n$ alors $[\![\boldsymbol{t}]\!]_{\boldsymbol{M}} = f^M([\![t_1]\!]_M, \dots, [\![t_n]\!]_M)$
La valeur d'un terme t ne dépend que de la valeur de ρ^M sur ses variables :
Pour $\rho_1, \rho_2 : V \to X$ valuations coïncidant <u>sur les variables de t</u> alors $[\![t]\!]_{M[\rho := \rho_1]} = [\![t]\!]_{M[\rho := \rho_2]}$
Si on met à part la valuation ρ, on peut aussi voir $[\![\boldsymbol{t}]\!]_{\boldsymbol{M}}$ comme la fonction $[\![\boldsymbol{t}]\!]_{\boldsymbol{M}} : M^V \to M : \alpha \mapsto [\![\boldsymbol{t}]\!]_{\boldsymbol{M}[\boldsymbol{\rho := \alpha}]}$
La **valeur = vérité d'une formule $\boldsymbol{f}$ dans un modèle $\boldsymbol{M}$** notée $[\![\boldsymbol{f}]\!]_{\boldsymbol{M}}$ est un élément de $\{0,1\}$ défini inductivement par :
Si $f = \bot$ alors $[\![\boldsymbol{f}]\!]_{\boldsymbol{M}} = 0$, Si $f = \top$ alors $[\![\boldsymbol{f}]\!]_{\boldsymbol{M}} = 1$

Si $f = A \in P$ alors $[\![\boldsymbol{f}]\!]_{\boldsymbol{M}} = \phi^M(A)$
Si $f = Rt_1 \dots t_n \in Atom$ alors $[\![\boldsymbol{f}]\!]_{\boldsymbol{M}} = R^M([\![t_1]\!]_M, \dots, [\![t_n]\!]_M) \in \{0,1\}$
Si $f = f_1 \wedge f_2$ alors $[\![\boldsymbol{f}]\!]_{\boldsymbol{M}} = 1$ ssi ($[\![f_1]\!]_M = 1$ et $[\![f_1]\!]_M = 1$)
Si $f = f_1 \vee f_2$ alors $[\![\boldsymbol{f}]\!]_{\boldsymbol{M}} = 1$ ssi ($[\![f_1]\!]_M = 1$ ou $[\![f_1]\!]_M = 1$)
Si $f = f_1 \Rightarrow f_2$ alors $[\![\boldsymbol{f}]\!]_{\boldsymbol{M}} = 1$ ssi ($[\![f_1]\!]_M = 1$ implique $[\![f_1]\!]_M = 1$)
Si $f = \neg f_1$ alors $[\![\boldsymbol{f}]\!]_{\boldsymbol{M}} = 1 - [\![f_1]\!]_M$
Si $f = \forall x\, f_1$ alors $[\![\boldsymbol{f}]\!]_{\boldsymbol{M}} = 1$ ssi pour tout $d \in M$ $[\![f_1]\!]_{M[x:=d]} = 1$
Si $f = \exists x\, f_1$ alors $[\![\boldsymbol{f}]\!]_{\boldsymbol{M}} = 1$ ssi il existe au moins un $d \in M$ $[\![f_1]\!]_{M[x:=d]} = 1$
Pour une formule <u>close</u> f, la vérité de f ne dépend pas de ρ^M :
Pour $\rho_1, \rho_2 : V \to X$ valuations, alors $[\![f]\!]_{M[\rho:=\rho_1]} = [\![f]\!]_{M[\rho:=\rho_2]}$.
Compatibilité avec la substitution. Pour $t(x_1, \dots, x_n)$ terme, $f(v, x_1, \dots, x_n, u_1, \dots, u_p)$ une formule, telle que la substitution $f[v := t]$ est licite, alors $[\![f[v := t]]\!]_M = [\![f]\!]_{M[v:=[\![t]\!]_M]}$
Pour f formule, g formule, p constante propositionnelle, alors $[\![f[p := g]]\!]_M = [\![f]\!]_{M[p:=[\![g]\!]_M]}$

Sémantique des théories logiques. Soit <u>un langage du premier ordre</u> compris comme un système formel.
Une **formule** est un élément de F.
Une **théorie** est un ensemble de formules <u>closes</u> $T \subseteq F$.
Une formule $\boldsymbol{f}$ est vraie = satisfaite, dans un modèle $\boldsymbol{M}$ ssi sa valeur est $[\![f]\!]_M = 1$.
On note $\vDash_{\boldsymbol{M}} \boldsymbol{f}$. On dit aussi que $\boldsymbol{M}$ **est un modèle de** $\boldsymbol{f}$.
Une formule $\boldsymbol{f}$ est fausse dans un modèle $\boldsymbol{M}$ ssi sa valeur est $[\![f]\!]_M = 0$.
On peut noter $\vDash_M \neg f$ car une formule est fausse ssi sa négation logique est vraie.
Une **théorie $\boldsymbol{T}$ est vraie = satisfaite, dans un modèle $\boldsymbol{M}$** ssi <u>toutes</u> ses formules sont vraies dans ce modèle.
On note $\vDash_{\boldsymbol{M}} \boldsymbol{T}$. On dit aussi que $\boldsymbol{M}$ **est un modèle de** $\boldsymbol{T}$.
Une **théorie $\boldsymbol{T}$ est fausse pour une modèle $\boldsymbol{M}$** ssi elle a <u>au moins une</u> formule fausse pour M.
Une **formule est universellement vraie = tautologie** ssi elle vraie dans <u>tout</u> modèle.
On note $\vDash^* \boldsymbol{f}$, ou $[\![\boldsymbol{f}]\!] = 1$.
Une **formule est universellement fausse = contradiction** ssi elle fausse dans <u>tout</u> modèle.
On note $\vDash^* \neg \boldsymbol{f}$, ou $[\![\boldsymbol{f}]\!] = 0$.
Une **théorie est universellement vraie = tautologique** ssi elle vraie dans <u>tout</u> modèle.
On note $\vDash^* \boldsymbol{T}$.
Une formule est une tautologie ssi sa négation est une contradiction.
Une formule est une tautologie ssi sa clôture universelle l'est.
Une formule $\boldsymbol{f}$ implique logiquement une formule $\boldsymbol{g}$ <u>dans un modèle $\boldsymbol{M}$</u> ssi (f vraie dans M entraine g vraie dans M) ssi $f \Rightarrow g$ est vraie dans M càd $\vDash_M (f \Rightarrow g)$
On note $\boldsymbol{f} \vDash_{\boldsymbol{M}} \boldsymbol{g}$
Une formule $\boldsymbol{f}$ implique logiquement une formule $\boldsymbol{g}$ ssi (tout modèle de f est un modèle de g) ssi $f \Rightarrow g$ est une tautologie càd $\vDash^* (f \Rightarrow g)$
On note $\boldsymbol{f} \vDash^* \boldsymbol{g}$
Une théorie $\boldsymbol{T}$ implique logiquement une théorie $\boldsymbol{T'}$ <u>dans un modèle $\boldsymbol{M}$</u> ssi (T vraie dans M entraine T' vraie dans M)
On note $\boldsymbol{T} \vDash_{\boldsymbol{M}} \boldsymbol{T'}$
Une théorie $\boldsymbol{T}$ implique logiquement une théorie $\boldsymbol{T'}$ ssi (tout modèle de T est un modèle de T')
On note $\boldsymbol{T} \vDash^* \boldsymbol{T'}$
Pour une formule close f et une théorie T on a ($T \vDash^* f$ ssi $T \cup \{\neg f\}$ est contradictoire).
Deux formules $\boldsymbol{f}, \boldsymbol{g}$ sont logiquement équivalentes <u>dans un modèle $\boldsymbol{M}$</u> ssi (f vraie dans M équivaut à g vraie dans M) ssi ($f \vDash_M g$ et $g \vDash_M f$) ssi $f \Leftrightarrow g$ est vraie dans M càd $\vDash_M (f \Leftrightarrow g)$
On note $\boldsymbol{f} \equiv_{\boldsymbol{M}} \boldsymbol{g}$
Deux formules $\boldsymbol{f}, \boldsymbol{g}$ sont logiquement équivalentes ssi (pour tout modèle M, f vrai dans M ssi g

vraie dans M) ssi ($f \vDash^* g$ et $g \vDash^* f$) ssi $f \Leftrightarrow g$ est une tautologie càd $\vDash^* (f \Leftrightarrow g)$
On note $\boldsymbol{f \equiv g}$
Deux théories $\boldsymbol{T}, \boldsymbol{T'}$ sont logiquement équivalentes <u>dans un modèle $\boldsymbol{M}$</u> ssi (T vraie dans M équivaut à T' vraie dans M) ssi ($T \vDash_M T'$ et $T' \vDash_M T$)
On note $\boldsymbol{T \equiv_M T'}$
Deux théories $\boldsymbol{T}, \boldsymbol{T'}$ sont logiquement équivalentes ssi T et T' ont les mêmes modèles ssi ($T \vDash^* T'$ et $T' \vDash^* T$)
On note $\boldsymbol{T \equiv T'}$
La théorie induite par une théorie $\boldsymbol{T}$ dans un modèle $\boldsymbol{M}$, est l'ensemble $\boldsymbol{Conseq_M(T)}$ des formules (closes) qu'elle implique logiquement pour ce modèle M. $\boldsymbol{Conseq_M(T)} = \{f \in F | \, T \vDash_M f\}$
La théorie induite par une théorie $\boldsymbol{T}$, est l'ensemble $\boldsymbol{Conseq^*(T)}$ des formules (closes) qu'elle implique logiquement. $Conseq^*(T) = \bigcap_M Conseq_M(T) = \{f \in F | \, T \vDash^* f\}$
La **théorie engendrée par une théorie $\boldsymbol{T}$**, est l'ensemble des formules qu'elle prouve. $\boldsymbol{Thm(T)} = \{f \in F \mid T \vdash f\}$.
Une **formule close $\boldsymbol{f}$ est prouvable = est un théorème d'une théorie $\boldsymbol{T}$** ssi $T \vdash f$ càd $f \in Thm(T)$.
Une **formule close $\boldsymbol{f}$ n'est pas prouvable dans une théorie $\boldsymbol{T}$** ssi $T \nvdash f$ càd $f \notin Thm(T)$
Une **formule close $\boldsymbol{f}$ est décidable dans une théorie $\boldsymbol{T}$** ssi $T \vdash f$ ou $T \vdash \neg f$
Une **formule close $\boldsymbol{f}$ est indécidable dans une théorie $\boldsymbol{T}$** ssi $T \nvdash f$ et $T \nvdash \neg f$
Une **théorie est sémantiquement consistante = satisfaisable** ssi elle n'est pas contradictoire ($T \nvDash^* \bot$) ssi elle admet au moins un modèle.
Une **théorie est sémantiquement inconsistante = contradictoire = universellement fausse** ssi elle fausse pour tout modèle, ssi elle entraine logiquement une contradiction : $T \vDash^* \bot$ ssi $\exists f$ formule close telle que $T \vDash^* f$ et $T \vDash^* \neg f$.
Une **théorie est sémantiquement complète** ssi elle est sémantiquement consistante et toute formule close ou sa négation en est une conséquence logique, ssi $\forall f$ close, <u>ou bien</u> $T \vDash^* f$, <u>ou bien</u> $T \vDash^* \neg f$.
Une **théorie est syntaxiquement consistante = cohérente** ssi elle n'est pas incohérente. ($T \nvdash \bot$)
Une **théorie est syntaxiquement inconsistante = incohérente** ssi elle prouve une contradiction $T \vdash \bot$ ssi il existe une formule close f telle que $T \vdash f$ et $T \vdash \neg f$.
Une **théorie est syntaxiquement complète** ssi elle est syntaxiquement consistante <u>et</u> toute formule close y est décidable, ssi $\forall f$ close, <u>ou bien</u> $T \vdash f$, <u>ou bien</u> $T \vdash \neg f$.
Un langage $\boldsymbol{L}$ est récursivement énumérable = semi-décidable = $\boldsymbol{RE}$ ssi c'est le langage accepté par une MT ssi il est énuméré par une MT énumératrice.
Un langage $\boldsymbol{L}$ est récursif = décidable = $\boldsymbol{R}$ ssi c'est le langage accepté par une MT <u>qui s'arrête toujours</u>.
Une **théorie $\boldsymbol{T}$ est récursivement axiomatisable** ssi T a une partie récursive, qui engendre $Thm(T)$
Une **théorie $\boldsymbol{T}$ est RE axiomatisable** ssi T contient une partie RE, qui engendre $Thm(T)$
Une théorie T est RE axiomatisable est récursivement axiomatisable. Donc RE axiomatisable inutile.
Une **théorie $\boldsymbol{T}$ est récursive** ssi le problème d'entrée une formule close $f \in F$, de question $f \in T$? est décidable ssi T est un langage récursif. (le Nour ne définit que théorie récursive, pour simplifier).
Une **théorie $\boldsymbol{T}$ est (syntaxiquement) décidable** ssi le problème de décision d'input une formule close quelconque $f \in F$, de question « $T \vdash f$? » càd « $f \in Thm(T)$?» est décidable.
Autrement dit ssi $Thm(T)$ l'est càd $Thm(T)$ est récursif.
Une théorie T est **sémantiquement décidable** ssi le problème de décision d'input une formule close quelconque $f \in F$, de question « $T \vDash^* f$? » est décidable.
Une théorie T est récursivement axiomatisable ssi $Thm(T)$ est RE
Une théorie T décidable est récursivement axiomatisable. (car $Thm(T)$ est R donc RE)
Une théorie T récursivement axiomatisable et syntaxiquement complète, est décidable.
Pour une formule close f et une théorie T, $T \vdash f$ ssi $T \cup \{\neg f\}$ est syntaxiquement inconsistante.
Pour une formule close f et une théorie T, $T \vDash^* f$ ssi $T \cup \{\neg f\}$ est sémantiquement inconsistante.
Si $T \vdash f$ alors il existe une partie finie $T_0 \subseteq T$ telle que $T_0 \vdash f$. (par définition d'une preuve).

Une théorie dont toutes les parties finies sont syntaxiquement consistantes, est syntaxiquement consistante. (contraposée de la propriété précédente) (ce résultat est une version syntaxique du th de compacité)

Correction de la logique. Pour T théorie du 1er ordre, et f formule, si $T \vdash f$ alors $T \vDash^* f$. (par induction sur règles du système formel)

Autrement dit une théorie consistante sémantiquement l'est syntaxiquement.

Théorème de complétude (Gödel). Pour T théorie du 1er ordre, et f formule, si $T \vDash^* f$ alors $T \vdash f$.

Autrement dit une théorie consistante syntaxiquement l'est sémantiquement.

Sous complétude, donc par ex pour une théorie du 1er ordre, on peut alors identifier consistance syntaxique et sémantique, on peut identifier prouvabilité et vérité dans tous modèles. $\vDash^* = \vdash$

Théorème de compacité. Une théorie du 1er ordre dont toutes les parties finies sont sémantiquement consistantes, est sémantiquement consistante. (conséquence immédiate du théorème de complétude)

Autrement dit une théorie du 1er ordre dont toute partie finie admet un modèle, admet aussi un modèle.

Lemmes pour le th de complétude et le th de compacité.

Une **théorie T du 1er ordre est de Henkin** ssi pour toute formule $f(x)$ à une variable libre x, il existe $c \in F_0$ tel que, si $T \vdash \exists x\, f$ alors $T \vdash T[x \coloneqq c]$ càd tel que $T \vdash \exists x\, f \to T[x \coloneqq c]$

Une théorie du 1er ordre syntaxiquement consistante, peut être étendue en une théorie de Henkin syntaxiquement complète sur un langage du 1er ordre plus grand.

Une théorie du 1er ordre de Henkin syntaxiquement complète admet un modèle. (syntaxique)

Une théorie du 1er ordre syntaxiquement consistante admet un modèle.

Théorème de déduction. Pour $\varphi \in F$ une formule close. $T \cup \{\varphi\} \vdash \psi$ alors $T \vdash \varphi \to \psi$

Théorème absurde. $T \vdash \varphi$ ssi $T \cup \neg\varphi \vdash \bot$

T syntaxiquement complète ssi $Thm(T)$ est maximal pour l'inclusion.

Thm(T) peut être inclus dans une théorie maximal cohérente, (lemme de Zorn) fournit une autre preuve du théorème de complétude.

Arithmétique de Robinson PA- de signature $(0, S, +, \times)$:

$\forall x\ \left(\neg\left(0 = S(x)\right)\right)$

$\forall x, y\ (S(x) = S(y) \Rightarrow x = y)$

$\forall x\ (x + 0 = x)$

$\forall x, y\ \left(x + S(y) = S(x + y)\right)$

$\forall x\ (x \times 0 = 0)$

$\forall x, y\ (x \times S(y) = x \times y + x)$

Schéma d'axiomes de récurrence du 1er ordre :

Pour chaque formule du 1er ordre de la forme $\varphi(x, y_1, \dots, y_k)$, on ajoute l'axiome :

$\forall \bar{y}\ \left(\left(\varphi(0, \bar{y}) \wedge \forall x\ \left(\varphi(x, \bar{y}) \Rightarrow \varphi(S(x), \bar{y})\right)\right) \Rightarrow \forall x\ \left(\varphi(x, \bar{y})\right)\right)$

Arithmétique de Péano PA = PA- + Schéma d'axiomes de récurrence

Arithmétique de Presburger PB de signature $(0, S, +)$:

$\forall x\ \left(\neg\left(0 = S(x)\right)\right)$

$\forall x, y\ (S(x) = S(y) \Rightarrow x = y)$

$\forall x\ (x + 0 = x)$

$\forall x, y\ \left(x + S(y) = S(x + y)\right)$

+ Schéma d'axiomes de récurrence du 1er ordre

Il existe des modèles non standards.
Une théorie consistante contenant PA est syntaxiquement indécidable.
Corollaire : PA est syntaxiquement indécidable.

1er théorème d'incomplétude de Gödel. Une théorie, récursivement axiomatisable, contenant PA^-, et syntaxiquement consistante, est syntaxiquement incomplète.
Il y a donc des énoncés indécidables en mathématiques, donc vrai dans certains modèles mais pas dans d'autres.
2e théorème d'incomplétude de Gödel. Pour une théorie T, récursivement axiomatisable, contenant PA, et syntaxiquement consistante, « T est syntaxiquement consistante » est exprimable dans T, mais ne peut pas être prouvé dans T.
On ne peut pas avoir de preuve que les mathématiques soient dépourvues de contradiction.
Tarski 1936. La théorie PA est indécidable.
L'arithmétique de Presburger est syntaxiquement complète.
La théorie du 1er ordre de l'arithmétique de Presburger est sémantiquement décidable (1929)*.
L'arithmétique de Presburger ne peut pas quantifier des suites infinies, donc on ne peut même pas définir la multiplication.
L'arithmétique de Péano peut quantifier des suites infinies, donc on peut y définir la puissance, la factorielle, les outils usuels de la combinatoire...
Théorème de Church. Toute théorie contenant $=$ et au moins un autre prédicat binaire, est indécidable.
Théorème de non définissabilité de la vérité de Tarski.
Axiomatisation finie
« être un ensemble a au moins n éléments » est finiment axiomatisable.
« être un ensemble a exactement n elements » est finiment axiomatisable.
« être un ensemble fini » n'est pas finiment axiomatisable.
Donc « être un ensemble infini » n'est pas finiment axiomatisable.
Donc PA n'est pas finiment axiomatisable.
L'arithmétique de Robinson est finiment axiomatisable.

Un **littéral** est soit p soit $\neg p$ où $p \in P$.
Une **clause (disjonctive)** est une disjonction finie de littéraux.
Une **clause conjonctive** est une conjonction finie de littéraux.
Une **formule est sous DNF** ssi elle est une disjonction finie de clauses conjonctives.
Une **formule est sous CNF** ssi elle est une conjonction finie de clauses disjonctives.
Toute formule du calcul prop est équivalente à une formule CNF, et à une formule DNF.
De plus pour toute table de vérité $T \in \{0,1\}^{\{0,1\}^P}$, il existe f une CNF/ ou une DNF telle que $TF(f) = T$
« Est-ce qu'une formule DNF φ du calcul prop est satisfaisable ?» est P
De même que sa reformulation « Est-ce qu'une formule CNF φ du calcul prop est tautologique ?»
« Est-ce qu'une formule CNF φ du calcul prop est satisfaisable ?» est NP-complet
« Existe-t-il une formule CNF qui est équi-satisfaisable à une formule φ du calcul prop ? » est P.

Pour toute formule F du calcul prop. construite avec les connecteurs $\wedge$,$\vee$, $\neg$ il existe une formule G sur un ensemble P' de symbole propositionnels tels que
1. F est satisfaisable ssi G l'est,
2. $|P'| = |sf(F)|$ $(\leq longueur(F))$
3. G est sous CNF avec # clauses de $G \leq 3|sf(F)| + 1$ et chaque clause de G contient au plus 3 littéraux. (réduction SAT à 3SAT)

L'unification

L'algorithme élémentaire d'unification

Soit un langage du premier ordre $L = ((F_n)_{n\in\mathbb{N}}, (R_n)_{n\in\mathbb{N}}, V)$.

Une substitution sur un langage du premier ordre $\boldsymbol{L}$ est une application $\sigma: V \to \tau(L)$

Pour un terme $u \in \tau(L)$, et une substitution σ sur L, on note $\boldsymbol{u[\sigma]}$ le terme u dans lequel toute occurrence d'une variable $x \in V$ est remplacée par $\sigma(x)$.

On note $\boldsymbol{id}$ la substitution $\sigma: V \to \tau(L): x \mapsto x$.

On note $[\boldsymbol{x_1} \coloneqq \boldsymbol{t_1}, \dots, \boldsymbol{x_n} \coloneqq \boldsymbol{t_n}]$ l'unique substitution σ telle que pour $i \in [\![1, n]\!]$, $x_i[\sigma] = t_i$ et $y[\sigma] = y$ pour les autres variables.

Pour σ, σ' 2 substitutions sur L, et un terme $u \in \tau(\mathrm{L})$, $u[\sigma][\sigma'] = u[\sigma' \circ \sigma]$.

Une substitution $\boldsymbol{\sigma}$ sur $\boldsymbol{L}$ unifie deux termes $\boldsymbol{u, v} \in \boldsymbol{\tau(L)}$ ssi $u[\sigma] = v[\sigma]$.

Deux termes sont unifiables ssi il existe une substitution qui les unifie.

Le terme $\boldsymbol{u} \in \boldsymbol{\tau(L)}$ filtre le terme $\boldsymbol{v} \in \boldsymbol{\tau(L)}$ ssi il existe une substitution σ sur L telle que $u[\sigma] = v$.

Un unificateur $\boldsymbol{\sigma}$ de deux termes $\boldsymbol{u, v} \in \boldsymbol{\tau(L)}$ est un m.g.u. ssi $\forall \sigma'$ unificateur de u, v, $\exists \sigma''$ substitution, $\sigma' = \sigma'' \circ \sigma$ càd ssi tout autre unificateur s'obtient en composant à gauche du m.g.u. par une substitution.

Déf-prop : Deux termes $u, v \in \tau(L)$ unifiables admettent un m.g.u. unique noté $\boldsymbol{mgu(u, v)}$.

Une équation de la forme $\boldsymbol{u \sim v}$ où $\boldsymbol{u, v} \in \boldsymbol{\tau(L)}$ est résolue par une substitution $\boldsymbol{\sigma}$ ssi σ unifie u et v càd ssi $u[\sigma] = v[\sigma]$.

Attention même si (σ résout $u \sim v$ ssi σ résout $v \sim u$), on identifie pas les équations $u \sim v$ et $v \sim u$.

Les équations sont donc des objets syntaxiques orientés.

Un ensemble d'équation $\{\boldsymbol{u_1} \sim \boldsymbol{v_1}, \dots, \boldsymbol{u_n} \sim \boldsymbol{v_n}\}$ est résolu par une substitution $\boldsymbol{\sigma}$ ssi σ unifie u_1 et v_1, σ unifie u_2 et v_2, …, σ unifie u_n et v_n.

Algorithme d'unification élémentaire : entrée E un ensemble d'équation

L'algorithme construit une suite (E_n, σ_n) par récurrence.

$E_0 \coloneqq E, \sigma_0 \coloneqq id$. Pour $n \in \mathbb{N}$:

Si $E_n = E' + \{f(u_1, \dots, u_q) \sim g(v_1, \dots, v_p)\}$ avec $f \neq g$ alors échec (clash)

Si $E_n = E' + \{f(u_1, \dots, u_p) \sim f(v_1, \dots, v_p)\}$ alors $E_{n+1} \coloneqq E' + \{u_1 \sim v_1, \dots, u_p \sim v_p\}$ et $\sigma_{n+1} = \sigma_n$

Si $E_n = E' + \{x \sim x\}$ alors $E_{n+1} \coloneqq E'$ et $\sigma_{n+1} = \sigma_n$

Si $E_n = E' + \{x \sim u\}$ (ou $\{u \sim x\}$) avec $u \neq x$, et x apparait dans u, alors échec (occur-check)

Si $E_n = E' + \{x \sim u\}$ (ou $\{u \sim x\}$) avec $u \neq x$, et x n'apparait pas dans u, $E_{n+1} \coloneqq E'[x \coloneqq u]$ et $\sigma_{n+1} = [x \coloneqq u] \circ \sigma_n$.

Si $E_n = \emptyset$, succès, retourne σ_n.

Théorème : l'algorithme d'unification termine toujours, et si succès renvoie le m.g.u. de E.

Résolution pour la logique propositionnelle.

Un littéral est une formule atomique propositionnelle ou sa négation.

Si A est une formule atomique, on note $\overline{\boldsymbol{A}} = \neg A$ et $\overline{\neg \boldsymbol{A}} = A$.

Une **clause** est un ensemble fini de littéraux $L_1, \dots, L_n$.

Une clause s'interprète comme la disjonction de ses littéraux.

La règle de résolution propositionnelle a pour prémisses : la clause $C_1 \vee L$; la clause $C_2 \vee \bar{L}$ et pour conclusion : la clause $C_1 \vee C_2$. (on met habituellement des , à la place des $\vee$)

Résolution : on met une formule sous CNF puis on applique l'algorithme de résolution (voir exemple

ci-dessous). Si l'algorithme dérive la clause vide (on parle de réfutation) on en déduit que la formule est contradictoire. Sinon elle est satisfiable. La résolution permet donc de résoudre le problème de satisfiabilité de CNF-SAT.
Ex : $(\neg p \vee \neg q \vee r \vee s) \wedge (\neg p \vee q \vee s) \wedge (\neg q \vee \neg r \vee s) \wedge (\neg p \vee \neg q) \wedge (\neg q \vee \neg s \vee p) \wedge (\neg q \vee p \vee r) \wedge (p \vee q)$.
1. $\bar{p}, \bar{q}, r, s$
2. $\bar{p}, q, s$
3. $\bar{q}, \bar{r}, s$
4. $\bar{p}, \bar{q}$
5. $\bar{q}, \bar{s}, p$
6. $\bar{q}, p, r$
7. p, q
8. $\bar{p}, r, s$ (par la règle de résolution sur q dans 1 et 2)
9. Etc… on choisit des règles.

On observe à chaque étape que la taille des clauses diminue donc l'algorithme doit terminer, soit sur une clause vide, soit sur des variables isolées impossibles à simplifier (il y a alors satisfiabilité).
Il existe des stratégies de choix des variables permettant d'accélérer la résolution en général.
Cependant : Théorème de Haken. En notant $\neg PHP_n$ une formule sous forme CNF niant le principe des tiroirs à l'ordre n on peut montrer que toute réfutation de cette formule par résolution est de longueur au moins $2^{\frac{n}{32}}$ donc la résolution n'est pas toujours efficace.

Résolution au 1er ordre (Dans le Nour mais à éviter, compliqué…)

Un littéral du 1er ordre est une formule atomique du 1er ordre ou sa négation.
Si A est une formule atomique du 1er ordre, on note $\overline{\boldsymbol{A}} = \neg A$ et $\overline{\neg \boldsymbol{A}} = A$.
Une **clause du 1er ordre** est un ensemble fini de littéraux du 1er ordre $L_1, \dots, L_n$.
Une clause du 1er ordre s'interprète comme la clôture universelle de la disjonction de ses littéraux.
La règle de résolution a pour prémisses : la clause C_1, L_1 ; la clause C_2, L_2 ; et la substitution $\sigma = mgu(L_1, \overline{L_2})$ et pour conclusion : la clause $C_1[\sigma], C_2[\sigma]$
La règle de contraction a pour prémisses : la clause C_1, L_1, L_2 ; la substitution $\sigma = mgu(L_1, L_2)$ et pour conclusion : la clause $C_1[\sigma], L_1[\sigma]$.
Dans le cas propositionnel : **La règle de résolution propositionnelle** a pour prémisses : la clause C_1, L_1 ; la clause $C_2, \overline{L_1}$ et pour conclusion : la clause C_1, C_2.
Dans le cas propositionnel : Il n'y a pas besoin de la règle de contraction.
Par exemple : La règle de résolution au 1er ordre sur $R(x), \neg S(y, f(x))$ et $T(y), S\big(a, f(y)\big)$ impose le choix $\sigma = [x \coloneqq a, y \coloneqq a]$ et donc conclut : $R(a), S\big(a, f(a)\big)$.
Lemme. Pour un ensemble de clause E et un littéral clos L, tels qu'on peut dériver la clause vide depuis $E \cup \{L\}$, alors on peut dériver depuis E la clause vide ou un littéral qui filtre $\overline{L}$.
Un ensemble de clauses $\boldsymbol{E}$ est vrai dans une interprétation $\boldsymbol{M}$ ssi toutes les clauses de E sont vraies dans M.
Un ensemble de clauses est contradictoire ssi il n'est vrai dans aucune interprétation.
Un ensemble de clauses est inconsistant si on peut en dériver la clause vide.
Théorème : Un ensemble de clauses est contradictoire ssi il est inconsistant.