LOGIQUE MADHELIS 2022

ANDREW ARANA
UNIVERSITÉ DE LORRAINE

Sujets du cours

(1) La différence entre la logique propositionnelle et la logique du premier ordre
(2) La différence entre la syntaxe et la sémantique
(3) La syntaxe de la logique propositionnelle : comment l'écrire
(4) La sémantique de la logique propositionnelle : la signification, la vérité
(5) La syntaxe de la logique propositionnelle : comment démontrer
(6) La logique du premier ordre : comment l'écrire
(7) La sémantique de la logique du premier ordre : la signification, la vérité
(8) La syntaxe de la logique du premier ordre : comment démontrer
(9) La métathéorie de la logique du premier ordre

1. La différence entre la logique propositionnelle et la logique du premier ordre

Le qualitatif « propositionnelle » signifie que les énoncés se composent de propositions et des *connecteurs* qui relient ces propositions. Une proposition est susceptible d'être vraie ou fausse.

Exemples :

- Nous sommes lundi.
- Charlotte a un cours de latin.
- Nous sommes lundi et Charlotte a un cours de latin.

La logique du premier ordre ajoute à la logique propositionnelle deux quantificateurs : la quantificateur existentielle « il existe » et la quantificateur universelle « pour tout ».

Exemples :

- Tous les félins sont des animaux.
- Il existe un animal qui n'est pas félin.

Le qualificatif « premier ordre » ici signifie que les quantificateurs range sur d'individus plutôt que des *propriétés* d'individus. C'est-à-dire, on dit « tout animal qui est mammifère »

dans la logique du premier ordre et pas « tout adaptation d'un animal vise à la valeur sélective » ; la dernière est un expression de la logique du *second* ordre, parce que une adaptation est une *propriété* d'un individu plutôt qu'un *individu propre*.

2. La différence entre la syntaxe et la sémantique

La syntaxe signifie concerne écrire les énoncés *bien formés*, et comment les *démontrer*. La sémantique concerne la *signification* et la *vérité* des énoncés. Alors : il s'agit de forme et contenu, de démonstration et vérité.

3. La syntaxe de la logique propositionnelle

Exemples d'énonces de la logique propositionnelle :

- Nous sommes lundi.
- Charlotte a un cours de latin.
- Nous ne sommes pas lundi.
- Nous sommes lundi et Charlotte a un cours de latin.
- Nous sommes lundi ou Charlotte a un cours de latin.
- Si nous sommes lundi, alors Charlotte a un cours de latin.

Les connecteurs propositionnels informels incluent : et, ou, ne pas, si… alors, seulement si, si et seulement si, donc, bien que, ou bien … ou bien…

Nous étudions la logique *formelle* : c'est-à-dire, nous remplaçons les propositions par des *noms* qui représente les propositions. Alors plutôt que « Charlotte a un cours de latin » on écrit A.

Un énoncé bien formé de la logique propositionnelle formelle se compose de noms propositionnels A, B, C, etc., et les connecteurs propositionnels qui relient les noms : la conjonction $\wedge$, la disjonction $\vee$, la négation $\neg$, et l'implication $\rightarrow$.

Pour nos exemples d'énoncés propositionnels, nous pouvons les écrire dans notre logique formelle par :

- A
- B
- $\neg A$
- $A \wedge B$
- $A \vee B$
- $A \rightarrow B$

Nous allons traiter les démonstrations un peu plus tard.

4. La sémantique de la logique propositionnelle

4.1. **La négation.** P = Nous sommes lundi.

$\neg P$ = Nous ne sommes pas lundi.

Nous pouvons montrer la signification de la négation par une *table de vérité* :

P	$\neg P$
V	**F**
F	**V**

4.2. **La conjonction.** P = Nous sommes lundi.

Q = Charlotte a un cours de latin.

$P \wedge Q$ = Nous sommes lundi et Charlotte a un cours de latin.

P	Q	$P \wedge Q$
V	**V**	**V**
V	**F**	**F**
F	**V**	**F**
F	**F**	**F**

4.3. **La disjonction.** $P \vee Q$ = Nous sommes lundi ou Charlotte a un cours de latin.

P	Q	$P \vee Q$
V	**V**	**V**
V	**F**	**V**
F	**V**	**V**
F	**F**	**F**

4.4. **Le conditionnel.** $P \to Q$ = Si nous sommes lundi, alors Charlotte a un cours de latin.

P	Q	$P \to Q$
V	**V**	**V**
V	**F**	**F**
F	**V**	**V**
F	**F**	**V**

Peut-être vous pensez que le deux dernières valeurs doivent être **F**. Pourquoi penser qu'une fausseté implique une vérité ? Si oui, vous penser que le conditionnel devrait signifier une relation causale entre antécédent et conséquent. Nous allons discuter ce type de conditionnel en novembre, par rapport au sujet de la *logique intuitionniste*. Aujourd'hui nous utilisons le type de conditionnel défini par notre table de vérité ; on l'appelle le *conditionnel materiel*.

4.5. **Le biconditionnel.** Les autres connecteurs propositionnels informels peuvent être définis par rapport aux les connecteurs nous avons défini déjà. Par exemple, le biconditionnel :

$P \leftrightarrow Q$ = Nous sommes lundi si et seulement si Charlotte a un cours de latin.

P	Q	$P \leftrightarrow Q$
V	**V**	**V**
V	**F**	**F**
F	**V**	**F**
F	**F**	**V**

Cette proposition $P \leftrightarrow Q$ est *équivalente* à $(P \rightarrow Q) \wedge (Q \rightarrow P)$. *Équivalente* signifie : les deux énoncés ont toujours les mêmes valeurs de vérité. Nous pouvons le vérifier par une table de vérité.

P	Q	$P \leftrightarrow Q$	$P \rightarrow Q$	$Q \rightarrow P$	$(P \rightarrow Q) \wedge (Q \rightarrow P)$
V	**V**	**V**	**V**	**V**	**V**
V	**F**	**F**	**F**	**V**	**F**
F	**V**	**F**	**V**	**F**	**F**
F	**F**	**V**	**V**	**V**	**V**

Parce que les colonnes de $P \leftrightarrow Q$ et $(P \rightarrow Q) \wedge (Q \rightarrow P)$ sont identiques, ligne par ligne, nous disons que les deux sont équivalentes. Pour tout « monde possible » (toute *valuation*), n'importe quels sont les valeurs de vérités des propositions impliquées, les deux propositions composites ont les mêmes valeurs de vérité.

Exemple. $\neg(P \wedge Q)$ et $\neg P \vee \neg Q$

P	Q	$P \wedge Q$	$\neg(P \wedge Q)$	$\neg P$	$\neg Q$	$\neg P \vee \neg Q$
V	**V**	**V**	**F**	**F**	**F**	**F**
V	**F**	**F**	**V**	**F**	**V**	**V**
F	**V**	**F**	**V**	**V**	**F**	**V**
F	**F**	**F**	**V**	**V**	**V**	**V**

Parce que les colonnes de $\neg(P \wedge Q)$ et $\neg P \vee \neg Q$ ont les mêmes valeurs de vérité pour toute valuation, les deux propositions sont équivalentes.

4.6. **Énoncés composites.** Nous pouvons évaluer la valeur de vérité d'autres propositions composites par la même méthode.

P	Q	$P \wedge Q$	$\neg(P \wedge Q)$	$\neg P$	$\neg Q$	$\neg P \vee \neg Q$	$\neg(P \wedge Q) \leftrightarrow (\neg P \vee \neg Q)$
V	**V**	**V**	**F**	**F**	**F**	**F**	**V**
V	**F**	**F**	**V**	**F**	**V**	**V**	**V**
F	**V**	**F**	**V**	**V**	**F**	**V**	**V**
F	**F**	**F**	**V**	**V**	**V**	**V**	**V**

Parce que toutes les valeurs de la colonne de $\neg(P \wedge Q) \leftrightarrow (\neg P \vee \neg Q)$ sont **V**, pour toute valuation, ligne par ligne, nous disons que $\neg(P \wedge Q) \leftrightarrow (\neg P \vee \neg Q)$ est une *tautologie*.

Exemple. Démontrer que $((P \to Q) \leftrightarrow (\neg Q \to \neg P))$ est une tautologie.

P	Q	$\neg P$	$\neg Q$	$(P \to Q)$	$(\neg Q \to \neg P)$	$((P \to Q) \leftrightarrow (\neg Q \to \neg P))$
V	**V**	**F**	**F**	**V**	**V**	**V**
V	**F**	**F**	**V**	**F**	**F**	**V**
F	**V**	**V**	**F**	**V**	**V**	**V**
F	**F**	**V**	**V**	**V**	**V**	**V**

Voilà : la colonne de $((P \to Q) \leftrightarrow (\neg Q \to \neg P))$ est complètement **V**, donc cette formule est une *tautologie*.

Si la colonne de la formule est complètement **V**, c'est une tautologie ; si par contre cette colonne est complètement **F**, on l'appelé une antilogie ; finalement s'il existe au moins un **V** et un **V**, c'est une *formule neutre*.

Exemple. Déterminer si $((P \wedge \neg Q) \wedge (P \to Q))$ est une tautologie, une antilogie, ou une formule neutre.

P	Q	$\neg Q$	$(P \wedge \neg Q)$	$(P \to Q)$	$((P \wedge \neg Q) \wedge (P \to Q))$
V	**V**	**F**	**F**	**V**	**F**
V	**F**	**V**	**V**	**F**	**F**
F	**V**	**F**	**F**	**V**	**F**
F	**F**	**V**	**F**	**V**	**F**

Voilà : la colonne de $((P \wedge \neg Q) \wedge (P \to Q))$ est complètement **F**, donc cette formule est une *antilogie*.

4.7. **Conséquence logique propositionnelle.** Une proposition B est conséquence logique de A (écrit $A \models B$) : si A est vrai, alors B est *nécessairement* vrai. Ou, il est impossible que A soit vrai et B soit faux. La notion de conséquence logique implique une modalité : il est *impossible* que le premier énoncé soit vrai et le second soit faux. Comment déterminer cette impossibilité ? Pour démontrer qu'un énoncé *n'est pas* une conséquence logique d'un autre, il suffit de trouver un contrexemple, une « situation » , une valuation, que démontre qu'il est possible que le premier énoncé soit vrai et le second soit faux. Alors le second n'est pas forcement vrai quand le premier est vrai.

Un exemple : on se demande si « La neige est blanche et l'herbe est verte » est une conséquence logique de « La neige est blanche ». Normalement, oui, on envisage le monde comme ça. Mais est-ce que c'est *nécessaire* ? Peut-on imaginer que l'herbe soit brune quand la neige est blanche ? Tout à fait. Dans cette « situation » le premier énoncé soit vrai et le second soit faux. Donc la réponse est non.

Par contre : on se demande si « La neige est blanche ou l'herbe est verte » est une conséquence logique de « La neige est blanche ». Notre situation en hausse ne marche pas ici : même si

l'herbe soit brune, il reste que la neige est blanche ou l'herbe est verte. Pourquoi ? À cause de la signification de la disjonction, qui dit qu'une disjonction est vraie si l'une ou l'autre partie de cette disjonction est vraie. Donc toute situation dans laquelle la neige est blanche est une situation où la neige est blanche ou l'herbe est verte. Donc la réponse est oui.

Pour notre logique propositionnel formelle, nous pouvons déterminer si $A \models B$ en faisant une table de vérité. Si B est vraie dans toute colonne dont A est vraie, alors $A \models B$.

Exemple. Détermine si $\neg P \wedge \neg Q$ est une conséquence logique de $\neg(P \vee Q)$.

P	Q	$P \vee Q$	$\neg(P \vee Q)$	$\neg P$	$\neg Q$	$\neg P \wedge \neg Q$
V	**V**	**V**	**F**	**F**	**F**	**F**
V	**F**	**V**	**F**	**F**	**V**	**F**
F	**V**	**V**	**F**	**V**	**F**	**F**
F	**F**	**F**	**V**	**V**	**V**	**V**

Il n'existe qu'une valuation dont $\neg(P \vee Q)$ est vraie, et pour celle-ci, $\neg P \wedge \neg Q$ est vraie aussi. Donc si $\neg(P \vee Q)$ est vraie, $\neg P \wedge \neg Q$ est nécessairement vraie. Alors $\neg P \wedge \neg Q$ est une conséquence logique de $\neg(P \vee Q)$.

Nous pouvons demander si une proposition est une conséquence logique de plusieurs propositions aussi. Par exemple, nous pouvons demander si A,$B \models C$ pour trois propositions A, B, C.

Exemple. Déterminer si

$$((P \wedge R) \to Q),(P \to (Q \wedge R)) \models (Q \leftrightarrow R).$$

P	Q	R	$(P \wedge R)$	$(Q \wedge R)$	$((P \wedge R) \to Q)$	$(P \to (Q \wedge R))$	$(Q \leftrightarrow R)$
V	**V**	**V**	**V**	**V**	**V**	**V**	**V**
V	**F**	**V**	**V**	**F**	**F**	**F**	**F**
F	**V**	**V**	**F**	**V**	**V**	**V**	**V**
F	**F**	**V**	**F**	**F**	**V**	**V**	**F**
V	**V**	**F**	**F**	**F**	**V**	**F**	**F**
V	**F**	**F**	**F**	**F**	**V**	**F**	**V**
F	**V**	**F**	**F**	**F**	**V**	**V**	**F**
F	**F**	**F**	**F**	**F**	**V**	**V**	**V**

Alors : il y a deux valuations telles que les prémisses sont vraies mais la conclusion est fausse : la quatrième et la septième. Donc la conclusion n'est pas une conséquence logique des prémisses.

5. La syntaxe de la logique propositionnelle : démonstrations

Ensuite nous introduisons un système formel $\mathcal{F}^P$ de la logique propositionnelle. Ce système est une variante de la *déduction naturelle* donne par Frederic Fitch, un logicien américain.

Le nom « déduction naturelle » signifie que le système est visé à la modélisation de notre raisonnement, plutôt qu'un système choisi pour ses propriétés « métalogiques ».

L'idée est de donner des règles de *l'introduction* et de *l'élimination* des opérations propositionnelles : la conjonction $\wedge$, la disjonction $\vee$, la négation $\neg$, et la fausseté $\bot$. On discutera aussi le conditionnel $\to$.

6. Les règles de la conjonction

On commence avec la règle d'élimination de conjonction.

($\wedge$-élim)

$$\begin{array}{|ll} & P_1 \wedge P_2 \wedge \cdots \wedge P_i \cdots \wedge P_n \\ & \vdots \\ \rhd & P_i \end{array}$$

Ensuite, la règle d'introduction de la conjonction.

($\wedge$-intro)

$$\begin{array}{|ll} & P_1 \\ & \Downarrow \\ & P_n \\ & \vdots \\ \rhd & P_1 \wedge \ldots P_n \end{array}$$

Exemple.

$$\begin{array}{|l} A \wedge B \wedge C \\ \hline C \wedge B \end{array}$$

Démonstration.

1	$A \wedge B \wedge C$	
2	B	$\wedge$-élim: 1
3	C	$\wedge$-élim: 1
4	$C \wedge B$	$\wedge$-intro: 3, 2

6.1. **Les règles de la disjonction.**

On commence avec la règle d'introduction de disjonction.

(∨-intro)

$$\begin{array}{ll} & P_i \\ & \vdots \\ \rhd & P_1 \vee P_2 \vee \cdots \vee P_i \cdots \vee P_n \end{array}$$

Ensuite, la règle d'élimination de la disjonction.

(∨-élim)

$$\begin{array}{ll} & P_1 \vee \cdots \vee P_n \\ & \vdots \\ & \begin{array}{|l} P_1 \\ \hline \vdots \\ S \end{array} \\ & \Downarrow \\ & \begin{array}{|l} P_n \\ \hline \vdots \\ S \end{array} \\ & \vdots \\ \rhd & S \end{array}$$

Ce règle introduit la notion d'une *sous-preuve*, « imbriquée » dans la preuve principale. On peut voir l'idée avec un exemple.

Exemple.

$$\begin{array}{|l} (A \wedge B) \vee (C \wedge D) \\ \hline B \vee D \end{array}$$

Démonstration.

1	$(A \wedge B) \vee (C \wedge D)$	
2	$\quad A \wedge B$	
3	$\quad B$	$\wedge$-élim: 2
4	$\quad B \vee D$	$\vee$-intro: 3
5	$\quad C \wedge D$	
6	$\quad D$	$\wedge$-élim: 5
7	$\quad B \vee D$	$\vee$-intro: 6
8	$B \vee D$	$\vee$-élim: 1, 2–4, 5–7

6.2. **Les règles de la négation.** On commence avec la règle d'élimination de négation, qui réalise la double négation.

(¬-élim)

$$\begin{array}{|ll} & \neg\neg P \\ & \vdots \\ \rhd & P \end{array}$$

Ensuite, la règle d'introduction de la négation, qui réalise la méthode de preuve par contradiction (réduction ad absurdum).

(¬-intro)

$$\begin{array}{|ll} & \quad P \\ & \quad \vdots \\ & \quad \bot \\ \rhd & \neg P \end{array}$$

où $\bot$ signifie la fausseté (toute contradiction). Pour utiliser ce nouveau symbole, il faut donner des règles d'introduction et d'élimination pour lui.

(⊥-intro)

	P
	⋮
	$\neg P$
	⋮
▷	$\bot$

(⊥-élim)

	$\bot$
	⋮
▷	P

Alors ⊥-élim réfléchit l'observation que toute proposition est une conséquence d'une fausseté.

Exemple. Prouver que $A \vee B$, $\neg A$, et $\neg B$ sont inconsistent.

Démonstration.

1	$A \vee B$	
2	$\neg A$	
3	$\neg B$	
4	A	
5	$\bot$	⊥-intro: 4, 2
6	B	
7	$\bot$	⊥-intro: 6, 3
8	$\bot$	∨-élim: 1, 4–5, 6–7

6.3. **Les règles du conditionnel.** On n'a aucun besoin des règles du conditionnel, parce que l'expression $A \to B$ est équivalent logiquement à $\neg A \vee B$ (démontré par un table de vérités). Mais on peut utiliser les règles suivants si vous voulez.

(→-intro)

$$\begin{array}{|l} \quad \begin{array}{|l} A \\ \hline \vdots \\ B \end{array} \\ \rhd \quad A \to B \end{array}$$

(→-élim)

$$\begin{array}{|l} A \to B \\ \vdots \\ A \\ \vdots \\ \rhd \quad B \end{array}$$

6.4. **Exemples.** Ensuite deux exemples qui réunissent la plupart des règles de la logique propositionnelle.

Exemple.

$$\begin{array}{|l} \neg(A \vee B) \\ \hline \neg A \wedge \neg B \end{array}$$

Démonstration.

1	$\neg(A \vee B)$	
2	$\quad A$	
3	$\quad A \vee B$	∨-intro: 2
4	$\quad \bot$	⊥-intro: 1, 3
5	$\neg A$	¬-intro: 2–4
6	$\quad B$	
7	$\quad A \vee B$	∨-intro: 6
8	$\quad \bot$	⊥-intro: 1, 7
9	$\neg B$	¬-intro: 6–8
10	$\neg A \wedge \neg B$	∧-intro: 5, 9

Exemple.

$$\begin{array}{|l} \neg A \wedge \neg B \\ \hline \neg(A \vee B) \end{array}$$

Démonstration.

1	$\neg A \wedge \neg B$	
2	$\neg A$	$\wedge$-élim: 1
3	$\neg B$	$\wedge$-élim: 1
4	$A \vee B$	
5	A	
6	$\bot$	$\bot$-intro: 2, 5
7	B	
8	$\bot$	$\bot$-intro: 3, 7
9	$\bot$	$\vee$-élim: 4-8
10	$\neg(A \vee B)$	$\neg$-intro: 4-9

6.5. **La métathéorie.** Notre système $\mathscr{F}^P$ vise la modélisation de notre raisonnement. Il est *formel* : il ne concerne que la *forme* des inférences, et vous pouvez substituer des propositions et prédicats comme vous voulez. Mais jusqu'ici on n'a pas discuté la *bonté* de ces règles. C'est-à-dire, pourquoi choisir *ces* règles ? On a dit : pour modéliser notre raisonnement, mais il est clair que personne ne raisonne comme ça. On fait des erreurs, on fait des raccourcis. Donc il faut chercher pourquoi choisir ces règles ; c'est-à-dire, pourquoi sont-elles meilleures que des autres règles logiques ?

Ensuite, pour détermine pourquoi choisir les règles de $\mathscr{F}^P$, on considère l'objectif de $\mathscr{F}^P$. Une démonstration formelle vise une série de *conséquences logiques* : on raison d'un énoncé à un autre énoncé, dans une façon que si le premier énoncé est vrai, le second est *nécessairement* vrai. Ou, il est impossible que le premier énoncé soit vrai et le second soit faux. La question, alors, est comment déterminer si tout inférence de $\mathscr{F}^P$ est une conséquence logique.

Ensuite on décrit ces propriétés plus précisément. On commence avec des symboles pour représenter les notions « métalogiques » employées ici. Soit γ un énoncé et soit α un autre. On écrit $\gamma \vdash \alpha$ s'il existe une démonstration dans $\mathscr{F}^P$ de α en prenant γ comme hypothèse. (On doive écrire $\gamma \vdash_{\mathscr{F}^P} \alpha$ pour spécifier qu'on a utilisé le système $\mathscr{F}^P$.)

Par contre, on écrit $\gamma \models \alpha$ quand α est une conséquence logique de γ.

En tous cas on peut remplacer l'énoncé γ avec un ensemble Γ d'énoncés, peut-être infini.

Pour discuter la « vertu » d'un system de logique, on demande les deux questions suivantes :

Correction. Si $\Gamma \vdash \alpha$, est-il que $\Gamma \models \alpha$?

Complétude. Si $\Gamma \models \alpha$, est-il que $\Gamma \vdash \alpha$?

La première question nous demande si tout énoncé démontré dans$\mathscr{F}^P$ d'un autre énoncé, est une conséquence logique de cet autre énoncé. Si oui, on dit que $\mathscr{F}^P$ est *correct*. Un système logique correct n'a aucun défaut. Sinon, on aurait besoin de réparer le système, en enlevant la règle ou les règles qui causent l'erreur, qui dérivent une non conséquence.

En revanche, la deuxième question nous demande si toute conséquence logique d'un énoncé est démontrable dans $\mathscr{F}^P$. Si oui, on dit que $\mathscr{F}^P$ est *complet*. Un système complet n'a aucun besoin d'autres règles d'inférence.

La démonstration du théorème de correction n'est pas difficile. Il faut prouver d'abord que tout axiome logique est valide, c'est-à-dire que pour tout application d'un règle de $\mathscr{F}^P$,

$$\begin{array}{l|l} & \beta \\ \hline \rhd & \alpha \end{array}$$

on a que $\models \alpha$. Ensuite, on prend un ensemble de premises Γ, et par l'induction sur la complexité de théorèmes de Γ, montrer que si $\Gamma \vdash \alpha$, alors $\Gamma \models \alpha$.

On peut exprimer la correction dans une autre façon. On dit $\Gamma \models \alpha$ quand α est une conséquence logique de Γ, quand toute « situation » ou « modèle » qui rend vrai les énoncés de Γ, nécessairement rend vrai α aussi ; ou, il est impossible de trouver un modèle qui rend vrai Γ et rend faux α. On dit ensuite qu'un ensemble Γ est *satisfaisable* quand il existe un modèle de Γ, une situation qui rend vrai les énoncés de Γ. Ensuite, on dit que Γ est *inconsistante* quand $\Gamma \vdash \bot$ et donc que Γ est *consistante* quand Γ n'est pas inconsistante.

Correction (version II). Si Γ est satisfaisable, alors Γ est consistante.

Démonstration (en utilisant correction version I). Supposons que Γ est inconsistante ; on preuve par contrapositive que Γ n'est pas satisfaisable. Alors $\Gamma \vdash \bot$. Par correction version I, $\Gamma \models \bot$. Donc tout modèle qui rend vrai Γ, nécessairement rend vrai $\bot$. Mais aucun modèle ne rend vrai $\bot$. Donc aucun modèle ne rend vrai Γ. Par conséquent Γ n'est pas satisfaisable.

Ensuite : pourquoi est-elle si importante la correction ? Par la règle de $\bot$-élim, on déduit tout énoncé de $\bot$. Donc un système logique inconsistante prouve tout énoncé. Si l'objectif d'un système logique est de prouver les vérités, un système inconsistante est un échec.

Complétude. (Post, 1921) Si $\Gamma \models \alpha$, alors $\Gamma \vdash \alpha$.

7. La syntaxe de la logique du premier ordre

On ajoute les *prédicats* et des *quantificateurs* à notre logique.

Exemples.

- Toute chose qui est un robot est froide.
- Il existe une chose petite.
- Pour tout chat, il existe une personne que l'aime.

Considérons ensuite comment formaliser la logique du premier ordre. On utilise le symbole $S(x)$ pour un prédicat avec une variable libre, x. Cette variable est la variable *liée* par la quantificateur dans ces règles. On peut utiliser des prédicats avec plus de variables libres, mais il faut bien se rappeler que seulement une variable est liée par chaque quantificateur. Les opérations de prédicats sont la quantificateur universelle $\forall$, et la quantificateur existentielle $\exists$.

Pour notre exemple, en choisissant les prédicats R et F, nous pouvons le formaliser par $\forall x R(x) \rightarrow F(x)$. Pour le prochain example, en choisissant le prédicat P, nous pouvons le formaliser par $\exists x P(x)$. Pour le troisième exemple, en choisissant les prédicats C (chat), P (personne) et A (aime), nous pouvons le formaliser par $\forall x \exists y (C(x) \rightarrow A(x,y))$. Pour le troisième exemple, A est un prédicat binaire ; les autres prédicats dans ces exemples sont des prédicats uneaires.

Nous allons traiter les démonstrations un peu plus tard.

8. La sémantique de la logique du premier ordre

Recalling our development of the semantics of propositional logic, suppose there that we'd had a different goal. Suppose we'd wanted to define truth for a *fully interpreted* version of propositional logic. For simplicity, suppose that there are only two propositional sentence symbols A_1 and A_2, and that they have the fixed interpretation of "snow is white" and "grass is green", respectively. We would have defined truth as follows:

(1) (a) A_1 is true iff snow is white.
    (b) A_2 is true iff grass is green.
(2) $(\neg\alpha)$ is true iff α is not true.
(3) $(\alpha \rightarrow \beta)$ is true iff either α is not true or β is true.

However, we wanted to let the interpretations of sentence symbols vary. We thus considered what it would be to be *true on a truth assignment* v:

(1) (a) A_1 is true on v iff $v(A_1) = \mathbf{T}$.

(b) A_1 is true on v iff $v(A_2) = \mathbf{T}$.

(2) $(\neg\alpha)$ is true iff α is not true on v.

(3) $(\alpha \to \beta)$ is true iff either α is not true on v or β is true on v.

Let's now consider how we can do the same for first-order logic.

Let's begin again with a *fully interpreted* first-order language.

Let G and W be predicates meaning "is green" and "is white", respectively.

Let g and s be constants meaning "grass" and "snow", respectively.

We could then attempt to define truth for this language as follows:

Denotation clauses:

(1) (a) s denotes snow.
    (b) g denotes grass.
(2) (a) G denotes the set of green things.
    (b) W denotes the set of white things.

Truth definition attempt:

(1) Pc is true iff the thing denoted by c is a member of the set denoted by P. (Here the only predicates P are G and W, where the sets they denote were specified above.)
(2) $(\neg\alpha)$ is true iff α is not true.
(3) $(\alpha \rightarrow \beta)$ is true iff either α is not true or β is true.
(4) $\forall v_i \alpha$ is true iff _____?

How can we fill this in? The problem is that α, the constituent of the quantified sentence $\forall v_i \alpha$, can be (and usually will be) an open formula, that is, with free variables. But formulas with free variables are neither true nor false.

We could try to patch this by adding to our truth definition clauses like

$\forall v_1 G v_1$ is true iff everything is green.

But what about

$$\forall v_1 (G v_1 \rightarrow \neg W v_1)$$

and

$$\forall v_2 (G v_2 \rightarrow (W v_2 \rightarrow \forall v_3 W v_3))?$$

We'd have to make a similar patch for these and for *infinitely many other* sentences. That is impractical.

Luckily, there is another, practical, solution, due to Tarski. Instead of defining *truth* recursively, we'll define recursively an auxiliary semantic notion called *satisfaction*. The idea behind satisfaction is as follows. Gv_1 may not be true or false on its own because v_1 is a free variable. But it is *true of* particular objects, like grass, and not *true of* others, like snow. We'll say that grass *satisfies* Gv_1, while snow does not.

To handle satisfaction methodically, urgent since we have infinitely many variables, we'll view satisfaction as a relation between wffs and *variable assignments*.

Variable assignments are functions s that map variables to objects. We can think of them as temporary interpretations of variables. Temporarily, the variable is a name for the object to

which it is mapped. For example, if $s(v_1)$ names grass, we'll say that s satisfies Gv_1. The wff $Gv_1 \vee \neg Wv_2$ will be satisfied by s if $s(v_1)$ is green or $s(v_2)$ is not white.

Continuing our informal treatment, we can define satisfaction as follows.

(1) (Again the only predicates P here are G and W, where the sets they denote were specified above.)
   (a) Pc is satisfied by s iff the thing denoted by c is a member of the set denoted by P.
   (b) Pv_i is satisfied by s thing denoted by $s(v_i)$ is a member of the set denoted by P.

(2) $(\neg\alpha)$ is satisfied by s iff α is not satisfied by s.

(3) $(\alpha \to \beta)$ is satisfied by s iff α is not satisfied by s or β is satisfied by s.

(4) $\forall v_i\alpha$ is satisfied by s iff for every s' that differs from s in at most in what it is assigns to v_i, the wff α is satisfied by s'.

This last clause is awkward. We can improve it as follows.

Given a variable assignment s and an object a (such as grass or snow), define:

$$s(v_i|a)(x) = \begin{cases} a & \text{if } x = v_i \\ s(x) & \text{if } x \neq v_i. \end{cases}$$

This modified variable assignment $s(v_i|a)(x)$ replaces what the variable assignment s assigns to v_i with a, while keeping every other assignment of s the same.

Then replace (4) with

(4) $\forall v_i\alpha$ is satisfied by s iff for every object a, α is satisfied by $s(v_i|a)$.

Let's see how this treatment of satisfaction works.

Example. Consider the conditions under which $\forall v_1((\neg Gv_1) \to Wv_1)$ is satisfied by a variable assignment s. It will be iff

$((\neg Gv_1) \to Wv_1)$ is satisfied by $s(v_1|a)$ for all a

iff

either $\neg Gv_1$ is not satisfied by $s(v_1|a)$ for all a or Wv_1 is satisfied by $s(v_1|a)$ for all a

iff

either Gv_1 is satisfied by $s(v_1|a)$ for all a or Wv_1 is satisfied by $s(v_1|a)$ for all a.

Say s assigns v_1 grass and v_2 snow, for instance. Then Gv_1 is not satisfied by $s(v_1|a)$ for all a, since $s(v_1|a)(v_1) = a$ and for one such a, snow, it doesn't belong to the denotation of G.

Similarly, Wv_1 is not satisfied by $s(v_1|a)$ since grass doesn't belong to the denotation of W and $grass$ is one such a.

Using satisfaction, we can define truth for our limited, *interpreted* first-order language:

α is *true* iff α is a sentence and every variable assignment satisfies α.

Thus, to define truth for *interpreted* first-order languages, it was sufficient to use the auxiliary notion of satisfaction by variable assignments.

What if we consider *uninterpreted* first-order languages, as is our goal? We will then need to define truth for *arbitrary interpretations* (e.g. for whatever c, P_1^1 or f_1^2 denote).

For propositional languages, interpretations were just truth assignments to the sentence symbols. For first-order languages, interpretations will be assignments of objects to all non-logical symbols of the language.

$\forall$ will be interpreted by sets, to be called the "domain of quantification".
c_i will be interpreted by objects in the domain of quantification.
P_i^n will be interpreted by sets of n-tuples in the domain of quantification.
f_i^n will be interpreted by n-ary operations on the domain of quantification.

These interpretations will be put together in *structures*. We will then define *truth in a structure*, via *satisfaction by variable assignments in a structure*.

Let's now make our talk of structures more precise. Structures are our means of reinterpreting first-order languages, in particular interpreting the non-logical symbols. Note that I will slide between calling structures "structures" and calling them "models".

We'll follow custom in using the German Fraktur alphabet to write structures in our language, e.g. $\mathfrak{A}$, $\mathfrak{B}$, $\mathfrak{C}$, ... We'll continue to use $\mathscr{L}$ to denote languages.

Definition. $\mathfrak{A}$ is a *structure* for a first-order language $\mathscr{L}$ if $\mathfrak{A}$ is a function with the following features:

(a) The domain of $\mathfrak{A}$ is $\{\forall\} \cup \mathbb{C} \cup \mathbb{P} \cup \mathbb{F}$
(b) $\mathfrak{A}(\forall)$ is a nonempty set (we write $\mathfrak{A}(\forall)$ as $|\mathfrak{A}|$ usually)
(c) $\mathfrak{A}(c) \in |\mathfrak{A}|$ for all $c \in \mathbb{C}$
(d) $\mathfrak{A}(P) \subseteq |\mathfrak{A}|^n$ for all $P \in \mathbb{P}^n$
(e) $\mathfrak{A}(f)$ is a function from $|\mathfrak{A}|^n$ into $|\mathfrak{A}|$ for all $f \in \mathbb{F}^n$.

Thus, a structure tells us what is the domain of quantification of the language (does it range over all dogs or all natural numbers, e.g.). It tells us what all the constants refer to (does c refer to Spot or 5, e.g.). It gives extensions to predicates (e.g. does P apply to all furry things or all odd numbers). It says which functions are expressed by the function symbols (e.g. does f express the "mother of" function or the successor function).

Notice that we don't interpret the predicate $\approx$. It always means identity. We don't interpret any of the logical symbols. We take their meanings to be fixed.

We use, as we said, $|\mathfrak{A}|$ to denote $\mathfrak{A}(\forall)$. Similarly, we write:

$c^{\mathfrak{A}}$ for $\mathfrak{A}(c)$
$P^{\mathfrak{A}}$ for $\mathfrak{A}(P)$
$f^{\mathfrak{A}}$ for $\mathfrak{A}(f)$

These all denote *interpreted* symbols. Don't confuse them with their uninterpreted counterparts.

Let's look at a (contrived) example of a structure.

Let $\mathscr{L}$ be the language of set theory, which has three non-logical symbols, $\forall$, $\in$, $\emptyset$, that we must interpret. Let $\mathfrak{A}$ be the following structure for $\mathscr{L}$.

$|\mathfrak{A}| =$ the set of people
$\in^{\mathfrak{A}} = \{\langle x,y\rangle : x \text{ is the child of } y\}$
$\emptyset^{\mathfrak{A}} =$ my son Teddy

We can then ask what particular sentences in the language express in this structure, e.g.

$\forall v_1(v_1 \notin \emptyset)$ expresses in $\mathfrak{A}$ that "no one is the child of Teddy"
$\forall v_1 \exists v_2(v_1 \in v_2)$ expresses in $\mathfrak{A}$ that "everyone is the child of someone"

Look at the axiom of extensionality,

$$\forall v_1 \forall v_2(\forall v_3(v_3 \in v_1 \leftrightarrow v_3 \in v_2) \rightarrow v1 \approx v_2$$

which says informally in the intended interpretation of set theory that two sets are equal iff they have the same elements. In our structure it expresses "for all people, if they have exactly the same children, then they are identical"!

Let's now return to our project of defining truth for reinterpretable languages. The notion we are after is truth in structures. We'll write

$$\models_{\mathfrak{A}} \alpha$$

for "α is true in structure $\mathfrak{A}$". (Compare with $\models_v \alpha$ for "α is true on truth assignment v.)

Our goal is to define $\models_{\mathfrak{A}} \alpha$. As we indicated before, we'll do so by defining the first the auxiliary notion of satisfaction:

$$\models_{\mathfrak{A}} \alpha[s]$$

for "α is satisfied by variable assignment s in structure $\mathfrak{A}$".

There is one more complication to which we must attend first. Recall that we defined the set of terms for a language inductively. Some times are simple and others are complex. The notion of a variable assignment is fine for simple terms, i.e. those that are variables. They give a temporary denotation to variables. But we must be able to assign these temporary denotations to *terms*, no matter how complex. We'll thus need to be able to *extend* variable assignments to assign values to *all* terms.

Definition. s is a *variable assignment* for $\mathfrak{A}$ if $s : \mathbb{V} \to |\mathfrak{A}|$.

Definition. If s is a variable assignment for $\mathfrak{A}$, then $\overline{s} : \mathbb{T} : |\mathfrak{A}|$ is defined recursively by:

(1) if $v \in \mathbb{V}$, then $\overline{s}(v) = s(v)$
(2) if $c \in \mathbb{C}$, then $\overline{s}(c) = c^{\mathfrak{A}}$
(3) $\overline{s}(ft_1t_2 \ldots t_n) = f^{\mathfrak{A}}(\overline{s}(t_1),\overline{s}(t_2), \ldots ,\overline{s}(t_n))$

Let's see this in action in an example.

Suppose $\mathfrak{A}$ is the "standard" interpretation of the language of arithmetic, so that $|\mathfrak{A}| = \mathbb{N}$, $S^{\mathfrak{A}}$ is the standard successor function, etc. Consider the term

$$+SS0SSSv_1.$$

Suppose the variable assignment s is such that $s(v_1) = 7$. Then

$$\begin{aligned}
\overline{s}(+SS0SSSv_1) &= +^{\mathfrak{A}}(\overline{s}(SS0),\overline{s}(SSSv_1)) \\
&= +^{\mathfrak{A}}(S^{\mathfrak{A}}(\overline{s}(S0)),S^{\mathfrak{A}}(\overline{s}(SSv_1))) \\
&= +^{\mathfrak{A}}(S^{\mathfrak{A}}(S^{\mathfrak{A}}(\overline{s}(0))),S^{\mathfrak{A}}(S^{\mathfrak{A}}(\overline{s}(Sv_1)))) \\
&= +^{\mathfrak{A}}(S^{\mathfrak{A}}(S^{\mathfrak{A}}(0^{\mathfrak{A}})),S^{\mathfrak{A}}(S^{\mathfrak{A}}(S^{\mathfrak{A}}(\overline{s}(v_1))))) \\
&= +^{\mathfrak{A}}(S^{\mathfrak{A}}(S^{\mathfrak{A}}(0)),S^{\mathfrak{A}}(S^{\mathfrak{A}}(S^{\mathfrak{A}}(7)))) \\
&= +^{\mathfrak{A}}(2,10) \\
&= 12
\end{aligned}$$

We can now define satisfaction for atomic wffs. We separate atomic wffs into two forms: $Pt_1t_2 \ldots t_n$ and $t_1 \approx t_n$, because we'll handle the semantics of $\approx$ different from the non-logical predicates. Satisfaction for atomic wffs thus has two clauses.

(a) $\models_{\mathfrak{A}} Pt_1 \ldots t_n\ [s]$ iff $\langle \overline{s}(t_1), \ldots ,\overline{s}(t_n)\rangle \in P^{\mathfrak{A}}$
(b) $\models_{\mathfrak{A}} t_1 \approx t_2\ [s]$ iff $\overline{s}(t_1) = \overline{s}(t_2)$

Since we don't vary the interpretation of $\approx$, it needs a separate treatment.

Using these, we can give the full definition of satisfaction.

Definition. $\models_{\mathfrak{A}} \alpha[s]$ is defined by the following recursion.

(1) (a) $\models_{\mathfrak{A}} Pt_1 \dots t_n\ [s]$ iff $\langle \overline{s}(t_1), \dots, \overline{s}(t_n)\rangle \in P^{\mathfrak{A}}$

(b) $\models_{\mathfrak{A}} t_1 \approx t_2\ [s]$ iff $\overline{s}(t_1) = \overline{s}(t_2)$

(2) $\models_{\mathfrak{A}} (\neg\alpha)\ [s]$ iff $\not\models_{\mathfrak{A}} \alpha\ [s]$

(3) $\models_{\mathfrak{A}} (\alpha \to \beta)\ [s]$ iff either $\not\models_{\mathfrak{A}} \alpha\ [s]$ or $\models_{\mathfrak{A}} \beta\ [s]$

(4) $\models_{\mathfrak{A}} \forall v_i \alpha\ [s]$ iff for all $a \in |\mathfrak{A}|$, $\models_{\mathfrak{A}} \alpha\ [s(v_i|a)]$

Since we have admitted abbreviations for $\wedge$, $\vee$ and $\exists$ into our language and can thus consider wffs with those symbols, it's good to consider how the clauses we gave would determine clauses for these abbreviations.

- $\models_{\mathfrak{A}} (\alpha \wedge \beta)\ [s]$ iff $\models_{\mathfrak{A}} \alpha\ [s]$ and $\models_{\mathfrak{A}} \beta\ [s]$
- $\models_{\mathfrak{A}} (\alpha \vee \beta)\ [s]$ iff either $\models_{\mathfrak{A}} \alpha\ [s]$ or $\models_{\mathfrak{A}} \beta\ [s]$
- $\models_{\mathfrak{A}} \exists v_i \alpha\ [s]$ iff there exists an $a \in |\mathfrak{A}|$ such that $\models_{\mathfrak{A}} \alpha\ [s(v_i|a)]$

Using this definition of satisfaction in a structure, we may define *truth* in a structure:

Definition. A sentence α is *true* in $\mathfrak{A}$ (written $\models_{\mathfrak{A}} \alpha$) iff $\models_{\mathfrak{A}} \alpha\ [s]$ for every variable assignment s for $\mathfrak{A}$.

Let's now consider some examples.

Example. Let $\mathfrak{M}$ have domain $D = \{a,b,c\}$. Let's suppose that our language only contains the binary predicate $Likes$, and that its extension in $\mathfrak{M}$ is

$$Likes^{\mathfrak{M}} = \{(a,a),(a,b),(c,a)\}$$

Consider the wff

$$\exists v_2[Likes(v_1,v_2) \wedge \neg Likes(v_2,v_2)]$$

This wff has a single free variable, x. If our definition is working correctly, it should turn out that a variable assignment s satisfies this wff in $\mathfrak{M}$ iff s assigns a to the variable v_1. That's because a is the only individual who likes someone who doesn't like himself.

Is this how things work out by our definition? Let's say that $s(v_1) = e$, where e is one of a,b,c (the only objects in our domain). Next, we see from the $\exists$ clause that s satisfies our wff iff there is some $d \in D$ such that $s(v_2|d)$ satisfies the wff

$$Likes(v_1,v_2) \wedge \neg Likes(v_2,v_2)$$

But $s(v_2|d)$ satisfies this wff iff it satisfies $Likes(v_1,v_2)$ but does not satisfy $Likes(v_2,v_2)$, by the conjunction and negation clauses. Looking at the atomic case, we see that this is true iff the pair (e,d) is in $Likes^{\mathfrak{M}}$, while the pair (d,d) is not. But this can only happen if $d = b$ and $e = a$. So the only way our original s can satisfy our wff in $\mathfrak{M}$ is if it assigns a to the variable v_1, as we anticipated.

Let's now check that the sentence

$$\exists v_1 \exists v_2 [Likes(v_1,v_2) \wedge \neg Likes(v_2,v_2)]$$

comes out as it should in our model $\mathfrak{M}$. Let s be a variable assignment for $\mathfrak{M}$. According to the definition of satisfaction, s satisfies our sentence in $\mathfrak{M}$ iff there's an object in $\mathfrak{M}$ that we can assign to the variable v_1 so that the resulting variable assignment satisfies

$$\exists v_2 [Likes(v_1,v_2) \wedge \neg Likes(v_2,v_2)]$$

We've seen in our last example that there is such an object, namely a. So the sentence is true in $\mathfrak{M}$; i.e.

$$\models_{\mathfrak{M}} \exists v_1 \exists v_2 [Likes(v_1,v_2) \wedge \neg Likes(v_2,v_2)]$$

Consider next the sentence

$$\forall v_1 \exists v_2 [Likes(v_1,v_2) \wedge \neg Likes(v_2,v_2)].$$

It is true in $\mathfrak{M}$ if every variable assignment for $\mathfrak{M}$ satisfies it. A variable assignment s for $\mathfrak{M}$ will satisfy it iff for every object e in the domain, if we assign e to v_1, the resulting variable assignment s satisfies

$$\exists v_2 [Likes(v_1,v_2) \wedge \neg Likes(v_2,v_2)]$$

But, as we showed earlier, s satisfies this only if s assigns a to v_1. So if it assigns b to v_1, then s doesn't satisfy the wff. Hence, s doesn't satisfy our sentence, and so our sentence isn't true in $\mathfrak{M}$.

Example. Let's show that $\models_{\mathbb{N}} \forall v_1(0 < Sv_1)$, where $\mathbb{N}$ is again the "standard" model of arithmetic. Recall that in the language of arithmetic, 0 is a constant symbol, S is a unary function symbol, and $<$ is a binary predicate symbol.

We need to check that $\models_{\mathbb{N}} \forall v_1(0 < Sv_1)\ [s]$ for every variable assignment s for $\mathbb{N}$. We can apply the definition of satisfaction to check this. First, we handle the quantifier.

$$\models_{\mathbb{N}} \forall v_1(0 < Sv_1)\ [s] \text{ iff for all } n \in |\mathbb{N}|, \models_{\mathbb{N}} (0 < Sv_1)\ [s(v_1)|n].$$

Next, we handle the binary predicate $<$. This holds iff for all $n \in |\mathbb{N}|$,

$$\langle \overline{s(v_1|n)}(0), \overline{s(v_1|n)}(Sv_1) \rangle \in <^{\mathbb{N}}.$$

Next we handle the variable assignment of the terms. Note that for $n \in |\mathbb{N}|$,

$$\overline{s(v_1|n)}(0) = 0^{\mathbb{N}}$$

and

$$\overline{s(v_1|n)}(Sv_1) = S^{\mathbb{N}}(\overline{s(v_1|n)}) = S^{\mathbb{N}}(n).$$

Going back to our check of satisfaction, its previous step holds iff for all $n \in |\mathbb{N}|$,

$$\langle 0^{\mathbb{N}}, S^{\mathbb{N}}(n) \rangle \in <^{\mathbb{N}}.$$

This is now something we can check directly in the model: is the object 0 in our intended model $\mathbb{N}$ less than, in the sense of less than in our intended model, the successor of n, for successor as in our intended model, and for n an object in that model? Yes, it is.

How can we answer this question, though? The answer is that when we specify a model, we are assuming that we know what is what in that model, and can answer questions like this. What we are doing when we ask if a wff is true in a structure, is asking if whether the wff, when its previously uninterpreted terms are given meanings by the structure we've selected, is true. We assume we know what's true and false in the structure we've articulated. The definition of truth in a structure is a way to make sense of the truth of otherwise uninterpreted, meaningless formulas, relative to some structure in which we already understand what is true and not true. It is thus a more robust version of the initial way we described truth of interpreted sentences: "snow is white" iff snow is white. We take truth to be correspondence with "reality", but we want to vary what is "reality". The notion of a structure is a way to do that. We then need to show how otherwise uninterpreted sentences can be given meaning in that created "reality", and then to ask if, for that newly given meaning, what the sentence expresses is true in that created "reality".

Let's now make use of Tarski's definition of truth in a structure. The first notion we can take on is that of *logical consequence*; we'll then turn to *logical truth*.

Intuitively, a sentence is a (first-order) logical consequence of others if it is impossible for the latter to be true and the former false.

Definition. α is a *logical consequence* of Σ (written $\Sigma \models \alpha$) iff for every structure $\mathfrak{A}$ and variable assignment s, if every element of Σ is satisfied by s in $\mathfrak{A}$, then α is satisfied by s in $\mathfrak{A}$. I.e. $\Sigma \models \alpha$ iff for all $\sigma \in \Sigma \models_{\mathfrak{A}} \sigma\ [s]$ implies $\models_{\mathfrak{A}} \alpha\ [s]$.

Thus we are not limited to *sentences* having logical consequences. If we did so limit ourselves, as some logicians do, then we would say that $\Sigma \models \alpha$ iff every $\mathfrak{A}$ in which all the elements of Σ are true is a structure in which α is also true.

If we did so limit ourselves, though, then we wouldn't have e.g.

(1) $\{\forall v_1 P v_1\} \models P v_1$

(2) $\{(P v_1 \rightarrow Q v_2), P v_1\} \models Q v_2$

Do you see why these hold? For (1), suppose we have a model $\mathfrak{A}$ in which a variable assignment s satisfies $\forall v_1 P v_1$. By clause (4) of the definition of satisfaction, for all $a \in |\mathfrak{A}|$ $P v_1$ is satisfied by $s(v_1|a)$. Then whatever element b of $|\mathfrak{A}|$ $s(v_1)$ picks out (i.e. such that $s(v_1) = b$), $s(v_1|b)$ satisfies $P v_1$. But $\overline{s}(v_1) = s(v_1) = s(v_1|b)(v_1) = b$, so $s(v_1)$ satisfies $P v_1$, as we wanted to show.

Definition. α is *valid* (written $\models \alpha$) iff $\emptyset \models \beta$.

As with logical consequence, we can count open formulas as validities. Thus $\models (Qv_2 \to Qv_2)$ and $\models \forall v_2(Qv_2 \to Qv_2)$ are both validities.

Using these definition, we can chase down the following fact: $\models \alpha$ iff for all structures $\mathfrak{A}$ and variable assignments s, $\models_{\mathfrak{A}} \alpha\ [s]$.

When we limit our discussion to sentences, there are some special terms to use.

If σ is a sentence and $\models \mathfrak{A}\sigma$, then we say that $\mathfrak{A}$ is a *model* of σ.

If Σ is a set of sentences and for all $\sigma \in \Sigma$, $\models \mathfrak{A}\sigma$, then we say that $\mathfrak{A}$ is a *model* of Σ.

Thus $\mathfrak{A}$ is a model of σ if it makes σ true, and a model of Σ if it makes all sentences in Σ true.

If σ is a valid sentence (i.e. $\models \sigma$), then we say that σ is *logically true*.

Thus σ is a logical truth iff every structure for the language is a model of σ, i.e. it is true in every model of the language.

In other words, there is no way to reinterpret the non-logical symbols and make the sentence false. They are "invariant" under non-logical reinterpretation. They are, in this sense, the "core" or "essential" truths of the language.

Example. $\forall v_1 Pv_1 \models Pv_2$ (slightly different from last case). Let's show the contrapositive, i.e., for $\mathfrak{A}$ and s an arbitrary structure and variable assignment, respectively, suppose $\not\models_{\mathfrak{A}} Pv_2\ [s]$. Then $\overline{s}(v_2) \notin P^{\mathfrak{A}}$. Then $\not\models_{\mathfrak{A}} Pv_1\ [s(v_1|\overline{s}(v_2)]$. So $\not\models \forall v_1 Qv_1\ [s]$.

Example. $Pv_1 \not\models \forall v_1 Qv_1$. We need to find some $\mathfrak{A}$, s such that $\models_{\mathfrak{A}} Pv_1\ [s]$ but $\not\models_{\mathfrak{A}} \forall v_1 Pv_1\ [s]$. Let $\mathfrak{A}$ be a structure with $|\mathfrak{A}| = \{0,1\}$, $P^{\mathfrak{A}} = \{0\}$. Let $s(v_1) = 0$. Then $\models_{\mathfrak{A}} Pv_1\ [s]$ (since $s(v_1) = 0 \in P^{\mathfrak{A}}$). However, $\not\models_{\mathfrak{A}} \forall v_1 Pv_1\ [s]$, because $\not\models_{\mathfrak{A}} Pv_1\ [(s(v_1|1)]$.

9. La syntaxe de la logique du premier ordre : démonstrations

L'idée est encore de donner des règles de *l'introduction* et de *l'élimination* des opérations logiques. Ces opérations logique sont composées de deux types : les opérations *propositionnelles* et les opérations des *prédicats* (y compris les quantificateurs). La première partie est la *logique propositionnelle* et la deuxième partie, la *logique des prédicats*. Ensemble il constituent la logique du premier ordre. Ensuite on introduit les règles des quantificateurs.

		$\forall x S(x)$
(∀-élim)		$\vdots$
	$\rhd$	$S(c)$

(∃-intro)

$$\begin{array}{|ll} & S(c) \\ & \vdots \\ \rhd & \exists x S(x) \end{array}$$

où $S(x)$ est une formule, x est une variable, c est une constante, et $S(c)$ est le résultat du remplacement des instances libres de x dans $S(x)$ avec c.

Exemple:

$$\begin{array}{|l} \forall x(Cube(x) \vee Tet(x) \\ \neg Cube(a) \\ \hline \exists x Tet(x) \end{array}$$

Démonstration.

1	$\forall x(Cube(x) \vee Tet(x)$	
2	$\neg Cube(a)$	
3	$Cube(a) \vee Tet(a)$	∀-élim: 1
4	$\quad Cube(a)$	
5	$\quad \bot$	⊥–intro: 2, 4
6	$\quad \exists x Tet(x)$	⊥-élim: 5
7	$\quad Tet(a)$	
8	$\quad \exists x Tet(x)$	∃-intro: 7
9	$\exists x Tet(x)$	∨-élim: 3, 4–6, 7–8

Ensuite, l'élimination de la quantificateur existentielle.

(∃-élim)

$$\begin{array}{|ll} & \exists x S(x) \\ & \vdots \\ & \left|\begin{array}{l} \boxed{c} S(c) \\ \hline \vdots \\ Q \end{array}\right. \\ \rhd & Q \end{array}$$

On note d'abord que, pour ce règle, c n'apparaisse jamais à l'extérieur de la sous-démonstration. C'est une nouvelle constante qui satisfait $S(x)$ « temporairement », pendant la sous-démonstration. Ensuite, la notation $\boxed{c}S(c)$ signifie que, pour cette sous-démonstration, la constante c désigne un objet quelconque qui satisfait $S(x)$. Parce que $\exists x S(x)$ s'obtient, on sait qu'on tel objet existe. Dans cette sous-démonstration, on agit comme cet objet est nommé c. Enfin, la règle dit que, si on peut déduire Q quand l'objet satisfaisant $S(x)$ est nommé, alors on peut déduire Q de $\exists x S(x)$.

Exemple:

$$\begin{array}{|l} \forall x(Cube(x) \to Petit(x)) \\ \exists x Cube(x) \\ \hline \exists x Petit(x) \end{array}$$

Démonstration.

1	$\forall x(Cube(x) \to Petit(x))$	
2	$\exists x Cube(x)$	
3	$\quad \boxed{c} Cube(c)$	
4	$\quad Cube(c) \to Petit(c)$	∀-élim: 1
5	$\quad Petit(c)$	→-élim: 3, 4
6	$\quad \exists x Petit(x)$	∃-intro: 5
7	$\exists x Petit(x)$	∃-élim: 2, 3–6

Enfin, on peut terminer la présentation de notre système formel avec la règle de l'introduction de la quantificateur universelle.

(∀-intro)

$$\begin{array}{|ll} & \boxed{c} \\ & \vdots \\ & S(c) \\ \rhd & \forall x S(x) \end{array}$$

où, encore, c n'apparaisse jamais à l'extérieur de la sous-démonstration.

Exemple:

$$\begin{array}{|l} \forall x(Cube(x) \to Petit(x)) \\ \forall x Cube(x) \\ \hline \forall x Petit(x) \end{array}$$

Démonstration.

1	$\forall x(Cube(x) \to Petit(x))$	
2	$\forall x Cube(x)$	
3	$\boxed{c}$	
4	$Cube(c)$	∀-élim: 2
5	$Cube(c) \to Petit(c)$	∀-élim: 1
6	$Petit(c)$	→-élim: 4, 5
7	$\forall x Petit(x)$	∀-intro: 3–6

Les deux exemples suivants montrent une grande variété de règles.

Exemple.

$$\begin{array}{|l} \neg\exists x S(x) \\ \hline \forall x \neg S(x) \end{array}$$

Démonstration.

1	$\neg\exists x S(x)$	
2	$\boxed{c}$	
3	$S(c)$	
4	$\exists x S(x)$	$\exists$-intro: 3
5	$\bot$	$\bot$-intro: 1, 4
6	$\neg S(c)$	$\neg$-intro: 2, 3–5
7	$\forall x \neg S(x)$	$\forall$-intro: 2–6

Exemple.

$\forall x \neg S(x)$	
$\neg\exists x S(x)$	

Démonstration.

1	$\forall x \neg S(x)$	
2	$\exists x S(x)$	
3	$\boxed{c} S(c)$	
4	$\neg S(c)$	$\forall$-élim: 1
5	$\bot$	$\bot$-intro: 3, 4
6	$\bot$	$\exists$-élim: 2, 3–5
7	$\neg\exists x S(x)$	$\neg$-intro: 2–6

10. La métathéorie de la logique du premier ordre

Notre système $\mathcal{F}$ vise la modélisation de notre raisonnement. Il est *formel* : il ne concerne que la *forme* des inférences, et vous pouvez substituer des propositions et prédicats comme vous voulez. Mais jusqu'ici on n'a pas discuté la *bonté* de ces règles. C'est-à-dire, pourquoi choisir *ces* règles ? On a dit : pour modéliser notre raisonnement, mais il est clair que personne ne raisonne comme ça. On fait des erreurs, on fait des raccourcis. Donc il faut chercher pourquoi choisir ces règles ; c'est-à-dire, pourquoi sont-elles meilleures que des autres règles logiques ?

D'abord, on note que on peut déterminer facilement si une inférence est une instance d'une telle règle. La forme des règles est complètement spécifiée et finie. Donc il n'est pas difficile d'envisager une machine pour identifier les applications des règles. On dit, informellement,

que ces règles sont mécaniques. Un système mécanique d'inférence logique était envisagé par Leibniz. Leibniz a cherché un *characteristica universalis*, un langage clair et logique pour l'expression de toute pensée rationnelle, et un *calcul ratiocinator*, une codification mécanique de tout bon raisonnement. Il croit qu'ils amélioreraient à la fois les condition *épistémique*, *sociales* et *matérielles* du monde. En particulier qu'il aideraient la paix et la justice, car les disputes à la vérité et la distribution des biens pourraient résoudre mécaniquement. Cette solution mettrait la paix et la justice à la disposition de tout le monde, même les penseurs faibles, car Leibniz pense que tout le monde ait la même capacité de calculer mécaniquement.

Pour dire plus précisément que notre système est bien mécanique, il faudrait avoir une notion plus précise d'être « mécanique ». Une telle analyse est donné par Alan Turing.

Ensuite, pour détermine pourquoi choisir les règles de $\mathscr{F}$, on considère l'objectif de $\mathscr{F}$. Une démonstration formelle vise une série de *conséquences logiques* : on raison d'un énoncé à un autre énoncé, dans une façon que si le premier énoncé est vrai, le second est *nécessairement* vrai. Ou, il est impossible que le premier énoncé soit vrai et le second soit faux. La question, alors, est comment déterminer si tout inférence de $\mathscr{F}$ est une conséquence logique.

Rappelons que la notion de conséquence logique implique une modalité : il est *impossible* que le premier énoncé soit vrai et le second soit faux. Comment déterminer cette impossibilité ? Pour démontrer qu'un énoncé n'est pas une conséquence logique d'un autre, il suffit de trouver un contrexemple, une « situation » que démontre qu'il est possible que le premier énoncé soit vrai et le second soit faux. Alors le second n'est pas forcement vrai quand le premier est vrai.

Un exemple : on se demande si « La neige est blanche et l'herbe est verte » est une conséquence logique de « La neige est blanche ». Normalement, oui, on envisage le monde comme ça. Mais est-ce que c'est *nécessaire* ? Peut-on imaginer que l'herbe soit brune quand la neige est blanche ? Tout à fait. Dans cette « situation » le premier énoncé soit vrai et le second soit faux. Donc la réponse est non.

Par contre : on se demande si « La neige est blanche ou l'herbe est verte » est une conséquence logique de « La neige est blanche ». Notre situation en hausse ne marche pas ici : même si l'herbe soit brune, il reste que la neige est blanche ou l'herbe est verte. Pourquoi ? À cause de la signification de la disjonction, qui dit qu'une disjonction est vraie si l'une ou l'autre partie de cette disjonction est vraie. Donc toute situation dans laquelle la neige est blanche est une situation où la neige est blanche ou l'herbe est verte. Donc la réponse est oui.

Un autre exemple : on se demande si « Alain n'est pas un robot » est une conséquence logique de « Toute chose qui est un robot est froide » et « Alain n'est pas froid ». Évidemment oui, mais comment prouver ? Pour aider notre capacité de trouver des contrexemples, il est utile de noter que les impossibilités pertinents sont les *impossibilités logiques* : c'est-à-dire, les impossibilités qui émanent des significations logiques. Pour identifier les significations logiques pertinentes, il est utile de trouver la forme logique de ces énoncés. En choisissant

le prédicats R et F, et la constante a, on trouve : $\neg R(a)$, $\forall x R(x) \to F(x)$, et $\neg F(a)$. Et on peut voir que par la signification de $\forall$, on a $R(a) \to F(a)$, et par la signification de $\to$, que si $\neg F(a)$, alors $\neg R(a)$ (autrement on a une contradiction). Cette méthode de remplacement montre qu'on a une conséquence logique.

Un autre exemple : on se demande si « il existe une cube » et « il existe une chose petite » implique logiquement que « il existe une petite cube ». Pour répondre, on utilise encore la méthode de remplacement : $\exists x C(x)$, $\exists x P(x)$, $\exists x(C(x) \wedge P(x))$. Peut-on décrire une contrexemple ? Par la signification de $\exists$, on a un objet vérifiant le prédicat C, et un objet vérifiant le prédicat P. Mais on n'est pas assuré que ces objets sont identiques. Cependant la conclusion, $\exists x(C(x) \wedge P(x))$, exige que les objets sont identiques. Donc la conclusion n'est pas une conséquence logique des premisses.

Ensuite on décrit ces propriétés plus précisément. On commence avec des symboles pour représenter les notions « métalogiques » employées ici. Soit γ un énoncé et soit α un autre. On écrit $\gamma \vdash \alpha$ s'il existe une démonstration dans $\mathscr{F}$ de α en prenant γ comme hypothèse. (On doive écrire $\gamma \vdash_{\mathscr{F}} \alpha$ pour spécifier qu'on a utilisé le système $\mathscr{F}$.)

Par contre, on écrit $\gamma \models \alpha$ quand α est une conséquence logique de γ.

En tous cas on peut remplacer l'énoncé γ avec un ensemble Γ d'énoncés, peut-être infini.

Pour discuter la « vertu » d'un system de logique, on demande les deux questions suivantes :

Correction. Si $\Gamma \vdash \alpha$, est-il que $\Gamma \models \alpha$?

Complétude. Si $\Gamma \models \alpha$, est-il que $\Gamma \vdash \alpha$?

La première question nous demande si tout énoncé démontré dans $\mathscr{F}$ d'un autre énoncé, est une conséquence logique de cet autre énoncé. Si oui, on dit que $\mathscr{F}$ est *correct*. Un système logique correct n'a aucun défaut. Sinon, on aurait besoin de réparer le système, en enlevant la règle ou les règles qui causent l'erreur, qui dérivent une non conséquence.

En revanche, la deuxième question nous demande si toute conséquence logique d'un énoncé est démontrable dans $\mathscr{F}$. Si oui, on dit que $\mathscr{F}$ est *complet*. Un système complet n'a aucun besoin d'autres règles d'inférence.

La démonstration du théorème de correction n'est pas difficile. Il faut prouver d'abord que tout axiome logique est valide, c'est-à-dire que pour tout application d'un règle de $\mathscr{F}$,

$$\rhd \;\begin{array}{|l} \beta \\ \hline \alpha \end{array}$$

on a que $\models \alpha$. Ensuite, on prend un ensemble de premises Γ, et par l'induction sur la complexité de théorèmes de Γ, montrer que si $\Gamma \vdash \alpha$, alors $\Gamma \models \alpha$.

On peut exprimer la correction dans une autre façon. On dit $\Gamma \models \alpha$ quand α est une conséquence logique de Γ, quand toute « situation » ou « modèle » qui rend vrai les énoncés de Γ, nécessairement rend vrai α aussi ; ou, il est impossible de trouver un modèle qui rend vrai Γ et rend faux α. On dit ensuite qu'un ensemble Γ est *satisfaisable* quand il existe un modèle de Γ, une situation qui rend vrai les énoncés de Γ. Ensuite, on dit que Γ est *inconsistante* quand $\Gamma \vdash \bot$ et donc que Γ est *consistante* quand Γ n'est pas inconsistante.

Correction (version II). Si Γ est satisfaisable, alors Γ est consistante.

Démonstration (en utilisant correction version I). Supposons que Γ est inconsistante ; on preuve par contrapositive que Γ n'est pas satisfaisable. Alors $\Gamma \vdash \bot$. Par correction version I, $\Gamma \models \bot$. Donc tout modèle qui rend vrai Γ, nécessairement rend vrai $\bot$. Mais aucun modèle ne rend vrai $\bot$. Donc aucun modèle ne rend vrai Γ. Par conséquent Γ n'est pas satisfaisable.

Ensuite : pourquoi est-elle si importante la correction ? Par la règle de $\bot$-élim, on déduit tout énoncé de $\bot$. Donc un système logique inconsistante prouve tout énoncé. Si l'objectif d'un système logique est de prouver les vérités, un système inconsistante est un échec.

Complétude. (Gödel, 1930) Si $\Gamma \models \alpha$, alors $\Gamma \vdash \alpha$.