Kit de survie - Logique

François Schwarzentruber

Préparation à l'option informatique de l'agrégation de mathématiques
ÉNS Rennes

Ces notes de cours survolent le programme en logique de l'option informatique de l'agrégation de mathématiques. Ce document a été débuté en 2016 à l'aide des élèves agrégatifs à l'ÉNS Rennes que je remercie. Elles sont volontairement laconiques mais illustrées. Merci à Hubert Comon et Jean Goubault-Larrecq pour leurs discussions au sujet de l'acronyme LJ.

Table des matières

Première partie

Logique propositionnelle

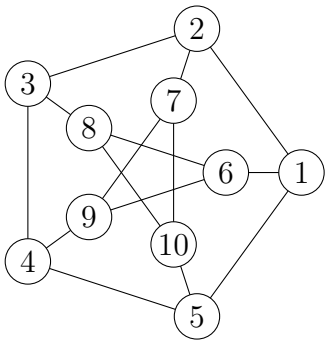
Chapitre 1

Logique propositionnelle : syntaxe et sémantique

Points du programme de l'agrégation
Calcul propositionnel : syntaxe et sémantique. Tables de vérité. Tautologies.

1.1 Motivation : problème de coloration

On souhaite trouver une 3-coloration d'un graphe G non orienté. L'idée est d'exprimer les contraintes de coloriage avec une formule de la logique propositionnelle.



1.2 Un modèle : une valuation

Soit AP un ensemble dénombrable de **propositions atomiques**. Un modèle de la logique propositionnelle est une valuation. Formellement :

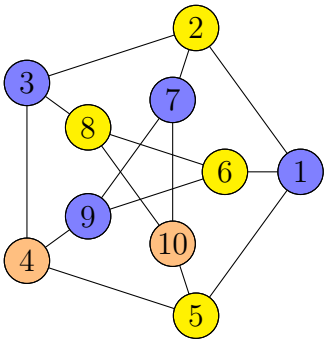
Définition 1 (valuation, [Dup15], p. 84)
Une **valuation** V est une fonction de AP dans $\{0, 1\}$.

Exemple 2 *Pour toute sommet s et toute couleur c, on introduit la proposition atomique*

$$p_{s,c}$$

qui signifie intuitivement que

la couleur du sommet s est c.



Une valuation représente une 3-coloriation. La 3-coloration donnée sur le graphe est représentée par la valuation V définie par :

- $V(p_{1,\bullet}) = V(p_{2,\bullet}) = \ldots = 1$;
- $V(p_{1,\bullet}) = V(p_{1,\bullet}) = \ldots = 0$.

1.3 Langage

1.3.1 Syntaxe

Définition 3 (syntaxe du langage de la logique propositionnelle)
Le **langage de la logique propositionnelle** $\mathcal{L}$ est défini par la grammaire suivante :

$$\varphi \ ::= \ \bot \ \mid \ p \ \mid \ \neg\varphi \ \mid \ (\varphi \vee \varphi) \ \mid \ (\varphi \wedge \varphi)$$

où p désigne une proposition atomique dans AP.

On introduit les abréviations suivantes :
— $(\varphi \to \psi)$ pour $(\neg\varphi \vee \psi)$;
— $(\varphi \leftrightarrow \psi)$ pour $((\varphi \to \psi) \wedge (\psi \to \varphi))$.
On omet les parenthèses quand elles sont évidentes.

Exemple 4 *On écrit $p \vee q \vee r$ au lieu de $(p \vee (q \vee r))$ ([Dup15], p. 83).*

1.3.2 Sémantique

Définition 5 (conditions de vérité, [Dup15], p. 87)
$V \models \varphi$ est défini par induction structurelle sur φ :
— $V \not\models \bot$;
— $V \models p$ si $V(p) = 1$;
— $V \models \neg\varphi$ si $V \not\models \varphi$;
— $V \models (\varphi \vee \psi)$ si $V \models \varphi$ ou $V \models \psi$;
— $V \models (\varphi \wedge \psi)$ si $V \models \varphi$ et $V \models \psi$.

Définition 6 (ensemble des modèles d'une formule)
On note $[\![\varphi]\!]$ l'ensemble $\{V \mid V \models \varphi\}$.

Tables de vérité

Dans une table de vérité, chaque ligne correspond à une valuation V, et on inscrit 1 dans la colonne pour φ si $V \models \varphi$ et 0 si $V \not\models \varphi$.

p	q	$\neg q$	$p \vee \neg q$
0	0	1	1
0	1	0	0
1	0	1	1
1	1	0	1

Définition 7 (satisfiable)
φ est **satisfiable**
s'il existe une valuation V telle que $V \models \varphi$.

		φ
		.
		1
		.
		.

Définition 8 (valide)
φ est **valide** (ou est une **tautologie**)
si pour toute valuation V, on a $V \models \varphi$.

		φ
		1
		1
		⋮
		1

1.4 Théories

Définition 9 (théorie, [Dup15], p. 109)
Une **théorie** T est un ensemble de formules.

Définition 10 (notation $V \models T$)
On écrit $V \models T$ pour dire que pour toute formule ψ de T, on a $V \models \psi$.

Définition 11 (conséquence sémantique)
φ est **conséquence sémantique** d'une théorie T, noté $T \models \varphi$,
si pour toute valuation V, on a $V \models T$ implique $V \models \varphi$.

1.5 Systèmes de connecteurs complets

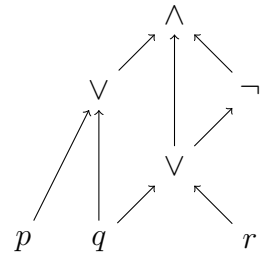
Définition 12 (formules équivalentes)
φ et ψ sont **équivalentes** si pour toute valuation V, ($V \models \varphi$ si et seulement $V \models \psi$).

Définition 13 (système de connecteurs complet, [Dup15], p. 175)
Un système de connecteurs est **complet** si toute formule est équivalente à une formule qui ne contient que ces connecteurs là.

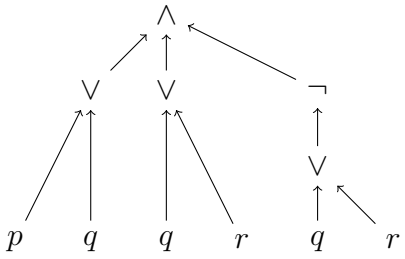
Proposition 14 *Les systèmes de connecteurs* $\{\neg, \wedge\}$, $\{\neg, \rightarrow\}$, $\{\neg, \vee\}$, $\{nand\}$ *sont complets.*

1.6 Circuits booléens

Circuit = graphe acyclique



Formule = arbre



Définition 15 (circuit booléen)
Un **circuit booléen** est un graphe acyclique avec une seule sortie. Les noeuds qui ne sont pas des entrées sont des **portes** et, ou, non.

1.7 Problèmes de décision

1.7.1 Problèmes d'évaluation

Définition 16 (évaluation de formules)

— entrée : une valuation V, une formule φ ;
— sortie : oui si $V \models \varphi$; non, sinon.

Théorème 17 *[Bus87] Le problème d'évaluation de formules est dans LOGSPACE.*

Définition 18 (évaluation de circuits)

— entrée : une valuation V, un circuit C ;
— sortie : oui si $V \models C$; non, sinon.

Théorème 19 *[Sip97] Le problème d'évaluation de circuits est P-complet.*

1.7.2 SAT et VALIDE

Définition 20 (SAT)

— entrée : une formule φ ;
— sortie : oui si φ est satisfiable ; non, sinon.

Théorème 21 *SAT est NP-complet.*

Définition 22 (VALIDE)

— entrée : une formule φ ;
— sortie : oui si φ est valide ; non, sinon.

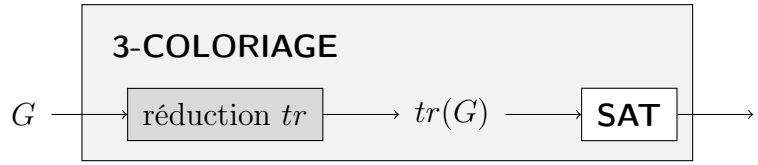
Théorème 23 *VALIDE est coNP-complet.*

Définition 24 (CIRCUIT SAT)

— entrée : un circuit C ;
— sortie : oui si C est satisfiable ; non, sinon.

Théorème 25 *CIRCUIT SAT est NP-complet.*

Application pratique. On réduit le problème de 3-coloriage au problème SAT :



Chapitre 2

Formes normales

Points du programme de l'agrégation
Formes normales, forme clausale.

2.1 Formes normales négatives

Définition 26 (forme normale négative)
Une **forme normale négative** est une formule générée par la grammaire

$$\varphi ::= \bot \mid \top \mid p \mid \neg p \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi)$$

où p est une proposition atomique dans AP.

Proposition 27 *Toute formule admet une forme normale négative équivalente.*

Idée de la démonstration.

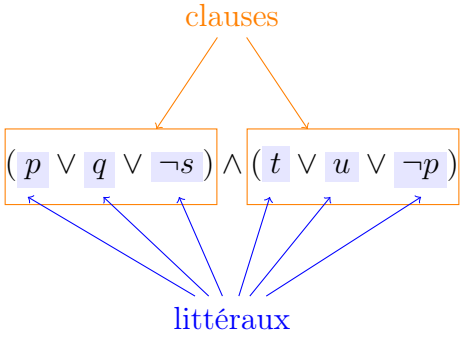
— $tr(\bot) = \bot$;
— $tr(p) = p$;
— $tr(\varphi \vee \psi) = tr(\varphi) \vee tr(\psi)$;
— $tr(\varphi \wedge \psi) = tr(\varphi) \wedge tr(\psi)$;
— $tr(\neg p) = \neg p$;
— $tr(\neg(\varphi \vee \psi)) = tr(\neg\varphi) \wedge tr(\neg\psi)$;
— $tr(\neg(\varphi \wedge \psi)) = tr(\neg\varphi) \vee tr(\neg\psi)$.

■

2.2 Formes normales conjonctives

Définition 28 (littéral)
Un **littéral** est une proposition atomique ou la négation d'une proposition atomique.

Définition 29 (forme normale conjonctive)
Une **forme normale conjonctive** (FNC) est une formule de la forme $\bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} \ell_{i,j}$ où les $\ell_{i,j}$ sont des littéraux.



Théorème 30 *Toute formule est équivalente à une FNC*

...potentiellement exponentiellement plus longue.

Idée de la démonstration.

1. Mettre sous forme normale négative ;
2. Appliquer les règles de distributivité.

■

Exemple 31

$$\begin{aligned}(p \wedge q) \vee (r \wedge s) &\equiv ((p \wedge q) \vee r) \wedge ((p \wedge q) \vee s) \\ &\equiv (p \vee r) \wedge (q \vee r) \wedge (p \vee s) \wedge (q \vee s).\end{aligned}$$

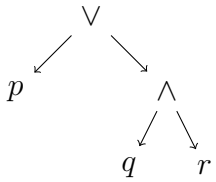
2.3 Transformation de Tseitin ([BA12], p. 91)

Définition 32 (équisatisfiable)
φ et ψ est **équisatisfiable** ssi (φ satisfiable ssi ψ satisfiable).

Théorème 33 *Toute formule φ est équisatisfiable à une FNC $tr(\varphi)$* *...de taille $O(\varphi)$.*

Idée de la démonstration.

1. Introduire une var. prop. α_ψ par sous-formules ψ de φ qui signifie 'ψ est vraie'.
2. Coder les conditions de vérité par des clauses.



$p \vee (q \wedge r)$ est satisfiable ssi $\alpha_{(p\vee(q\wedge r))} \wedge$
$(\alpha_{(p\vee(q\wedge r))} \rightarrow (\alpha_p \vee \alpha_{(q\wedge r)})) \wedge$
$(\alpha_p \rightarrow \alpha_{(p\vee(q\wedge r))}) \wedge (\alpha_{(q\wedge r)} \rightarrow \alpha_{(p\vee(q\wedge r))}) \wedge$
$(\alpha_{(q\wedge r)} \rightarrow \alpha_q) \wedge$
$(\alpha_{(q\wedge r)} \rightarrow \alpha_r) \wedge$
$((\alpha_q \wedge \alpha_r) \rightarrow \alpha_{(q\wedge r)})$
satisfiable.

■

2.4 Formes normales disjonctives

Définition 34 (forme normale disjonctive)
Une **forme normale disjonctive** est une formule de la forme $\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} \ell_{i,j}$ où les $\ell_{i,j}$ sont des littéraux.

2.5 Algorithme de Davis, Putnam, Logemann et Loveland

— Entrée : une forme normale conjonctive φ
— Sortie : une valuation ν telle que $\nu \models \varphi$; ou UNSAT si φ n'est pas satisfiable.

Principe de l'algorithme DPLL : construire itérativement une valuation ν partielle
— Optimisation :
  — Propagation unitaire
  — Elimination des littéraux purs
— Choisir une valeur de vérité à une variable non assignée selon une heuristique
— Backtracking.

Définition 35 (clause unitaire)
Une clause $\bigvee_j \ell_j$ est ν-unitaire s'il existe un j_0 avec :
— pour tout $j \neq j_0$, $\nu \models \neg\ell_j$;
— ℓ_{j_0} non ν-assigné.

$$\not{p} \vee \not{q} \vee \not{\neg s} \vee \ell.$$

Définition 36 (propagation unitaire)
Si $\bigvee_j \ell_j$ est unitaire et ℓ_{j_0} non assigné, on met ℓ_{j_0} à vrai.

Définition 37 (littéral pur)
Un littéral est pur s'il apparait positivement dans toutes les clauses.

$$(p \vee q \vee \ell) \wedge (r \vee s \vee \neg u \vee \ell) \wedge (s \vee \neg t).$$

Définition 38 (élimination d'un littéral pur)
Si ℓ est pur et non assigné, on le met à vrai.

Backjumping et apprentissage de clauses
En cas de contradiction, trouver une nouvelle clause :
— qui explique la contradiction ;
— qui ne contient qu'un seul littéral du dernier niveau des choix
— qui permet d'oublier les choix non pertinents et de revenir tôt dans les choix.

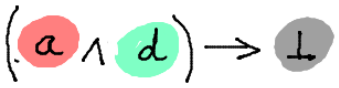
`http://people.irisa.fr/Francois.Schwarzentruber/dpll_demo/`

2.6 Fragments syntaxiques basés sur les FNC

2.6.1 Formules de Horn ([DPV16], p. 157)

Définition 39 (clause de Horn)
Une **clause de Horn** est une clause de la forme $(p_1 \wedge \cdots \wedge p_n) \to p$ ou $(p_1 \wedge \cdots \wedge p_n) \to \bot$.

Définition 40 (formule de Horn)
Une **formule de Horn** est une conjonction de clauses de Horn.



Définition 41 (problème HORN-SAT)
Le **problème HORN-SAT** est le problème décision :
- Entrée : une formule de Horn ;
- Sortie : oui si elle est satisfiable ; non sinon.

Théorème 42 *Le problème HORN-SAT est dans P.*

Idée de la démonstration.

fonction satHorn(φ)
  $V :=$ valuation où toutes les propositions atomiques sont fausses
  tant que il existe une clause $\Gamma \to p$ avec $V \not\models \Gamma \to p$ **do**
    $V := V[p := \top]$
  si une clause $\Gamma \to \bot$ avec $V \models \Gamma$ **alors**
    retourner insatisfiable
  sinon
    retourner V

Lemme 43 *satHorn(φ) retourne une valuation V ssi φ est satisfiable.*

Idée de la démonstration.
$[\Rightarrow]$ La valuation V satisfait φ.
$[\Leftarrow]$ Soit V' telle que $V' \models \varphi$. La propriété '$V \subseteq V'$ est un invariant. ■
■

Théorème 44 *[Sip97] HORN-SAT est P-complet.*

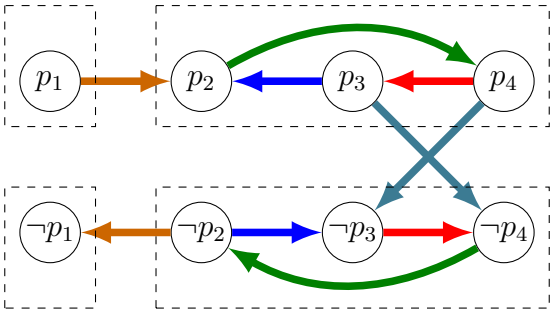
Applications : calcul de premier, suivant en analyse syntaxique LL(1), des non-terminaux productifs d'une grammaire algébrique, des non-terminaux qui engendre ϵ [LS15].

2.6.2 2-formes normales

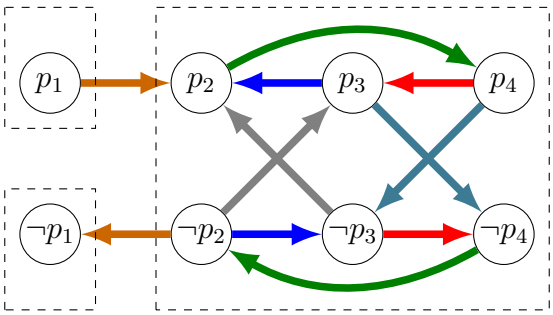
Définition 45 (problème 2SAT)
Le **problème 2SAT** est le problème décision :
— Entrée : une FNC avec 2 littéraux par clause ;
— Sortie : oui si elle est satisfiable ; non sinon.

Définition 46 (graphe d'implication)
Soit φ une 2-CNF. Le **graphe d'implication** G_φ est le graphe où :
— les sommets sont les propositions atomiques de φ et leurs négations ;
— il y a un arc (α, β) si $\alpha \to \beta$ est équivalente à une clause de φ.

Exemple 47 $(p_3 \vee \neg p_4) \wedge (p_2 \vee \neg p_3) \wedge (p_4 \vee \neg p_2) \wedge (\neg p_1 \vee p_2) \wedge (\neg p_4 \vee \neg p_3)$ *est satisfiable.*



Exemple 48
$(p_3 \vee \neg p_4) \wedge (p_2 \vee \neg p_3) \wedge (p_4 \vee \neg p_2) \wedge (\neg p_1 \vee p_2) \wedge (\neg p_4 \vee \neg p_3) \wedge (p_3 \vee p_2)$ *n'est pas satisfiable.*



Théorème 49 *[Sip97] Le problème 2-SAT est dans NLOGSPACE-complet.*

Théorème 50 *Il existe un algorithme en temps linéaire pour décider le problème 2-SAT.*

IDÉE DE LA DÉMONSTRATION.

fonction sat2sat(φ)
  Construire le graphe d'implication G_φ
  Calculer les composantes fortement connexes de G_φ
  si il existe p tel que p et $\neg p$ sont dans la même composante fortement connexe **alors**
    retourner insatisfiable
  sinon
    retourner satisfiable

Lemme 51 *sat2sat(φ) retourne* satisfiable *ssi φ est satisfiable.*

Idée de la démonstration.
($\Leftarrow$) Soit V une valuation telle que $V \models \varphi$. Par l'absurde, supposons qu'il existe p et $\neg p$ dans la même composante fortement connexe (cfc). Sans perte de généralité, supposons que $V \models p$. Comme il y a un chemin de $p = \ell_0, \ell_1, \ldots, \ell_n = \neg p$ dans G_φ, et comme $V \models \ell_i \to \ell_{i+1}$, on montre par récurrence sur i que $V \models \ell_i$ pour tout i. Donc $V \models \neg p$. Contradiction.

($\Rightarrow$) Supposons que sat2sat(φ) retourne satisfiable. Considérons l'algorithme suivant :

```
fonction construireValuation(G)
    si le graphe G est vide alors
        retourner valuation partielle vide
    sinon
        Soit C une cfc finale de G
        V := construireValuation(G − (C ∪ {¬ℓ | ℓ ∈ C}))
        Mettre tous les littéraux de C à vrai dans V
        retourner V
```

On considère la valuation $V :=$ construireValuation(G_φ).

Fait 1 *L'opération 'Mettre tous les littéraux de C à vrai dans V' est bien définie.*

Idée de la démonstration.
Comme p et $\neg p$ ne sont pas dans la même cfc, on donne une valeur unique à p. ■

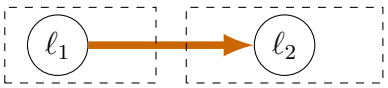
Fait 2 *V est une valuation totale sur les propositions apparaissant dans φ.*

Idée de la démonstration.
Car on considère tous les sommets de G_φ. ■

Fait 3 $V \models \varphi$.

Idée de la démonstration.
On considère une clause $\ell_1 \to \ell_2$ de φ. Montrons $V \models \ell_1 \to \ell_2$. Supposons que $V \models \ell_1$. Ainsi, ℓ_1 a été mis à vrai dans construireValuation : on avait $\ell_1 \in C_1$ où C_1 est finale dans un certain graphe G_1.

— Si ℓ_1 et ℓ_2 dans une même cfc de G_φ, alors ℓ_2 est aussi mis à vraie, comme ℓ_1.
— Sinon, dans G_φ :



Mais au moment, où ℓ_1 est affecté, la classe de ℓ_2 a déjà été supprimée car C_1 est finale :

Mais lorsque ℓ_2 a été supprimé, il n'était pas dans une classe initiale $\bar{C}$ car $\ell_1 \to \ell_2$ était un arc entrant dans la classe de ℓ_2. Ainsi, ℓ_2 était dans une classe finale C_2 et il a été affecté à vraie. Ainsi, $V \models \ell_2$.

■ ■ ■

Application

Soit $\mathcal{C}$ un ensemble fini de créneaux horaires, soit $\mathcal{P}$ un ensemble fini de professeurs et G un ensemble fini de groupes d'élèves. Chaque professeur $i \in \mathcal{P}$ dispose d'un ensemble de créneaux disponibles $\mathcal{C}_i \subseteq \mathcal{C}$, chaque groupe $j \in \mathcal{G}$ possède un ensemble de créneaux disponibles $\mathcal{D}_i \subseteq \mathcal{C}$ et d'un ensemble d'heures $\mathcal{R}_{ij}$ à enseigner à un groupe d'élèves $j \in \mathcal{G}$. On souhaite créer un emploi du temps qui satisfait les contraintes.

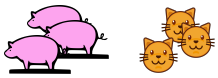
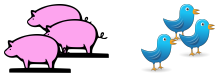
Dans le cas général, le problème est NP-complet [EIS75] mais si pour tout professeur $i \in \mathcal{P}$, $\mathbf{card}(\mathcal{C}_i) \leq 2$, alors le problème est réductible à 2SAT en temps polynomial et est donc dans P. Dans ce cas, si $\mathcal{R}_{ij} \leq 2$. Si $\mathcal{R}_{ij} = 2$, alors on peut supprimer le professeur i et considérer le groupe j comme disponible sur ces deux créneaux. Si $\mathbf{card}(\mathcal{C}_i) = 1$, alors on supprime le professeur i et on affectue un créneau à l'unique groupe j demandé et on considère le groupe j comme non disponible sur ce créneau.

On considère alors que $\mathcal{R}_{ij} \leq 1$. Soit $\mathcal{COMP}$ l'ensemble des couples (i, j) où $i \in \mathcal{P}$, $j \in \mathcal{G}$ et $\mathcal{R}_{ij} = 1$. Pour tout $(i, j) \in \mathcal{COMP}$, on introduit la variable propositionnelle v_{ij} qui signifie 'i fait cours à j sur son premier créneau disponible'.

On note c_i^1 et c_i^2 le créneau n° 1 et n° 2 du professeur i. L'instance de 2-SAT est alors l'ensemble des clauses suivantes :

- v_{ij} si $(i, j) \in \mathcal{COMP}$ si $c_i^2 \notin \mathcal{D}_j$;
- $\neg v_{ij}$ si $(i, j) \in \mathcal{COMP}$ si $c_i^1 \notin \mathcal{D}_j$;
- pour tout (i, i', j) tel que $(i, j), (i', j) \in \mathcal{COMP}$ et $i \neq i'$:
  - $(v_{ij} \wedge v_{i'j}) \rightarrow \bot$ si $c_i^1 = c_{i'}^1$;
  - $(\neg v_{ij} \wedge v_{i'j}) \rightarrow \bot$ si $c_i^2 = c_{i'}^1$;
  - $(v_{ij} \wedge \neg v_{i'j}) \rightarrow \bot$ si $c_i^1 = c_{i'}^2$;
  - $(\neg v_{ij} \wedge v_{i'j}) \rightarrow \bot$ si $c_i^2 = c_{i'}^2$;
- pour tout (i, j, j') tel que $(i, j), (i', j) \in \mathcal{COMP}$ et $j \neq j'$:
  - $(v_{ij} \wedge v_{ij'}) \rightarrow \bot$;
  - $(\neg v_{ij} \wedge \neg v_{ij'}) \rightarrow \bot$.

	lundi	mardi	mercredi	jeudi	vendredi
8h-10h					
10h-12h					
14h-16h					
16h-18h					

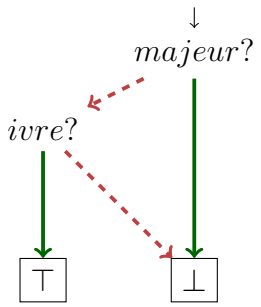
Professeur	Groupes
	
	
	

Chapitre 3

Diagrammes de décision binaire

Définition 52 (diagramme de décision binaire, [BA12])
Un **diagramme de décision binaire** (BDD) est un graphe acyclique orienté. Il n'y a qu'une racine. Les nœuds internes sont des propositions. Il y a au plus deux feuilles : $\top$ et $\bot$. Chaque proposition apparaît au plus qu'une fois sur chaque branche.

Exemple 53 *Voici un diagramme de décision binaire pour savoir si une personne est hors-la-loi selon la règle 'une personne ivre est une personne majeure' :*



3.1 Réduction

On réduit la taille d'un BDD en appliquant les opérations suivantes.

1. Si un noeud x admet deux arcs sortants (positif et négatif) qui pointe vers le même noeud, on supprime le noeud x (voir figure 3.1) ;
2. Si deux sous-graphes sont les même, on les fusionne.

3.2 Ordonnancement des propositions atomiques

Souvent on demande à ce que les propositions atomiques apparaissent dans le même ordre sur toutes les branches d'un diagramme de décision binaire. On parle alors de **diagramme de décision binaire ordonné** (OBDD). Dans toute la suite, on suppose que les diagrammes de décision binaire sont ordonnés.

3.3 Opérations

La négation d'un BDD s'obtient en échangeant les feuilles $\top$ et $\bot$. Nous décrivons maintenant comment obtenir le BDD conjonction de deux OBDDs ordonnés avec le même ordre des

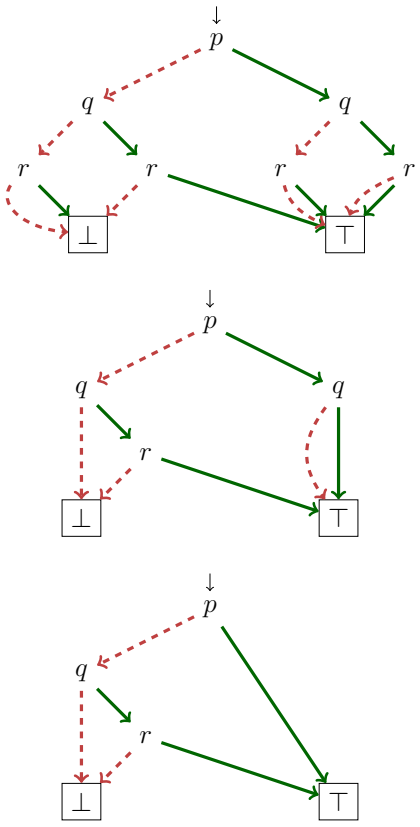


FIGURE 3.1 – Opérations de réduction.

variables. La figure 3.2 montre le processus. On construit un graphe temporaire, sorte de graphe produit, qui simule une descente dans les deux BDDs. Les feuilles sont obtenues en réalisant la conjonction.

Le BDD de la conjonction est obtenu en fusionnant les feuilles identiques puis en réduisant la taille du BDD comme montré en section 3.1.

3.4 Propriétés

Théorème 54 ([BK08], p. 399) *Considérons un ordre sur les propositions atomiques. Deux OBDDs réduits sont isomorphes ssi ils sont équivalents.*

Théorème 55 ([BK08], p. 406 ; [THY93] ; [BW96]) *Le problème de décider si un ordre sur les propositions atomiques donne le OBDD réduit le plus petit est NP-dur.*

3.5 Exercices

Exercice 1 *Construire le BDD de* $(p \vee q) \wedge (\neg p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee \neg q)$.

Exercice 2 *Donner le BDD réduit du BDD montré à la figure 3.2.*

Exercice 3 *Expliquer comment utiliser les BDDs pour voir si une formule est satisfiable.*

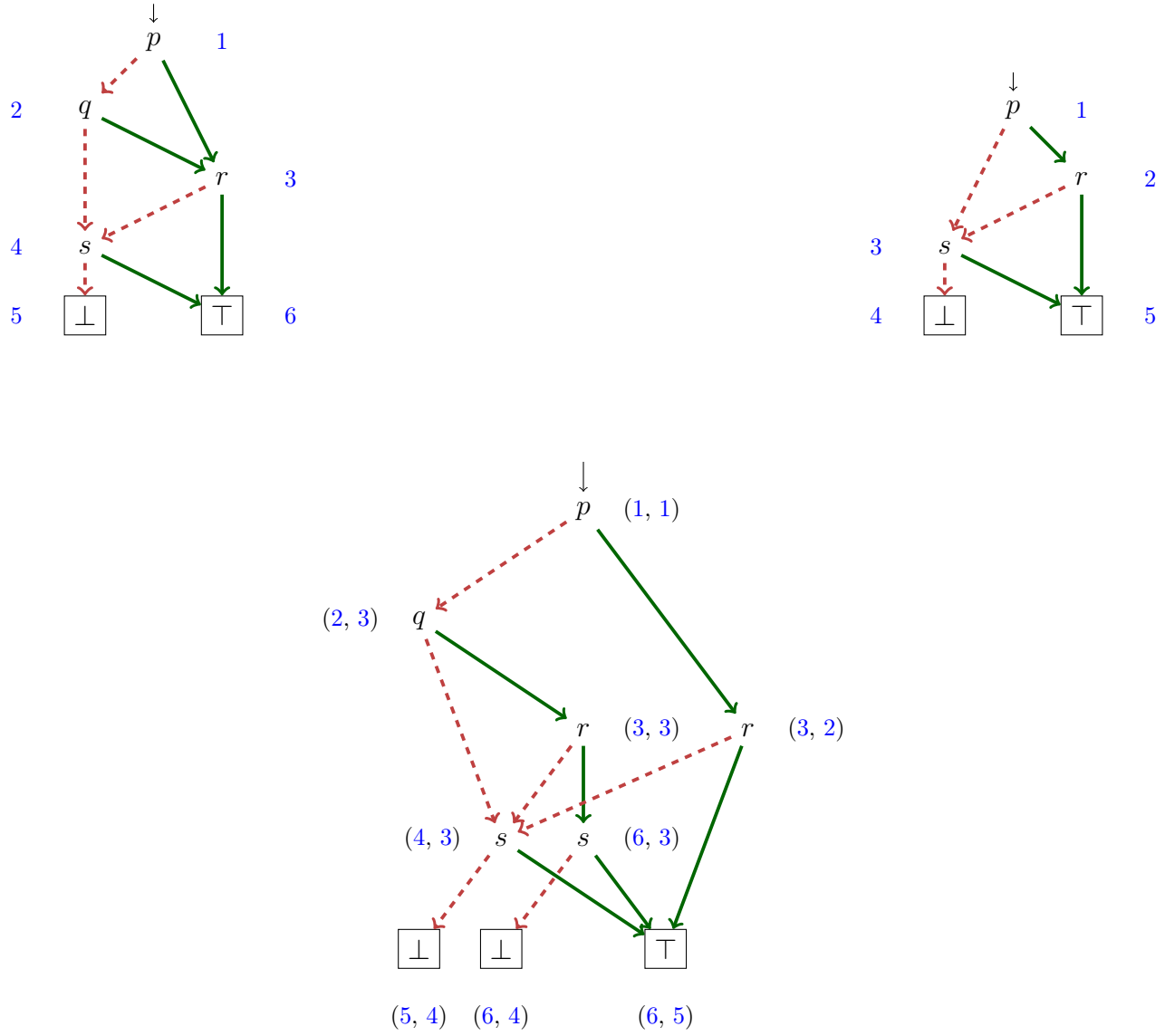
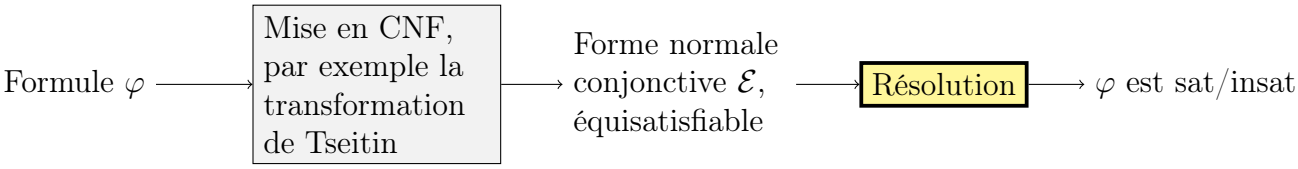


FIGURE 3.2 – Opération **et**.

Chapitre 4

Résolution en logique propositionnelle

Points du programme de l'agrégation
Théorème de complétude du calcul propositionnel



4.1 Règle de résolution

Définition 56 (règle de résolution)
([BA12], p. 82) La **règle de résolution** est la règle

$$\frac{(p \vee \ell_1 \vee \dots \ell_n) \qquad (\neg p \vee \ell'_1 \vee \dots \vee \ell'_k)}{(\ell_1 \vee \dots \ell_n \vee \ell'_1 \vee \dots \vee \ell'_k)}$$

à permutation près des littéraux et en supprimant les répétitions de littéraux [1].
La nouvelle clause obtenue $(\ell_1 \vee \dots \ell_n \vee \ell'_1 \vee \dots \vee \ell'_k)$ s'appelle le **résolvant**.

Notation 57 *On note $\bot$ la **clause vide** [2].*

Exemple 58 *Le **modus ponens** $\frac{p \qquad (p \to q)}{q}$ en est un cas particulier.*

Exemple 59

$$\frac{(p \vee q) \qquad (\neg p \vee r)}{(q \vee r)}$$

1. Une présentation plus bas niveau les clauses sont représentées des ensembles évite ce problème. Par exemple $(p \vee \neg q)$ est représentée par l'ensemble $\{p, \neg q\}$. Nous préférons ici une présentation haut niveau.
2. Dans [BA12], p. 82, elle est notée $\square$.

4.2 Arbres de preuve

Définition 60 (preuve par résolution)
Soit $\mathcal{E}$ une forme normale conjonctive[3]. Une **preuve par résolution** de φ pour $\mathcal{E}$ est un arbre fini étiqueté par des clauses tel que :
- Les feuilles sont des clauses de $\mathcal{E}$;
- La racine est φ ;
- Chaque nœud interne correspond à l'application de la règle de résolution.

Exemple 61 *([BA12], p. 83)*

$$\dfrac{\dfrac{(p \vee q \vee \neg r) \qquad r}{(p \vee q)} \qquad \dfrac{(p \vee \neg q \vee r) \qquad \neg r}{(p \vee \neg q)}}{p}$$

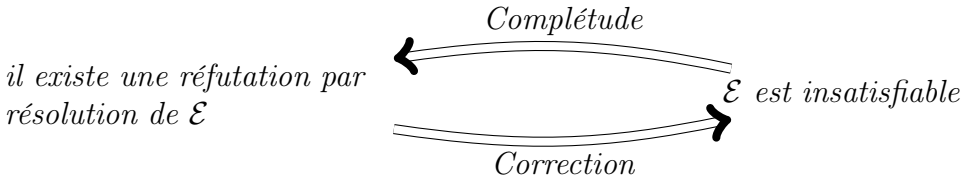
Définition 62 (réfutation par résolution)
Soit $\mathcal{E}$ une forme normale conjonctive. Une **réfutation par résolution** de $\mathcal{E}$ est une preuve par résolution de $\bot$ pour $\mathcal{E}$.

Exemple 63 *([BA12], p. 83)*

$$\dfrac{\dfrac{\dfrac{(\neg p \vee \neg q \vee r) \qquad \neg r}{(\neg p \vee \neg q)} \qquad (\neg p \vee q)}{\neg p} \qquad p}{\bot}$$

4.3 Correction et complétude

Théorème 64 *([BA12], p. 82) Soit $\mathcal{E}$ un forme normale conjonctive.*



Idée de la démonstration.
$\boxed{\Rightarrow}$ Par contraposée, supposons $\mathcal{E}$ satisfiable, i.e. il existe une valuation V telle que $V \models \mathcal{E}$.

Lemme 65 *Si* $\dfrac{C_1 \qquad C_2}{C}$, *$V \models C_1$ et $V \models C_2$ alors $V \models C$.*

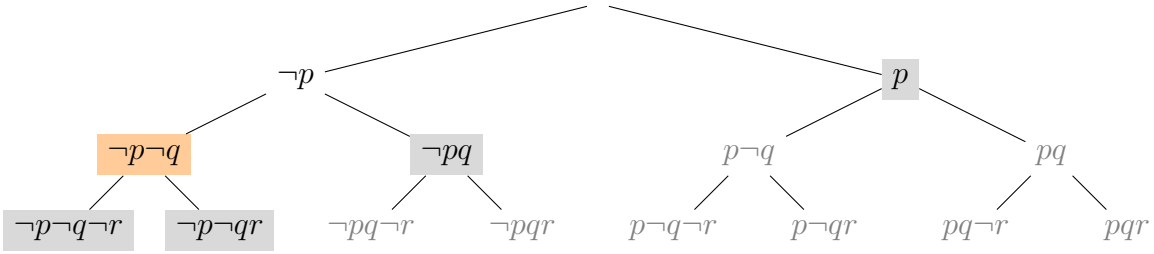
On montre par induction structurelle sur les preuves $\dfrac{\vdots}{\varphi}$ pour $\mathcal{E}$ la propriété

$$\mathcal{P}(\dfrac{\vdots}{\varphi}) : V \models \varphi$$

Ainsi, il n'existe pas de preuve par résolution de $\bot$ pour $\mathcal{E}$.

3. Certains auteurs [BA12] parlent d'ensemble de clauses.

$\boxed{\Leftarrow}$ Supposons $\mathcal{E}$ insatisfiable. Considérons l'arbre de décision sur les propositions $p_1, \ldots, p_n$ apparaissant dans $\mathcal{E}$. Un nœud interne correspond à une valuation partielle et une feuille à une valuation totale sur $p_1, \ldots, p_n$. Un **nœud d'échec** est un nœud le plus proche de la racine, dont la valuation partielle rend une des clauses de $\mathcal{E}$ fausse. Un **nœud d'inférence** est un nœud dont les fils sont des nœuds d'échec.



Lemme 66 *Un nœud est un nœud d'échec ssi il y a une clause dont tous les négations des littéraux de cette clause apparaîssent dans ce nœud.*

Lemme 67 *La racine est un noeud d'échec ssi la clause vide $\bot$ est dans $\mathcal{E}$.*

Lemme 68 *Si $\mathcal{E}$ est insatisfiable et que la racine n'est pas un nœud d'échec, alors il existe un nœud d'inférence.*

Idée de la démonstration.

Par l'absurde, supposons qu'il n'y a pas de nœud d'inférence. Alors on considère un nœud d'échec n_0. Son frère n'est pas un nœud d'échec mais son sous-arbre en contient un strictement plus profond. On construit une suite infinie n_0, n_1, etc. de nœuds d'échec de plus en plus profond. Contradiction. ■

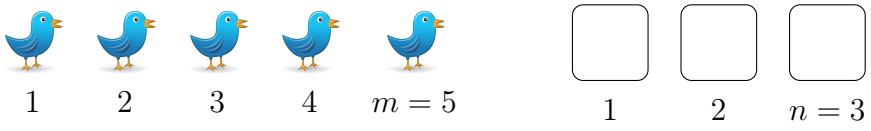
Lemme 69 *Soit n un nœud d'inférence de fils n_1 et n_2. Si C_1 est falsifié par n_1 et C_2 est falsifié par n_2, alors on peut appliquer la règle de résolution $\dfrac{C_1 \quad C_2}{C}$ et n falsifie C.*

L'algorithme suivant construit une preuve par résolution de $\bot$ pour $\mathcal{E}$:

tant que la clause vide $\bot$ n'est pas produite **do**
| Appliquer la règle de résolution sur un nœud d'inférence.

L'algorithme termine car le nombre de nœuds d'échec décroit strictement : avec $\mathcal{E} \wedge C$ à la place de $\mathcal{E}$, un ancêtre du nœud d'inférence considéré devient nœud d'échec. ■

4.4 Taille des arbres de preuve



Définition 70 (principe des pigeons et trous)
PHP_n^m est la formule

$$\bigwedge_{i=1}^{m} \bigvee_{j=1}^{n} \left(i \text{ dans } j \right) \wedge \bigwedge_{i=1}^{m} \bigwedge_{j=1}^{m} \bigwedge_{k=1}^{n} \left(\neg \left(i \text{ dans } k \right) \vee \neg \left(j \text{ dans } k \right) \right)$$

Théorème 71 *([AB09], p. 310) Pour tout $n \geq 2$, tout arbre de réfutation de PHP_{n-1}^n contient au moins $2^{n/20}$ nœuds.*

Certificat pour SAT :	valuation	(de taille polynomiale)
Certificat pour VALIDE :	arbre de preuve par résolution	(pas toujours de taille polynomiale, cf. Th. 71)

Chapitre 5

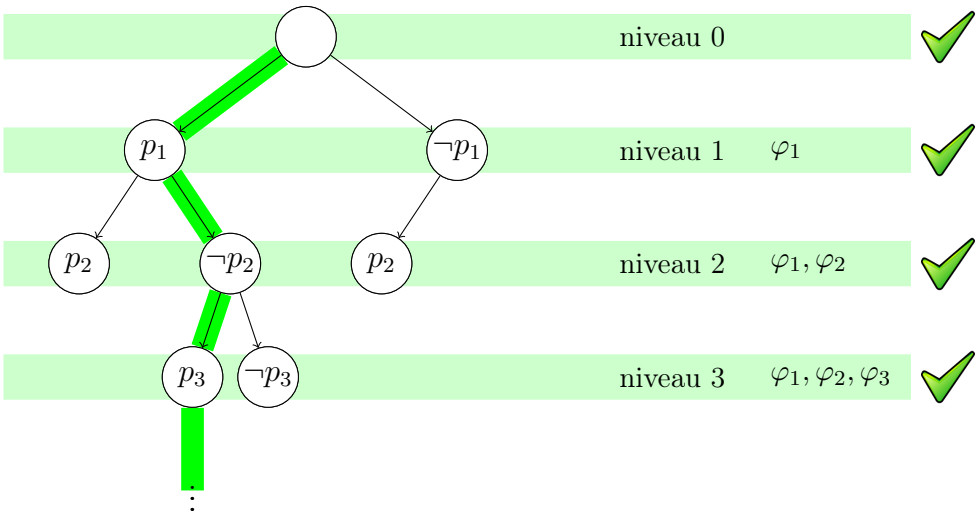
Théorème de compacité

5.1 Énoncé

Théorème 72 *Soit E un ensemble de formules de la logique propositionnelle.*
Si tout sous-ensemble fini de E est satisfiable, alors E l'est aussi.

Idée de la démonstration.
On considère :
— $\varphi_1, \varphi_2, \ldots$ une énumération des formules de E ;
— et $p_1, p_2, \ldots$ une énumération des propositions atomiques.

On construit l'arbre [1] suivant par récurrence.
— La racine est le nœud de niveau 0.
— Supposons que tous les nœuds de niveau $n-1$ soient construits. On attache :
  — on attache un nœud p_n à une branche $\pm p_1, \ldots, \pm p_{n-1}$
    si $\{\pm p_1, \ldots, \pm p_{n-1}, p_n, \varphi_1, \ldots, \varphi_n\}$ est satisfiable ;
  — on attache un nœud $\neg p_n$ à une branche $\pm p_1, \ldots, \pm p_{n-1}$
    si $\{\pm p_1, \ldots, \pm p_{n-1}, \neg p_n, \varphi_1, \ldots, \varphi_n\}$ est satisfiable ;



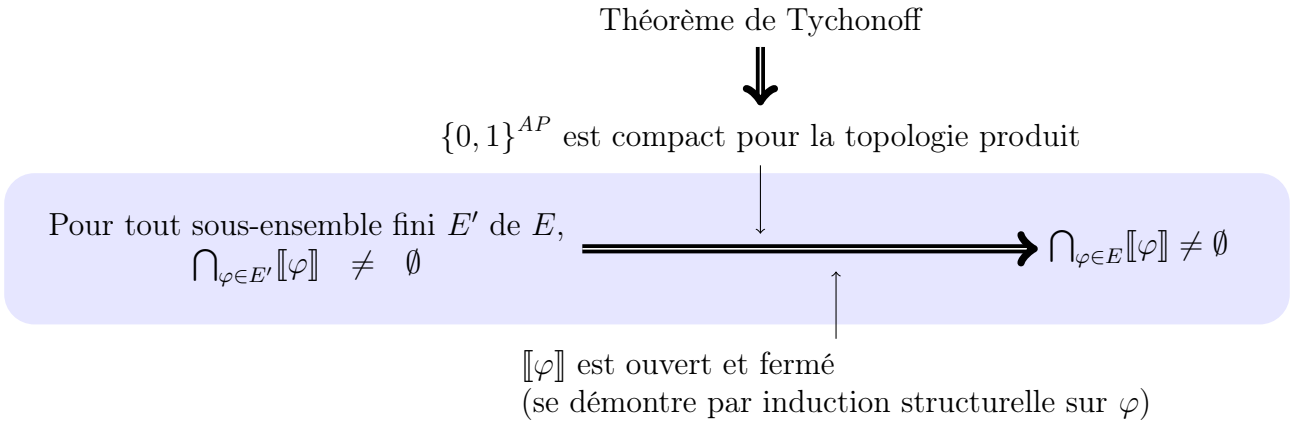
Comme pour tout $n \in \mathbb{N}$, $\{\varphi_1, \ldots, \varphi_n\}$ est satisfiable. On peut toujours continuer à attacher des nœuds. Ainsi, l'arbre est infini.

Comme l'arbre est infini et à branchement fini, d'après le **lemme de König**, il a une branche infinie. Cette branche définit une valuation qui satisfait T. ■

1. C'est un sous-arbre de l'arbre de décision.

5.2 Démonstration avec des notions de topologie

Référence : [Tru97], p. 330, point (v) ⇒ (vi).



5.3 Applications

Corollaire 73 (Théorème de De Bruijn-Erdős)
Un graphe G est 3-coloriable si et seulement si tout sous-graphe fini de G est 3-coloriable.

IDÉE DE LA DÉMONSTRATION.

⇒ Immédiat.
⇐ Supposons que tout sous-graphe fini de G est 3-coloriable. Soit S' un sous-ensemble de sommets de $G = (S, A)$. On note :

$$E_{S'} := \{\Big(\bigvee_{c\in\{●,●,●\}} p_{s,c}\Big) \wedge \bigwedge_{c,c'\in\{●,●,●\}c\neq c'} (p_{s,c} \to \neg p_{s,c'}) \mid s \in S'\}$$
$$\{\bigwedge_{c\in\{●,●,●\}} (p_{s,c} \to \neg p_{t,c}) \mid s,t \in S' \text{ et } (s,t) \in A\}.$$

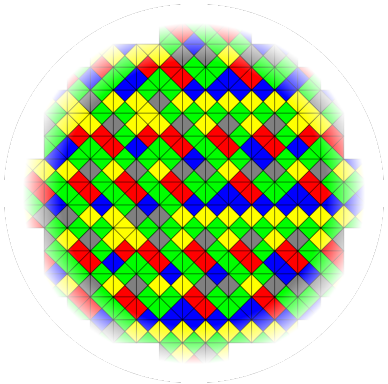
On a :

Lemme 74 *$G_{|S'}$ est 3-coloriable ssi $E_{S'}$ est satisfiable.*

Soit E' une partie finie de E_S. Il existe $S' \subseteq S$, S' fini, tel que $E' \subseteq E_{S'}$. Comme $G_{S'}$ est un graphe fini, il est 3-coloriable, et donc par le lemme 74, $E_{S'}$ est satisfiable. Comme $E' \subseteq E_{S'}$, E' est satisfiable.

D'après le **théorème de compacité de la logique propositionnelle**, E_S est satisfiable. D'après le lemme 74, $G = G_{|S}$ est 3-coloriable. ■

Corollaire 75 *Le plan est pavable par un ensemble de type de tuiles T si et seulement si tout carré fini est pavable par T.*



Deuxième partie

Logique des prédicats du premier ordre

Chapitre 6

Syntaxe et sémantique

Points du programme de l'agrégation
Logique du premier ordre : aspects syntaxiques. Langages, termes, formules. Variables libres et variables liées.
Logique du premier ordre : aspects sémantiques. Interprétation d'une formule dans un modèle. Validité, satisfiabilité.

But : écrire des propriétés et raisonner sur des structures
(base de données, groupes, arithmétiques, etc.).

6.1 Modèles

Définition 76 (signature)
Une **signature** Σ est un ensemble dénombrable de **symboles de fonctions** et un ensemble de **symboles de prédicats**, munis de leurs arités.

Exemple 77 *La signature avec les symboles de fonctions 0 et 1 d'arité 0, + d'arité 2, et les symboles de prédicats* estpair *d'arité 1 et = d'arité 2.*

Définition 78 (structure)
Une Σ-**structure** (ou Σ-modèle) est la donnée $\mathcal{M} = \langle D, .^{\mathcal{M}} \rangle$ où :
- D est un ensemble **non-vide** appelé **domaine** ;
- $.^{\mathcal{M}}$ est une fonction appelée **interprétation** qui :
  - à tout symbole de fonction f d'arité n dans Σ associe une fonction $f^{\mathcal{M}} : D^n \to D$;
  - à tout symbole de prédicat p d'arité n dans Σ associe une fonction $p^{\mathcal{M}} : D^n \to \{0, 1\}$.

Définition 79 (symbole de constante)
Un symbole de fonction d'arité 0 s'appelle un **symbole de constante**.

Exemple 80 *Soit Σ la signature avec les symboles de constantes e, a, b, le symbole de fonction $\star$ et le symbole de prédicat $=$ d'arité 2. Le* ***groupe diédral*** *d'ordre 4 (aussi appelé* ***groupe de Klein****) est la Σ-**structure** $\mathcal{M} = \langle D, e^{\mathcal{M}}, a^{\mathcal{M}}, b^{\mathcal{M}}, \star^{\mathcal{M}}, =^{\mathcal{M}} \rangle$ où :*
- *$D = \{e, a, b, c\}$;*
- *$e^{\mathcal{M}} = e$; $e^{\mathcal{M}} = e$; $a^{\mathcal{M}} = a$; $b^{\mathcal{M}} = b$;*
- *$\star^{\mathcal{M}}$ est la fonction de D^2 dans D définie par la table ci-contre.*
- *$=^{\mathcal{M}}$ est $=_D$.*

$\star$	e	a	b	c
e	e	a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

Soit $\mathcal{V}$ un ensemble dénombrable de **variables**.

Définition 81 (assignation)
Etant donnée une Σ-structure $\mathcal{M} = (D, .^{\mathcal{M}})$, une **assignation** des variables v est une fonction de $\mathcal{V}$ dans D.

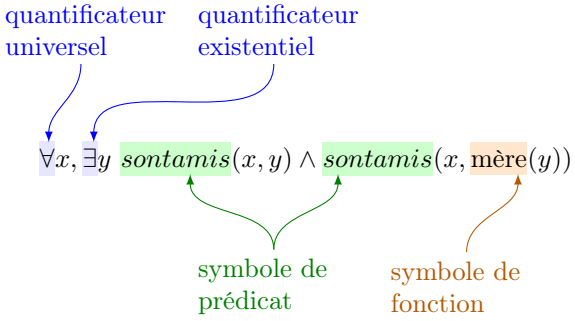
6.2 Langage

6.2.1 Syntaxe

Définition 82 (terme)
L'ensemble des Σ-termes est défini par induction :
- Une variable x de $\mathcal{V}$ est un Σ-terme ;
- Pour tout symbole de fonction f d'arité n de Σ, et pour tous Σ-termes $t_1, \ldots, t_n$,
  - $f(t_1, \ldots, t_n)$ est un Σ-terme.

$$1 + 1$$
$$1 + (2 + x)$$
$$(x \times cos(y))$$

Définition 83 (formule)
L'ensemble des Σ-formules est défini par induction :
- Pour tout symbole de prédicat p d'arité n de Σ, et pour tous Σ-termes $t_1, \ldots, t_n$,
  - $p(t_1, \ldots, t_n)$ est une Σ-formule ;
- Pour toute Σ-formule φ,
  - $\neg\varphi$ est une Σ-formule ;
- Pour toutes formules φ et ψ,
  - $(\varphi \vee \psi)$ est une Σ-formule ;
  - $(\varphi \wedge \psi)$ est une Σ-formule ;
- Pour toute variable x, toute Σ-formule φ,
  - $\forall x\varphi$ est une Σ-formule ;
  - $\exists x\varphi$ est une Σ-formule.



Définition 84 (formule atomique)
Une **formule atomique** est une formule de la forme $p(t_1, \ldots, t_n)$.

6.2.2 Sémantique

On va définir $\mathcal{M}, v \models \varphi$ qui signifie la Σ-formule φ est vraie dans $\mathcal{M}$ avec l'assignation v.

Définition 85 (assignation étendue aux termes)
D'abord, on étend v en $v^{\mathcal{M}}$ aux termes avec :
- $v^{\mathcal{M}}(x) = v(x)$;
- $v^{\mathcal{M}}(f(t_1, \ldots, t_n)) = f^{\mathcal{M}}(v^{\mathcal{M}}(t_1), \ldots, v^{\mathcal{M}}(t_n))$.

Définition 86 (Conditions de vérité)

- $\mathcal{M}, v \models p(t_1, \ldots, t_n)$ si $p^{\mathcal{M}}(v^{\mathcal{M}}(t_1), \ldots, v^{\mathcal{M}}(t_n)) = 1$;
- $\mathcal{M}, v \models \neg\varphi$ si $\mathcal{M}, v \not\models \varphi$;
- $\mathcal{M}, v \models (\varphi \vee \psi)$ si $\mathcal{M}, v \models \varphi$ ou $\mathcal{M}, v \models \psi$;
- $\mathcal{M}, v \models (\varphi \wedge \psi)$ si $\mathcal{M}, v \models \varphi$ et $\mathcal{M}, v \models \psi$;
- $\mathcal{M}, v \models \exists x\varphi$ s'il existe $d \in D$ tel que $\mathcal{M}, v[x := d] \models \varphi$.
- $\mathcal{M}, v \models \forall x\varphi$ si pour tout $d \in D$ on a $\mathcal{M}, v[x := d] \models \varphi$.

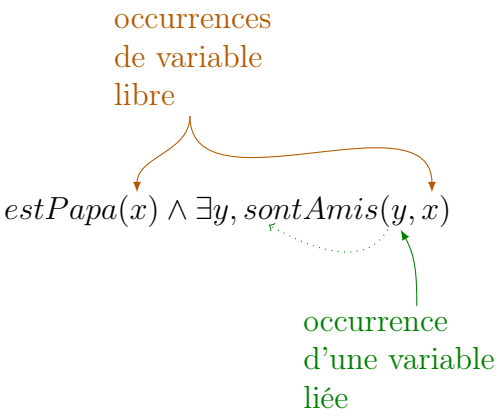
Remarque 87 *Dans le contexte de la* ***logique du premier ordre égalitaire****, le symbole de prédicats $=$ est interprété par l'égalité :*
- $\mathcal{M}, v \models t = t'$ *si* $v(t) = v(t')$.

6.3 Variables libres/liées

Définition 88 (occurrence libre)
Une occurrence d'une variable x est libre si elle n'est pas sous la portée d'un quantificateur.

Définition 89 (occurrence liée)
Une occurrence est liée si elle n'est pas libre.

Définition 90 (variable libre)
x est une variable libre dans φ s'il existe une occurrence libre de x dans φ.



Notation 91 *On note $\varphi(x_1, \ldots, x_n)$ pour dire que les variables libres de φ sont dans $\{x_1, \ldots, x_n\}$.*

Définition 92 (terme clos)
Un terme est **clos** s'il est sans variables libres.

Définition 93 (formule close)
Une formule est **close** si elle est sans variables libres.

Définition 94 (clôture universelle)
La **clôture universelle** de $\varphi(x_1, \ldots, x_n)$ est la formule $\forall x_1 \ldots \forall x_n \varphi$.

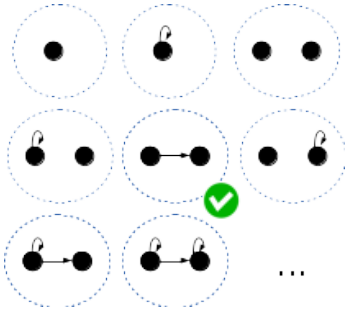
Notation 95 *Si φ est close, $\mathcal{M}, v \models \varphi$ ne dépend pas de v. On note alors $\mathcal{M} \models \varphi$.*

6.4 Satisfiable, valide, conséquence sémantique

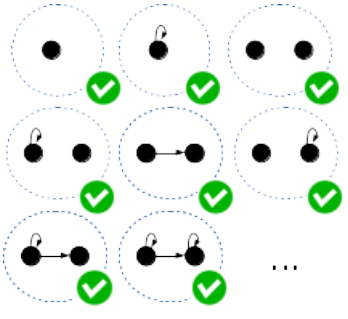
On ne s'intéresse qu'aux formules closes ([BA12], p. 138).

Définition 96 (satisfiable)
Une formule φ close est **satisfiable** s'il existe un modèle $\mathcal{M}$ telle que $\mathcal{M} \models \varphi$.

Définition 97 (valide)
Une formule φ close est **valide** (ou est une tautologie) si pour tout modèle $\mathcal{M}$, on a $\mathcal{M} \models \varphi$.



φ est satisfiable



φ est valide

6.5 Model checking

Définition 98 (problème de model checking en logique du premier ordre)
Le problème de model checking est :
— Entrée : un modèle fini $\mathcal{M}$, une formule close φ ;
— Sortie : oui si $\mathcal{M} \models \varphi$, non sinon.

Proposition 99 *Le problème du model checking en logique du premier ordre est PSPACE-complet.*

6.6 Problème de la satisfiabilité et problème de la validité

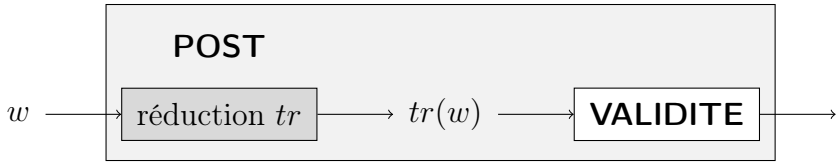
Définition 100 (problème de la satisfiabilité)
Le **problème de la satisfiabilité** est :
— Entrée : une formule close φ ;
— Sortie : oui si φ est satisfiable, non sinon.

Définition 101 (problème de la validité)

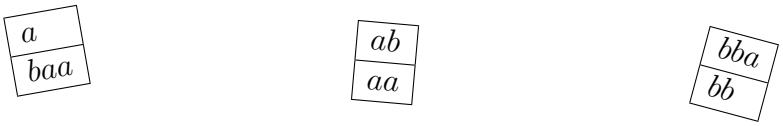
Le **problème de la validité** est :
— Entrée : une formule close φ ;
— Sortie : oui si φ est valide, non sinon.

Proposition 102 *([GLM01], p. 205) Le problème de la satisfiabilité et le problème de la validité en logique du premier ordre sont indécidables.*

Idée de la démonstration.



Considérons une instance w **POST** ne contenant pas de tuile $\begin{array}{|c|}\hline \epsilon \\ \hline \epsilon \\ \hline \end{array}$:



On construit la formule $tr(w) = (\varphi \to \psi)$ où :
— φ est la conjonction de
  — $p(\epsilon, \epsilon)$ où ϵ est un symbole de constante ;
  — $\forall x, y, p(x, y) \to p(u_i(x), v_i(y))$ pour toute tuile $\begin{array}{|c|}\hline u_i \\ \hline v_i \\ \hline \end{array}$ où, si $m = a_1 \dots a_n$ est un mot, on note $m(.) = a_n(\dots a_1(t) \dots)$;
— $\psi := \exists x, p(a(x), a(x)) \vee p(b(x), b(x))$.
On a :
— $tr(w)$ est calculable à partir de w ;
— w est une instance positive de **POST** ssi $tr(w)$ est valide.

■

Chapitre 7

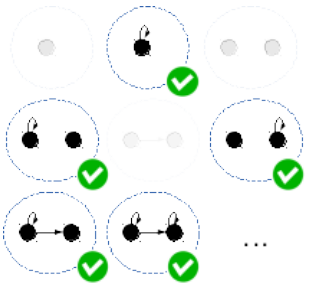
Théories

Points du programme de l'agrégation
Théories cohérentes, théories complètes. Théories décidables, indécidables. Exemples de théories : égalité, arithmétique de Peano.

7.1 Définitions

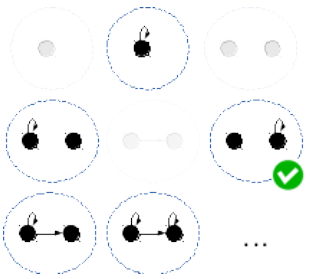
Définition 103 (Théorie)
Une **théorie** T est un ensemble de formules closes.

Définition 104 (T-valide [1])
φ close est **T-valide**, noté $T \models \varphi$, si pour tout modèle $\mathcal{M}$, on a $\mathcal{M} \models T$ implique $\mathcal{M} \models \varphi$.



Le **problème de la T-validité** est :
- Entrée : une formule close φ ;
- Sortie : oui si φ est T-valide, non sinon.

Définition 105 (T-satisfiable)
Une formule φ est **T-satisfiable** s'il existe un modèle $\mathcal{M}$ tel que $\mathcal{M} \models T$ et $\mathcal{M} \models \varphi$.



Le **problème de la T-satisfiabilité** est :
- Entrée : une formule close φ ;
- Sortie : oui si φ est T-satisfiable, non sinon.

Définition 106 (théorie décidable)
Une théorie T est **décidable** si le problème de T-validité est décidable.
ou de manière équivalente le problème de T-satisfiabilité est décidable.

Définition 107 (théorie cohérente)
Une théorie T est **cohérente** si elle est satisfiable.

Définition 108 (théorie (sémantiquement) complète)
Une théorie T est **(sémantiquement) complète**
si pour toute formule φ, on a $T \models \varphi$ ou $T \models \neg\varphi$.

1. Ou conséquence sémantique de T.

7.2 Exemples de théories

7.2.1 Théorie de l'égalité

$$T_{eq}\begin{cases}\forall x, x = x\\ \forall x, y, (x = y) \to (y = x)\\ \forall x, y, z, (x = y \wedge y = z) \to x = z\\ \forall x_1, \ldots, x_n, y_1, \ldots, y_n, (\bigwedge_i (x_i = y_i)) \to f(\vec{x}) = f(\vec{y}) \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{pour tout symbole de fonction } f \text{ d'arité } n\\ \forall x_1, \ldots, x_n, y_1, \ldots, y_n, (\bigwedge_i (x_i = y_i)) \to (p(\vec{x}) \leftrightarrow p(\vec{y}))\\ \qquad\qquad\qquad\qquad\qquad\qquad \text{pour tout symbole de prédicat } p \text{ d'arité } n\end{cases}$$

Proposition 109 *Si φ est T_{eq}-satisfiable ssi φ est satisfiable dans un modèle où le symbole $=$ est interprété par l'égalité.*

Idée de la démonstration.

($\Leftarrow$) Soit $\mathcal{M}$ un modèle où le symbole $=$ est interprété par l'égalité tel que $\mathcal{M} \models \varphi$. Par ailleurs, on peut montrer que $\mathcal{M} \models T_{eq}$, donc φ est T_{eq}-satisfiable.

($\Rightarrow$) Soit $\mathcal{M} = (D, .^{\mathcal{M}})$ un modèle tel que $\mathcal{M} \models T_{eq}$ et $\mathcal{M} \models \varphi$. Comme $\mathcal{M} \models \forall x, x = x$, $\mathcal{M} \models \forall x, y, (x = y) \to (y = x)$ et $\mathcal{M} \models \forall x, y, z, (x = y \wedge y = z) \to x = z$, la relation $=^{\mathcal{M}}$ est une relation d'équivalence.

Soit $\mathcal{M}' = (D', .^{\mathcal{M}'})$ tel que

1. $D' = D/_{=^{\mathcal{M}}}$;
2. Pour tout symbole de fonction f d'arité n,

$$\begin{array}{rccc} f^{\mathcal{M}'} : & D'^n & \to & D' \\ & ([t_1], \ldots, [t_n]) & \mapsto & [f^{\mathcal{M}}(t_1, \ldots, t_n)] \end{array}$$

3. Pour tout symbole de prédicat p d'arité n,

$$\begin{array}{rccc} p^{\mathcal{M}'} : & D'^n & \to & \{0, 1\} \\ & ([t_1], \ldots, [t_n]) & \mapsto & p^{\mathcal{M}}(t_1, \ldots, t_n) \end{array}$$

$f^{\mathcal{M}'}$ est bien définie car $\mathcal{M} \models \forall x_1, \ldots, x_n, y_1, \ldots, y_n, (\bigwedge_i (x_i = y_i)) \to f(\vec{x}) = f(\vec{y})$.
$p^{\mathcal{M}'}$ est bien définie car $\mathcal{M} \models \forall x_1, \ldots, x_n, y_1, \ldots, y_n, (\bigwedge_i (x_i = y_i)) \to (p(\vec{x}) \leftrightarrow p(\vec{y}))$.
Enfin, on démontre que pour toute formule ψ,

$$\mathcal{P}(\psi) : \mathcal{M}, v \models \psi \text{ ssi } \mathcal{M}', v' \models \psi$$

où $v'(x) = [v(x)]$.

■

7.2.2 Théorie des groupes ([Lal90], p. 139) ([DNRC01], p. 105)

$$T_{groupes}\begin{cases}\text{Théorie de l'égalité}\\ \forall x, (x \times e = x)\\ \forall x, x \times x^{-1} = e\\ \forall x, y, z, ((x \times y) \times z) = (x \times (y \times z))\end{cases}$$

Proposition 110 *$T_{groupes} \models \varphi$ ssi pour tout groupe $(G, .)$, on a $(G, .) \models \varphi$.*

7.2.3 Théorie des ordres denses ([DNRC01], p. 130)

$$T_O \begin{cases} \text{Théorie de l'égalité} \\ \forall x, y, \neg(x < y \wedge y < x) \\ \forall x, y, z, ((x < y \wedge y < z) \to x < z) \\ \forall x, y, (x < y \vee x = y \vee y < x) \\ \forall x, y, \exists z, (x < y \to (x < z \wedge z < y)) \\ \forall x, \exists y, (x < y) \\ \forall x, \exists y, (y < x) \end{cases}$$

Proposition 111 *([DNRC01], p. 130) T_O est complète.*

Idée de la démonstration.

On utilise l' **élimination des quantificateurs**. On transforme une formule close φ en la formule $\top$ ou $\bot$ en utilisant les règles de réécriture de sous-formules suivantes, qui préservent la T_O-équivalence :

Supprimer les négations		
$\neg(x = x)$	se réécrit en	$\bot$
$(x = x)$	se réécrit en	$\top$
$\neg(x = y)$	se réécrit en	$x < y \vee y < x$
$\neg(x < y)$	se réécrit en	$x = y \vee y < x$

Se ramener à une sous-formule $\exists x K$ avec K conjunction		
$\exists x \psi$ où ψ sans quantificateur	se réécrit en	$\exists x \text{miseendnf}(\psi)$
$\exists x(\psi \vee \psi')$	se réécrit en	$(\exists x \psi) \vee (\exists x \psi')$
$\exists x \top$	se réécrit en	$\top$
$\exists x \bot$	se réécrit en	$\bot$
$(\psi \vee \bot)$	se réécrit en	ψ
$(\psi \wedge \bot)$	se réécrit en	$\bot$
$(\psi \vee \top)$	se réécrit en	$\top$
$(\psi \wedge \top)$	se réécrit en	ψ

Élimination de $\exists x$		
$\exists x K$ avec K contenant $x = y$	se réécrit en	$K[x := y]$
$\exists x(K \wedge (y < z)$	se réécrit en	$(y < z) \wedge (\exists x K)$
$\exists x(K \wedge (y = z)$	se réécrit en	$(y = z) \wedge (\exists x K)$
$\exists x(\bigwedge_{i \in I}(x < x_i) \wedge (\bigwedge_{j \in J}(x_j < x)$ avec $I \cap J \neq \emptyset$	se réécrit en	$\bot$
$\exists x(\bigwedge_{i \in I}(x < x_i) \wedge (\bigwedge_{j \in J}(x_j < x)$ avec $I \cap J = \emptyset$	se réécrit en	$\bigwedge_{i \in I, j \in J}(x_j < x_i)$ ($\top$ si $I = \emptyset$ ou $J = \emptyset$)

■

Proposition 112 *$T_O \models \varphi$ ssi $(\mathbb{Q}, =, <) \models \varphi$ ssi $(\mathbb{R}, =, <) \models \varphi$.*

Proposition 113 *(pas de réf sauf `http://www.lsv.fr/~comon/Logique1b/cours2.4.pdf`) Le problème de T_O-validité est PSPACE-complet.*

Idée de la démonstration.

— On construit un algorithme en espace polynomial, récursif sur la formule φ, et qui se souvient à chaque étape d'une assignation des variables libre $v : \mathcal{V} \to \mathbb{Q}$.
— On réduit en temps polynomial TQBF au problème de T_O-validité.

■

7.2.4 Théorie des corps clos ([DNRC01], p. 133)

$$T_{CC}\begin{cases}\text{Théorie de l'égalité}\\ \forall x,y,z,(x+y)+z=x+(y+z)\\ \forall x,y,x+y=y+x\\ \forall x,x+0=x\\ \forall x,x+(-x)=0\\ \forall x,y,z,(x\times y)\times z=x\times(y\times z)\\ \forall x,y,x\times y=y\times x\\ \forall x,x\times 1=x\\ \forall x,(x\neq 0)\rightarrow\exists y,x\times y=1\\ \forall x,y,z,x\times(y+z)=x\times y+x\times z\\ 0\neq 1\\ \forall x,y,(x\leq y\wedge y\leq x)\rightarrow x=y\\ \forall x,y,z,(x\leq y\wedge y\leq z)\rightarrow x\leq z\\ \forall x,y,(x\leq y)\vee(y\leq x)\\ \forall x,y,z,x\leq y\rightarrow(x+z\leq y+z)\\ \forall x,y,(x\leq 0\wedge y\leq 0)\rightarrow x\times y\leq 0\\ \forall x\exists y,(x=y\times y)\vee(x=-y\times y)\\ \forall x_0,\ldots,x_{n-1},\exists y,y^n+x_{n-1}y^{n-1}+\ldots x_1y+x_0=0 \text{ pour tout nombre impair } n\end{cases}$$

Proposition 114 *T_{CC} est complète.*

Idée de la démonstration.
Via **élimination des quantificateurs.**

`http://people.irisa.fr/Francois.Schwarzentruber/realqelim/`

■

Proposition 115 *$T_{CC} \models \varphi$ ssi $(\mathbb{R},=,0,1,+,\times) \models \varphi$.*

7.2.5 Arithmétique de Presburger ([DNRC01], p. 136)

$$T_{Pres}\begin{cases}\text{Théorie de l'égalité}\\ \forall x,x^{+1}\neq 0\\ \forall x,(x=0)\vee\exists y,x=y^{+1}\\ \forall x,y,x^{+1}=y^{+1}\rightarrow x=y\\ \forall x,x+0=x\\ \forall x,y,x+(y^{+1})=((x+y)^{+1})\\ (\varphi(0)\wedge(\forall y,\varphi(y)\rightarrow\varphi(y^{+1})))\rightarrow\forall x,\varphi(x) \text{ pour toute formule } \varphi(x)\end{cases}$$

Proposition 116 *$T_{Pres} \models \varphi$ ssi $(\mathbb{N},=,0,1,+) \models \varphi$.*

Proposition 117 *[Car08] Le problème de savoir si une $(=,0,1,+)$-formule close φ est vraie dans $(\mathbb{N},=,0,1,+)$ est décidable.*

Idée de la démonstration.
On note $[\![\psi]\!]$ l'ensemble des assignations $v : \mathcal{V} \to \mathbb{N}$ telles que $(\mathbb{N},=,0,1,+), v \models \psi$.

procédure deciderFormuleVraieSur$\mathbb{N}(\varphi)$
  Calculer par induction, pour toute sous-formule ψ de φ, une représentation finie de $[\![\psi]\!]$
  Si $[\![\varphi]\!] \neq \emptyset$ alors **accepter** sinon **rejeter**

Sans perte de généralité, on suppose que φ appartient au langage généré par la grammaire :

$$\varphi ::= (x=0) \mid (x=1) \mid (z=x+y) \mid \neg\varphi \mid (\varphi\vee\varphi) \mid \exists x\varphi$$

Détails de la représentation.

un entier	$\rightsquigarrow$	mot sur l'alphabet $\{0,1\}$ qui est la représentation binaire avec avec le bit de faible à gauche
Ex : 5	$\rightsquigarrow$	n'importe quel mot de la forme 1010^*
Ex : 6	$\rightsquigarrow$	n'importe quel mot de la forme 0110^*
une assignation, i.e. un n-uplet d'entiers	$\rightsquigarrow$	la **convolution** des représentations des entiers, c'est un mot sur l'alphabet $\{0,1\}^n$
Ex : l'assignation $[x{:=}5, y{:=}2, z{:=}3]$, i.e. le 3-uplet $(5,2,3)$	$\rightsquigarrow$	n'importe quel mot de la forme $\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}0\\1\\1\end{pmatrix}\begin{pmatrix}1\\0\\0\end{pmatrix}\begin{pmatrix}0\\0\\0\end{pmatrix}^*$
$[\![\psi]\!]$	$\rightsquigarrow$	un **automate** $\mathcal{A}_\psi$ qui reconnait le langage des mots qui représentent les assignations dans $[\![\psi]\!]$

Algorithme pour calculer la représentation finie (automate) $\mathcal{A}_\psi$ de $[\![\psi]\!]$.

fonction automate(φ)
  match φ **do**
    case $x = 0$: **retourner** automate pour 0^*
    case $x = 1$: **retourner** automate pour 10^*
    case $x + y = z$:
      retourner automate pour $\left[\begin{pmatrix}0\\0\\0\end{pmatrix} \cup \begin{pmatrix}1\\0\\1\end{pmatrix} \cup \begin{pmatrix}0\\1\\1\end{pmatrix} \cup \left\{\begin{pmatrix}1\\1\\0\end{pmatrix}.\left[\begin{pmatrix}0\\1\\0\end{pmatrix} \cup \begin{pmatrix}1\\0\\0\end{pmatrix} \cup \begin{pmatrix}1\\1\\1\end{pmatrix}\right]^*.\begin{pmatrix}0\\0\\1\end{pmatrix}\right\}\right]^*$
    case $(\psi_1 \vee \psi_2)$: **retourner** automate union de $automate(\psi_1)$ et $automate(\psi_2)$
    case $\neg\psi$: **retourner** automate complémentaire de $automate(\psi)$
    case $\exists x\psi$:
      retourner projection de $automate(\psi)$ sur la coordonnée de x
      (i.e. on efface la coordonnée x des étiquettes des transitions de $automate(\psi)$)

Algorithme de complexité non élémentaire.

■

7.2.6 Arithmétique de Peano ([DNRC01], p. 111)

$$PA \begin{cases} \text{Arithmétique de Presburger} \\ \forall x, x \times 0 = x \\ \forall x, y, x \times (y^{+1}) = (x \times y) + x \end{cases}$$

Proposition 118 $PA \models \varphi$ *implique* $(\mathbb{N}, =, 0, 1, +, \times) \models \varphi$.

7.2.7 Arithmétique vraie sur $\mathbb{N}$

Soit $\Sigma = \{0, 1, +, \times\}$.
$T_{\mathbb{N}} = \{\varphi \mid \varphi \text{ est une } \Sigma\text{-formule et } (N, =, 0, 1, +, \times) \models \varphi\}$.

Proposition 119 $T_{\mathbb{N}} \models \varphi$ *ssi* $(\mathbb{N}, =, 0, 1, +, \times) \models \varphi$.

7.3 Panorama

	Théorie complète ?	SAT	SAT d'une formule existentielle	SAT d'une conjonction existentielle
$\emptyset$	✗	indéc, dans co-RE	?	?
Théorie de l'égalité	✗	indéc, dans co-RE	?	?
Théorie de l'égalité (avec que des symboles de fonction)	✗	indéc [? ?]	NP-complet	dans P [KS08]
Théorie de l'égalité (sans autres symboles de fonction et symboles de prédicats) ([DNRC01], p. 132)	✗	PSPACE-complet [SM73]	NP-complet	dans P [KS08]
Théorie des ordres denses ([DNRC01], p. 130) ([Har09], p. 333)	✓	PSPACE-complet	NP-complet	dans P
Théorie des corps clos ([DNRC01], p. 133)	✓	dans EXPSPACE [BOKR86], PSPACE-dur	NP-dur dans PSPACE [Can88]	?
Arithmétique linéaire sur $\mathbb{R}$	✓		NP-complet	dans P
Arithmétique linéaire sur $\mathbb{N}$ (de Presburger) ([DNRC01], p. 136) ([Har09], p. 336) ([Car08])	✓	dans 3EXPTIME [Opp78], 2EXPTIME-dur [FFR74]	NP-complet	NP-complet
Arithmétique de Robinson ([CL93], p. 73)	✗	indéc, dans co-RE	?	?
Arithmétique de Peano ([DNRC01], p. 123)	✗	indéc, dans co-RE	indéc [?]	indéc [?]
Arithmétique sur $\mathbb{N}$ (théorie non récursive)	✓	indéc, ni dans RE, ni dans co-RE	indéc, dans RE	indéc[Mat03], dans RE

Chapitre 8

Déduction naturelle

Points du programme de l'agrégation
... substitutions, capture de variables.
Logique du premier ordre : systèmes formels de preuve. Déduction naturelle. Théorème de complétude du calcul des prédicats du premier ordre.

8.1 Substitution

Définition 120 (substitution ([BA12], p. 187))
Une **substitution** est un ensemble de la forme

$$[x_1 := t_1, \ldots, x_n := t_n]$$

où chaque x_i est une variable et chaque t_i est un terme.

Définition 121 (instantiation)
La formule $\varphi[x_1 := t_1, \ldots, x_n := t_n]$ est la formule φ dans laquelle on a remplacé simultanément les occurrences libres x_i par le terme t_i.

Attention, on autorise uniquement les **substitutions licites**. Par exemple

$$\underbrace{(\forall x, x < y)}_{\varphi}[y := \overbrace{x - 1}^{t}] \quad = \quad \forall x, x < x - 1$$

n'est pas licite. On parle de **capture de variables**. Sans perte de généralité, on renomme les variables liées de φ par des variables fraîches afin qu'elles n'apparaissent pas dans t ([DNRC01], p. 20-22) :

$$(\forall x', x' < y)[y := x - 1] \quad = \quad \forall x', x' < x - 1$$

8.2 Règles de la déduction naturelle ([DNRC01], p. 25)

Définition 122 (séquent en déduction naturelle ([DNRC01], p. 24))
Un **séquent** est un couple (Γ, φ) où Γ est un ensemble fini de formules et φ une formule. Il se note $\Gamma \vdash \varphi$ et se lit « à partir des hypothèses Γ, j'ai prouvé la formule φ ».

Définition 123 (règles de la déduction naturelle)
Les **règles de la déduction naturelle** sont données ci-dessous :

$$\text{axiome}\frac{}{\Gamma, A \vdash A} \qquad \text{affaiblissement}\frac{\Gamma \vdash A}{\Gamma, B \vdash A} \qquad \text{absurde}\frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}$$

	Règles d'introduction	Règles d'élimination
$\to$	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash (A \to B)}$	$\dfrac{\Gamma \vdash A \quad \Gamma \vdash (A \to B)}{\Gamma \vdash B}$
$\wedge$	$\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash (A \wedge B)}$	$\dfrac{\Gamma \vdash (A \wedge B)}{\Gamma \vdash A}$ $\dfrac{\Gamma \vdash (A \wedge B)}{\Gamma \vdash B}$
$\vee$	$\dfrac{\Gamma \vdash A}{\Gamma \vdash (A \vee B)}$ $\dfrac{\Gamma \vdash B}{\Gamma \vdash (A \vee B)}$	$\dfrac{\Gamma \vdash (A \vee B) \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}$
$\neg$	$\dfrac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}$	$\dfrac{\Gamma \vdash \neg A \quad \Gamma \vdash A}{\Gamma \vdash \bot}$
$\exists$	$\dfrac{\Gamma \vdash A[x := t]}{\Gamma \vdash \exists x A}$	$\dfrac{\Gamma \vdash \exists x A \quad \Gamma, A \vdash C}{\Gamma \vdash C}$ où x non libre dans Γ, C
$\forall$	$\dfrac{\Gamma \vdash A}{\Gamma \vdash \forall x A}$ où x non libre dans Γ	$\dfrac{\Gamma \vdash \forall x A(x)}{\Gamma \vdash A[x := t]}$
$=$	$\dfrac{}{\Gamma \vdash (t = t)}$	$\dfrac{\Gamma \vdash \varphi(x := t) \quad \Gamma \vdash (t = u)}{\Gamma \vdash \varphi(x := u)}$

Remarque 124 *Les règles $\neg$ particularisent les règles $\to$ avec $\neg A$ abbréviation de $A \to \bot$.*

Exemple 125 (règle d'introduction du $\forall$) *Soit Γ avec x non libre dans Γ.*

$$\cancel{\frac{p(x) \vdash p(x)}{p(x) \vdash \forall x, p(x)}}$$ 

$$\frac{\Gamma \vdash p(x)}{\Gamma \vdash \forall x, p(x)}$$ 

Exemple 126 (règle d'élimination du $\exists$) *Soit Γ avec x non libre dans Γ.*

$$\cancel{\frac{\Gamma, p(x) \vdash \exists x, q(x) \quad \Gamma, p(x), q(x) \vdash r(x)}{\Gamma, p(x) \vdash r(x)}}$$ 

$$\frac{\Gamma \vdash \exists x, p(x) \quad \Gamma, p(x) \vdash q}{\Gamma \vdash q}$$ 

8.3 Arbres de preuve

Définition 127 (arbre de preuve en déduction naturelle)
Une **(arbre de) preuve (en déduction naturelle)** de $\Gamma \vdash \varphi$ est un arbre fini tel que :
— La racine est $\Gamma \vdash \varphi$;
— Chaque nœud correspond à l'application d'une des règles de la déduction naturelle.

Exemple 128 *([DNRC01], p. 40)*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg\exists xA, A \vdash \neg\exists xA}\,ax \quad \dfrac{\dfrac{}{\neg\exists xA, A \vdash A}\,ax}{\neg\exists xA, A \vdash \exists xA}\,\exists_i}{\neg\exists xA, A \vdash \bot}\,\neg_e}{\neg\exists xA \vdash \neg A}\,\neg_i}{\neg\exists xA \vdash \forall x\neg A}\,\forall_i}{\emptyset \vdash (\neg\exists xA) \to (\forall x\neg A)}\,\to_i$$

Exemple 129 *([DNRC01], p. 40) Présentation alternative avec* ***hypothèses déchargées*** *:*

$$\dfrac{\neg\exists xA \qquad \dfrac{A}{\exists xA}\,\exists_i}{\bot}\,\neg_e$$

$$\dfrac{\dfrac{\neg\exists xA \qquad \dfrac{\not{A}}{\exists xA}\,\exists_i}{\bot}\,\neg_e}{\neg A}\,\neg_i$$

$$\dfrac{\dfrac{\dfrac{\neg\exists xA \qquad \dfrac{\not{A}}{\exists xA}\,\exists_i}{\bot}\,\neg_e}{\neg A}\,\neg_i}{\forall x\neg A}\,\forall_i$$

$$\dfrac{\dfrac{\dfrac{\not{\neg\exists xA} \qquad \dfrac{\not{A}}{\exists xA}\,\exists_i}{\bot}\,\neg_e}{\neg A}\,\neg_i}{\forall x\neg A}\,\forall_i}{(\neg\exists xA) \to (\forall x\neg A)}\,\to_i$$

Exemple 130 *Présentation* ***à la Fitch*** *([DLL12], p. 70) :*

1.	*supposons* $\neg\exists xA$		
2.	*supposons* A		
3.		$\exists xA$	$\exists_i$ *sur 2.*
4.		$\bot$	$\neg_e$ *sur 1. et 3.*
5.	$\neg A$		$\neg_i$ *sur 5. avec décharge de l'hypothèse 2.*
6.	$\forall x\neg A$		$\forall_i$ *sur 5.*
7.	$(\neg\exists xA) \to (\forall x\neg A)$		$\to_i$ *sur 6. avec décharge de l'hypothèse 1.*

Définition 131 (existence d'un arbre de preuve ([DNRC01], p. 80))
Soit T une théorie. Soit φ une formule close. On note $T \vdash_{DN} \varphi$ s'**il**[1] **existe un arbre de preuve** de $\Gamma \vdash \varphi$ avec Γ fini et $\Gamma \subseteq T$.

Définition 132 (théorie consistante)
Soit T une théorie. T est **consistante** si $T \nvdash_{DN} \bot$.

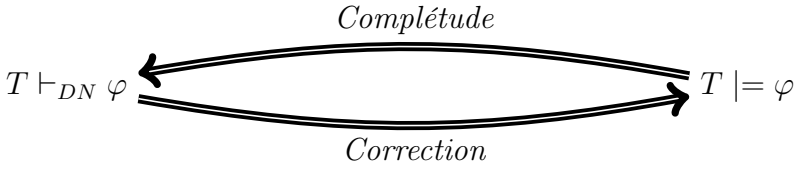
Définition 133 (théorie syntaxiquement complète)
Soit T une théorie. T est **syntaxiquement complète** si
pour toute formule close φ, $T \vdash_{DN} \varphi$ ou $T \vdash_{DN} \neg\varphi$.

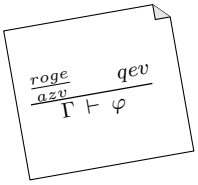
1. Certains livres surchargent la notation et écrivent $\vdash$.

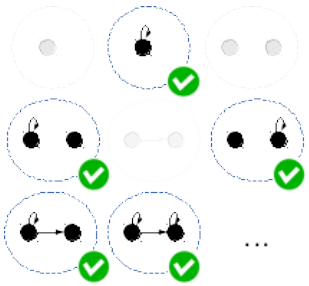
8.4 Correction et complétude

8.4.1 Énoncé

Théorème 134 *Soit T un ensemble de formules closes et φ une formule close.*







Idée de la démonstration.

$\boxed{\Rightarrow}$ On démontre, par induction sur π, que pour tout arbre de preuve π, on a

$\mathcal{P}(\pi)$: Si π est une preuve de $\Gamma \vdash \varphi$, alors
pour tout modèle $\mathcal{M}$, pour tout assignation v, $\mathcal{M}, v \models \Gamma$ implique que $\mathcal{M}, v \models \varphi$.

$\boxed{\Leftarrow}$

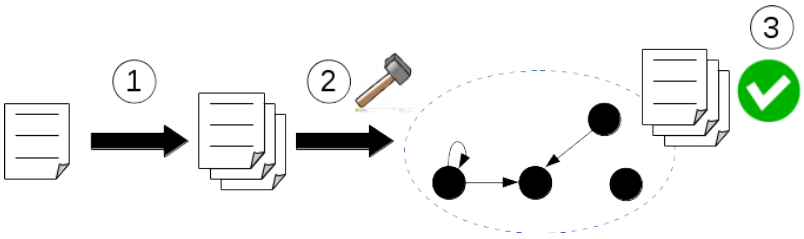
$T \models \varphi$	ssi	$T \cup \{\neg\varphi\}$ insatisfiable	
	implique	$T \cup \{\neg\varphi\}$ inconsistante	(par le lemme 135)
	ssi	$T \vdash_{DN} \varphi$	

■

8.4.2 Idée de la démonstration

Lemme 135 *([DNRC01], p. 83) Δ est consistante alors Δ est satisfiable.*

Idée de la démonstration.



1. On étend Δ en Δ' tel que :
   — Pour toute formule $\psi(x)$, on introduit un symbole de constante c_ψ tel que

$$\Delta' \vdash_{DN} (\exists x \psi(x) \to \psi(c_\psi)).$$

   Un tel c_ψ s'appelle un **témoin de Henkin**.
   — Δ' soit une théorie syntaxiquement complète.

2. On construit le modèle $\mathcal{M}$:
   — Le domaine D est l'ensemble des termes clos [2] ;
   — Si f est un symbole de fonction d'arité n,

$$\begin{array}{rlcl} f^{\mathcal{M}} : & D^n & \to & D \\ & (t_1, \ldots, t_n) & \mapsto & f(t_1, \ldots, t_n) \end{array}$$

   — Si p est un symbole de prédicats n-aire,

$$\begin{array}{rlcl} p^{\mathcal{M}} : & D^n & \to & \{0,1\} \\ & (t_1, \ldots, t_n) & \mapsto & 1 \text{ si } \Delta' \vdash_{DN} p(t_1, \ldots, t_n) \\ & & \mapsto & 0 \text{ sinon.} \end{array}$$

3. On montre :

Lemme 136 (lemme de la vérité)
Soit $\varphi(x_1, \ldots, x_n)$ une formule et $t_1, \ldots t_n$ des termes clos. On a :

$$\mathcal{M}[x_1 := t_1, \ldots, x_n := t_n] \models \varphi \text{ ssi } \Delta' \vdash_{DN} \varphi(t_1, \ldots, t_n)$$

■

8.5 Conséquences

8.5.1 Problème de validité dans RE

Théorème 137 *Le problème de validité en logique du premier ordre est dans RE.*

Idée de la démonstration.
Voici une machine qui accepte le langage des formules valides :

```
procédure valide(φ)
    pour toute preuve π
        si π est une preuve de ⊢ φ alors
            accepter
```

■

Théorème 138 *Si $T \in RE$, la T-validité en logique du premier ordre est dans RE.*

Idée de la démonstration.
Soit $\psi_1, \psi_2, \ldots$ une énumération effective de T. Voici une machine qui accepte le langage des formules T-valides :

```
procédure Tvalide(φ)
    pour tout n = 0, 1, ...
        énumérer les formules ψ_1, ..., ψ_n
        pour toute preuve π de longueur au plus n
            si π est une preuve de {ψ_1, ..., ψ_n} ⊢ φ alors
                accepter
```

■

2. Certaines personnes ([DNRC01], p. 83) quotientent l'ensemble des termes clos avec la relation d'équivalence $\sim$ défini par $t \sim t'$ ssi $\Delta' \vdash (t = t')$ car le symbole de prédicat $=$ s'interprète par l'égalité. Aussi ils ont rajouté les règles correspondantes à $=$.

8.5.2 Théorème de compacité

Théorème 139 (Compacité de la logique du premier ordre ([DNRC01], p. 86))
Soit T un ensemble de formules closes.

Si toute partie finie de T est satisfiable alors T satisfiable.

Idée de la démonstration.
Par contraposée.

T insatisfiable **implique** T inconsistante (par le lemme 135)
implique il existe une preuve de $\Gamma \vdash \bot$ avec Γ fini et $\Gamma \subseteq T$
implique il existe Γ fini et $\Gamma \subseteq T$ et Γ insatisfiable (par correction).

■

Corollaire 140 *Il n'existe pas de formule close φ de la logique du premier ordre telle que*

$$\mathcal{M} \models \varphi \text{ ssi le domaine de } \mathcal{M} \text{ est fini.}$$

Idée de la démonstration.
Par l'absurde, supposons qu'il existe une telle formule φ et posons

$$T = \{\varphi\} \cup \bigcup_{n\in\mathbb{N}} \{\exists x_1 \dots x_n, \bigwedge_{i,j\in\{1,\dots n\}, i\neq j} x_i \neq x_j\}.$$

Toute partie finie de T est satisfiable mais T est insatisfiable.
Contradiction avec le théorème de compacité. ■

Corollaire 141 (existence de modèles non standards de l'arithmétique vraie)
([DNRC01], p. 112, le Th. 3.4.6 donne un résultat plus faible avec l'arithmétique de Peano)
([BJ87], p. 150, Problem 12.9)
$T_{\mathbb{N}} = \{\varphi \mid (\mathbb{N}, =, 0, 1, +, \times) \models \varphi\}$ admet un modèle non isomorphe à $(\mathbb{N}, =, 0, 1, +, \times)$.

Idée de la démonstration.
Soit c un symbole frais de constante. Soit $T = T_{\mathbb{N}} \cup \{i < c \mid i \in \mathbb{N}\}$. Toute partie finie de T est satisfiable. D'après le **théorème de compacité**, T est satisfiable. ■

8.5.3 Théorème de Lowenheim-Skolem

Théorème 142 (Lowenheim-Skolem ([DNRC01], p. 99) ([BA12] , p. 228))
Rappelons que l'on travaille avec une signature Σ dénombrable.

Si T admet un modèle infini alors T admet un modèle infini dénombrable.

Idée de la démonstration.
Soit $T' = T \cup \{c_i \neq c_j \mid i, j \in \mathbb{N}, i \neq j\}$ où $c_0, c_1, \dots$ sont des nouveaux symboles de constante.

T' satisfiable implique T' consistante
implique T' est satisfiable dans le modèle à l'étape 2
de la démonstration du lemme 135.

■

Corollaire 143 (modèle dénombrable pour la théorie des corps clos)
La théorie des corps clos admet un modèle infini dénombrable.

8.6 Logique minimale, logique intuitionniste

Définition 144 (logique classique, [DNRC01], p. 147)
La **logique classique** (LK pour 'klassische Prädikatenlogik') est la logique en utilisant toutes les règles de la Déf. 123.

Définition 145 (logique minimale, [DNRC01], p. 147)
La **logique minimale** (LM) s'obtient en utilisant toutes les règles sauf l'absurde.

Définition 146 (logique intuitionniste, [DNRC01], p. 147)
La **logique intuitionniste** (LJ pour 'intuitionnistisch' et le J était le i majuscule en vieil allemand [3]) est la logique obtenue en utilisant toutes les règles mais en remplaçant l'absurde par **ex falso sequitur quod libet** :

$$\text{ex falso sequitur quod libet}\frac{\Gamma \vdash \bot}{\Gamma \vdash A}$$

Définition 147 (tiers exclu)

$$\text{tiers exclu}\frac{\Gamma, A \vdash B \qquad \Gamma, \neg A \vdash B}{\Gamma \vdash B}$$

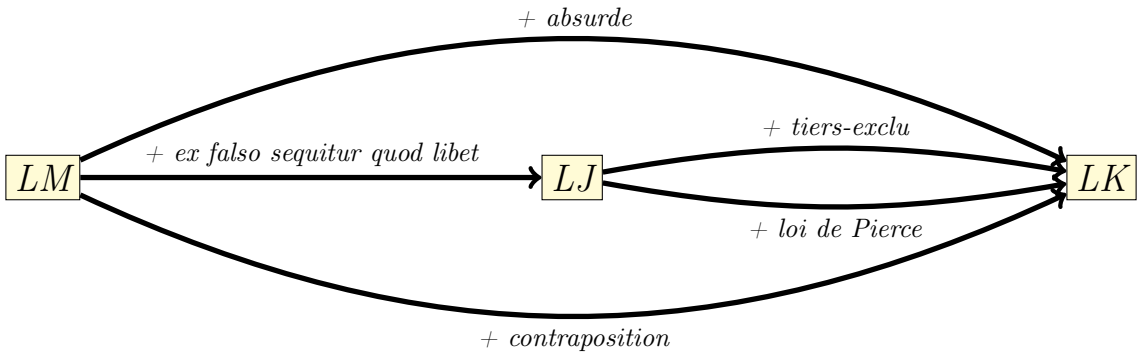
Définition 148 (contraposition)

$$\text{contraposition}\frac{\Gamma \vdash \neg B \to \neg A}{\Gamma \vdash A \to B}$$

Définition 149 (loi de Pierce)

$$\text{loi de Pierce}\frac{\Gamma, \neg A \vdash A}{\Gamma \vdash A}$$

Théorème 150 *([DNRC01], p. 148)*



Remarque 151 *([DNRC01], p. 160 et 169) On peut démontrer un théorème de complétude pour LJ avec des* ***modèles de Kripke****.*

3. https://plato.stanford.edu/entries/proof-theory-development/

Chapitre 9

Calcul des séquents

Points du programme de l'agrégation
Calcul des séquents.

9.1 Motivation

— Règles du calcul des séquents plus symétriques que celles de la déduction naturelle ([DNRC01], p. 58).

éliminable ↓

— La créativité humaine est confinée dans les substitutions et la règle de coupure :
  — Propriété de la sous-formule. D'où la décidabilité de la logique propositionnelle ;
  — Calcul effectif d'un **interpolant** à partir d'un arbre de preuve ([DNRC01], p. 216).

9.2 Règles du calcul des séquents ([DNRC01], p. 187)

Définition 152 (séquent en calcul des séquents ([DNRC01], p. 186))
Un **séquent** est un couple (Γ, Δ) où Γ et Δ sont des multi-ensembles[1] finis de formules.
Il se note $\Gamma \vdash \Delta$ et se lit

« à partir de la conjonction des hypothèses Γ, j'ai prouvé la disjonction des formules de Δ ».

1. Il existe aussi des présentations avec des ensembles, cf. [BA12], p. 70.

Définition 153 (règles du calcul des séquents)
Les **règles du calcul des séquents** sont données ci-dessous :

$$\bot_g \frac{}{\bot \vdash} \qquad \text{axiome} \frac{}{A \vdash A}$$

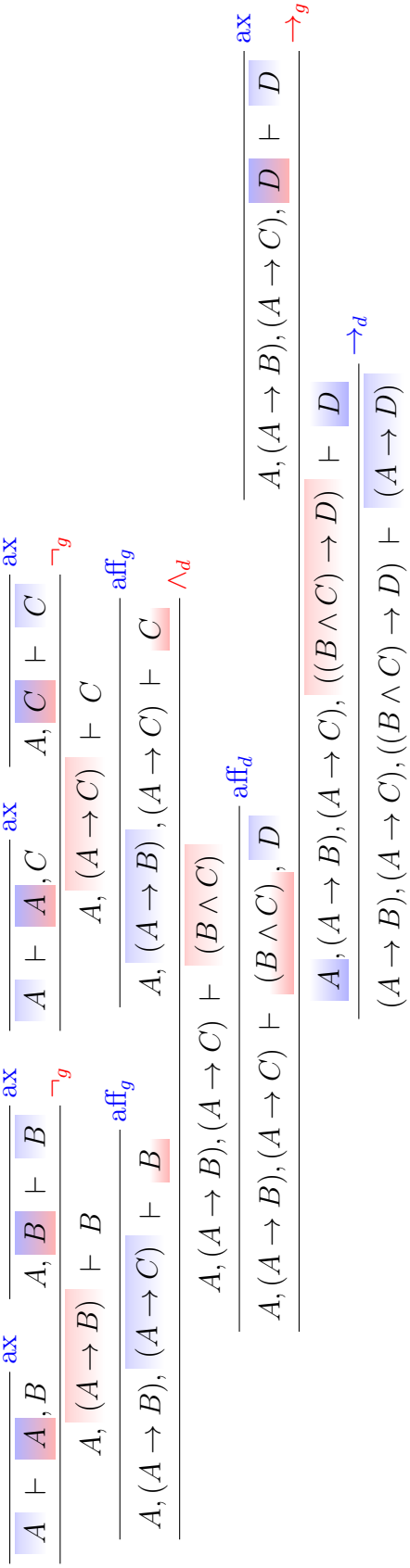
	Règles d'introduction à gauche	Règles d'introduction à droite
affaiblissement	$\dfrac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}$	$\dfrac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}$
contraction	$\dfrac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}$	$\dfrac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}$
$\neg$	$\dfrac{\Gamma \vdash \Delta, A}{\Gamma, \neg A \vdash \Delta}$	$\dfrac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}$
$\wedge$	$\dfrac{\Gamma, A, B \vdash \Delta}{\Gamma, (A \wedge B) \vdash \Delta}$	$\dfrac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash (A \wedge B), \Delta}$
$\vee$	$\dfrac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, (A \vee B) \vdash \Delta}$	$\dfrac{\Gamma, \vdash \Delta, A, B}{\Gamma \vdash \Delta, (A \vee B)}$
$\rightarrow$	$\dfrac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, (A \rightarrow B) \vdash \Delta}$	$\dfrac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash (A \rightarrow B), \Delta}$
$\exists$	$\dfrac{\Gamma, A \vdash \Delta}{\Gamma, \exists x A \vdash \Delta}$ où x non libre dans Γ, Δ	$\dfrac{\Gamma \vdash A[x := t], \Delta}{\Gamma \vdash \exists x A, \Delta}$
$\forall$	$\dfrac{\Gamma, A[x := t] \vdash \Delta}{\Gamma, \forall x A \vdash \Delta}$	$\dfrac{\Gamma \vdash A, \Delta}{\Gamma \vdash \forall x A, \Delta}$ où x non libre dans Γ, Δ

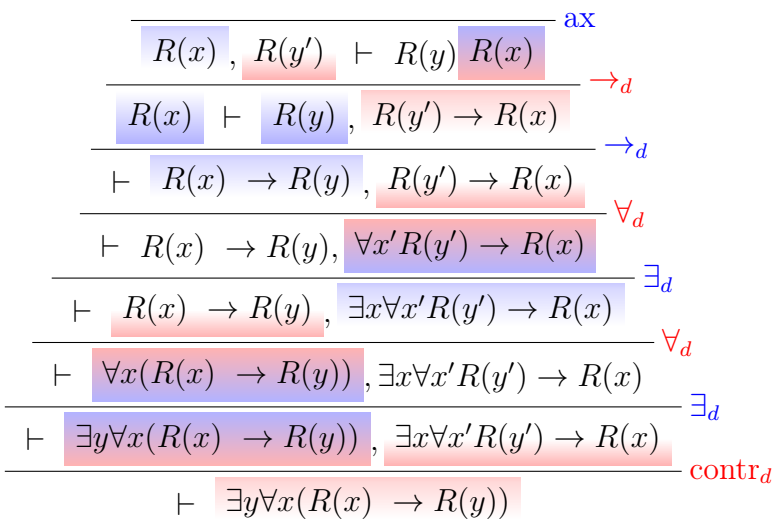
éliminable

$$\text{coupure} \frac{\Gamma \vdash A, \Delta \qquad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}$$

Tutoriel : `http://logitext.mit.edu/tutorial`

9.3 Exemple d'arbres de preuve ([DNRC01], p. 190)





9.4 Correction et complétude

Théorème 154

$\Gamma \vdash \varphi$ *est prouvable en déduction naturelle* *ssi* $\Gamma \vdash \varphi$ *est prouvable en calcul des séquents.*

IDÉE DE LA DÉMONSTRATION.

(⇒) ([DNRC01], p. 195) On transforme un arbre de preuve en déduction naturelle

$$\dfrac{\dfrac{}{\neg\neg A, \neg A \vdash \neg\neg A}\,\text{ax} \qquad \dfrac{}{\neg\neg A, \neg A \vdash \neg A}\,\text{ax}}{\dfrac{\neg\neg A, \neg A \vdash \bot}{\neg\neg A \vdash A}\,\textit{absurde}}\,\neg_e$$

en un arbre de preuve en calcul des séquents :

$$\dfrac{\dfrac{\dfrac{}{\neg\neg A, \neg A \vdash \neg\neg A}\,\text{ax} \qquad \dfrac{\dfrac{}{\neg\neg A, \neg A \vdash \neg A}\,\text{ax}}{\neg\neg A \vdash}\,\neg_e}{\neg\neg A, \neg A \vdash \bot}\,\textit{coupure} \qquad \dfrac{}{\bot \vdash}\,\bot_g}{\neg\neg A, \neg A \vdash}\,\textit{coupure} \qquad \dfrac{\dfrac{}{A \vdash A}\,\text{ax}}{\vdash \neg A, A}\,\neg_r}{\neg\neg A \vdash A}\,\textit{coupure}$$

(⇐) On transforme un arbre de preuve du calcul des séquents en un arbre de la déduction naturelle ([DNRC01], p. 197).

■

9.5 Élimination de la règle de la coupure

Théorème 155

$\Gamma \vdash \varphi$ *est prouvable en calcul des séquents* *ssi* $\Gamma \vdash \varphi$ *est prouvable en calcul des séquents sans utiliser la règle de la coupure.*

IDÉE DE LA DÉMONSTRATION.
(⇐) Immédiat.
(⇒) Voir ([DNRC01], p. 200-208)
■

9.6 Logique intuitionniste

La logique intuitionniste s'obtient en n'autorisant que les séquents $\Gamma \vdash \Delta$ où $\mathbf{card}(\Delta) \leq 1$ (cf. [DNRC01], p. 192-193).

Chapitre 10

Unification

Points du programme de l'agrégation
Algorithme d'unification des termes.

10.1 Motivation : résolution

En logique propositionnelle

$$\frac{p \vee q \qquad \neg p \vee r}{q \vee r}$$

En logique du premier ordre (cf. Chapitre 11)

$$\frac{p(a,x) \vee q(x) \qquad \neg p(y,b) \vee r(y)}{q(a) \vee r(b)}$$

10.2 Définitions

Références : [LDR93], p. 86 ; [BA12], p. 189 ; [BN98], p. 71

Définition 156 (problème d'unification)
Un **problème d'unification** est un ensemble fini $E = \{(s_1, t_1), \ldots, (s_n, t_n)\}$ où les $s_1, \ldots, s_n$ et les $t_1, \ldots, t_n$ sont des termes. Il est noté $\{s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\}$.

Exemple 157 $\{x \stackrel{?}{=} b, y \stackrel{?}{=} a\}$.

Exemple 158 $\{f(x) \stackrel{?}{=} f(f(a)), \quad g(y) \stackrel{?}{=} g(z)\}$.

Définition 159 (unificateur)
Un **unificateur** de $\{s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\}$ est une substitution σ t.q. pour tout $i \in \{1, \ldots, n\}$, on a $s_1\sigma = \ldots = t_i\sigma$.

Exemple 160 $\sigma := [x \mapsto f(a), z \mapsto y]$ *est un unificateur de* $\{f(x) \stackrel{?}{=} f(f(a)), \quad g(y) \stackrel{?}{=} g(z)\}$ *car*

- $f(x)\sigma = f(f(a))$ *et* $f(f(a))\sigma = f(f(a))$;
- $g(y)\sigma = g(y)$ *et* $g(z)\sigma = g(y)$.

Exemple 161 $\alpha := [x \mapsto f(a), y \mapsto h(b)), z \mapsto h(b))]$ *est un unificateur de* $\{f(x) \stackrel{?}{=} f(f(a)), \quad g(y) \stackrel{?}{=} g(z)\}$ *car*

- $f(x)\alpha = f(f(a))$ *et* $f(f(a))\alpha = f(f(a))$;
- $g(y)\alpha = g(h(b))$ *et* $g(z)\alpha = g(h(b))$.

Définition 162 (unifiable)
E est **unifiable** s'il admet un unificateur.

Exemple 163 $\{f(x) \stackrel{?}{=} f(f(a)),\quad g(y) \stackrel{?}{=} g(z)\}$ *est unifiable.*

Exemple 164 $\{f(a) = g(a)\}$ *n'est pas unifiable.*

Exemple 165 $\{x = f(x)\}$ *n'est pas unifiable.*

Définition 166 (unificateur principal, most general unifier ou mgu)
Un **unificateur principal** d'un problème d'unification E est un unificateur σ de E tel que pour tout unificateur α de E, il existe une substitution β telle que $\alpha = \sigma\beta$.

Exemple 167 $\sigma := [x \mapsto f(a), z \mapsto y]$ *est un unificateur principal de* $\{f(x) \stackrel{?}{=} f(f(a)),\quad g(y) \stackrel{?}{=} g(z)\}$. *En particulier,* $\alpha := [x \mapsto f(a), y \mapsto h(b)), z \mapsto h(b))]$ *en est un unificateur et on a :*

$$[x \mapsto f(a), y \mapsto h(b)), z \mapsto h(b))] = [x \mapsto f(a), z \mapsto y][y \mapsto h(b)]$$

Théorème 168 *Un problème d'unification unifiable admet un unificateur principal.*

Idée de la démonstration.
Cf. algorithme de la section suivante. Voir Th. 169 et 176 ■

10.3 Algorithme d'unification

Référence : [BN98], p. 74

10.3.1 Description de l'algorithme

Règles de réécriture du problème d'unification

				nb variables non traitées	taille du problème d'unification	nb équations de la forme $t = x$
Supprimer	$E \sqcup \{t \stackrel{?}{=} t\}$	$\Rightarrow$	E	$\geq$	$\searrow$	
Décomposer	$E \sqcup \{f(\vec{t}) \stackrel{?}{=} f(\vec{u})\}$	$\Rightarrow$	$E \cup \{t_1 \stackrel{?}{=} u_1, \ldots, t_n \stackrel{?}{=} u_n\}$	$\geq$	$\searrow$	
Échanger	$E \sqcup \{f(\vec{t}) \stackrel{?}{=} x\}$	$\Rightarrow$	$E \cup \{x \stackrel{?}{=} f(\vec{t})\}$	$\geq$	$=$	$\searrow$
Éliminer	$E \sqcup \{x \stackrel{?}{=} t\}$ si x n'apparaît pas dans t	$\Rightarrow$	$E[x \mapsto t] \cup \{x \stackrel{?}{=} t\}$	$\searrow$		
Conflit	$E \sqcup \{f(\vec{t}) \stackrel{?}{=} g(\vec{u})\}$ avec $f \neq g$	$\Rightarrow$	✖			
Vérifier	$E \sqcup \{x \stackrel{?}{=} t\}$ si x apparaît dans t	$\Rightarrow$	✖			

fonction unification(E)
Tant que c'est possible et que cela change E,
appliquer une règle de réécriture sur E
retourner E

10.3.2 Terminaison

Théorème 169 *L'algorithme termine.*

Idée de la démonstration.
Les valeurs prises par
— nb variables non traitées de E = nb de variables x qui ont une occurrence apparaissant en partie droite ;
— taille de $E = \{t_1 = u_1, \ldots, t_n = u_n\} = \sum_{i=1}^{n} |t_i| + |u_i|$;
— nb d'équations de la forme $t = x$ (i.e. une variable en partie droite)
forment une suite de $\mathbb{N}^3$ strictement décroissante pour l'**ordre lexicographique** sur $\mathbb{N}^3$, qui est un **ordre bien fondé**. ■

10.3.3 Correction

Notation 170 *$\mathcal{U}(E)$ = ensemble des unificateurs de E.* $\mathcal{U}(\text{✖}) = \emptyset$.

Proposition 171 *Si $E \Rightarrow E'$, alors $\mathcal{U}(E) = \mathcal{U}(E')$.*

Idée de la démonstration.
Règle Éliminer.

$\sigma \in \mathcal{U}(E \sqcup \{x \stackrel{?}{=} t\})$ ssi.pour tout $(s,u) \in E$, $s\sigma = u\sigma$ et $\sigma(x) = t$
ssi pour tout $(s,u) \in E$, $s[x \mapsto t]\sigma = u[x \mapsto t]\sigma$ et $\sigma(x) = t$
(faire un dessin pour s'en convaincre)
ssi $\sigma \in \mathcal{U}(E[x \mapsto t] \cup \{x \stackrel{?}{=} t\})$

■

Définition 172 (problème d'unification en forme résolue)
Un **problème d'unification en forme résolue** est un problème d'unification de la forme

$$\{x_1 \stackrel{?}{=} t_1, \ldots, x_n \stackrel{?}{=} t_n\}$$

où $x_1, \ldots, x_n$ sont des variables qui n'apparaissent dans aucun des t_j.

Notation 173 *Etant donné un problème d'unification en forme résolue $E := \{x_1 \stackrel{?}{=} t_1, \ldots, x_n \stackrel{?}{=} t_n\}$, on note $u(E)$ la substitution*

$$[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n].$$

Proposition 174 *L'algorithme retourne soit ✖ ou alors un problème d'unification en forme résolue.*

Idée de la démonstration.
Par l'absurde. Sinon, on peut appliquer une règle. En particulier, si x_i apparaît dans un t_j, on pourrait appliquer Supprimer ou Éliminer. ■

Proposition 175 *Si E est une forme résolue, alors $u(E)$ est un unificateur principal de E.*

Idée de la démonstration.
En notant $E = \{x_1 \stackrel{?}{=} t_1, \ldots, x_n \stackrel{?}{=} t_n\}$, on a pour tout $i \in \{1, \ldots, n\}$, $u(E)(x_i) = t_i$ et $t_i u(E) = t_i$ car les variables x_i n'apparaissent pas dans t_i.
De plus, si α unifie E, $\alpha(x_i) = t_i\alpha = u(E)\alpha(x_i)$. ■

Théorème 176 *unification(E) retourne :*
— *(un problème en forme résolue qui représente) un unificateur principal si E est unifiable ;*
— *✖, sinon.*

Idée de la démonstration.
Si E est unifiable, comme l'ensemble des unificateurs sont conservées, l'algorithme retourne un problème d'unification E' en forme résolue, et $u(E')$ est un unificateur principal.
■

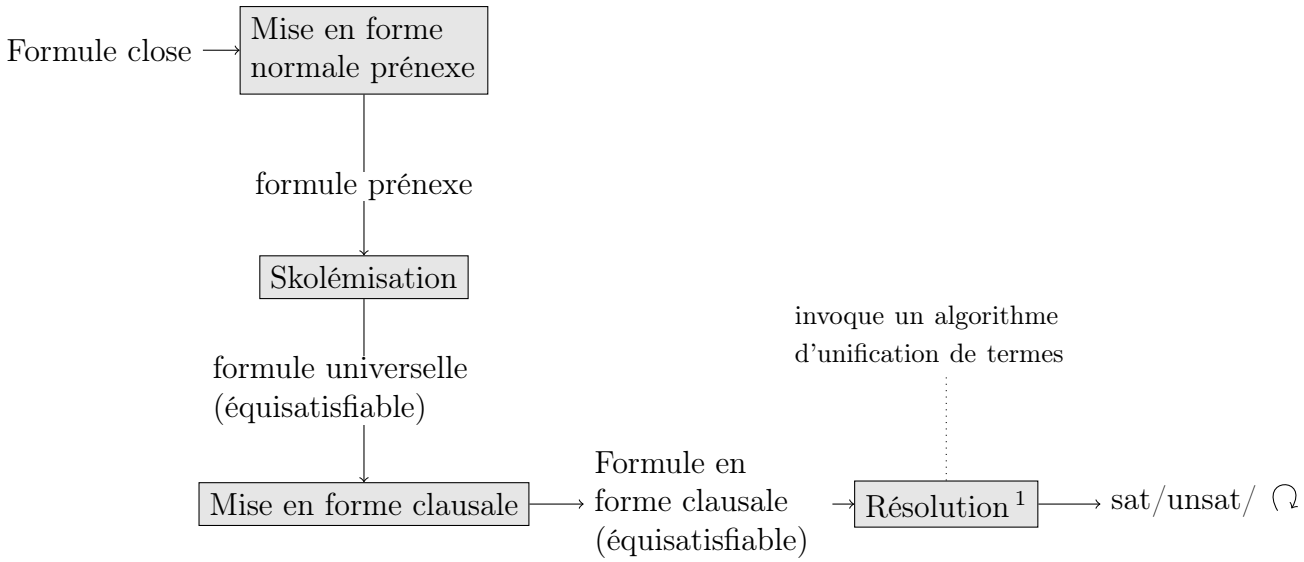
Remarque 177 *L'unificateur principal $\sigma = [x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$ retournée par l'algorithme est **idempotent**, i.e. $\sigma = \sigma\sigma$. C'est équivalent au fait que les x_i n'apparaîssent pas dans les t_j.*

Chapitre 11

Résolution en logique du premier ordre

Points du programme de l'agrégation
Preuves par résolution.

([BA12], p. 174) ([DNRC01], p. 268)



11.1 Mise en forme normale prénexe

Définition 178 (formule prénexe)
Une **formule prénexe** est une formule close de la formule de la forme $Q_1 x_1 \dots Q_n x_n \psi$ où chaque Q_i est un quantificateur et ψ est sans quantificateur.

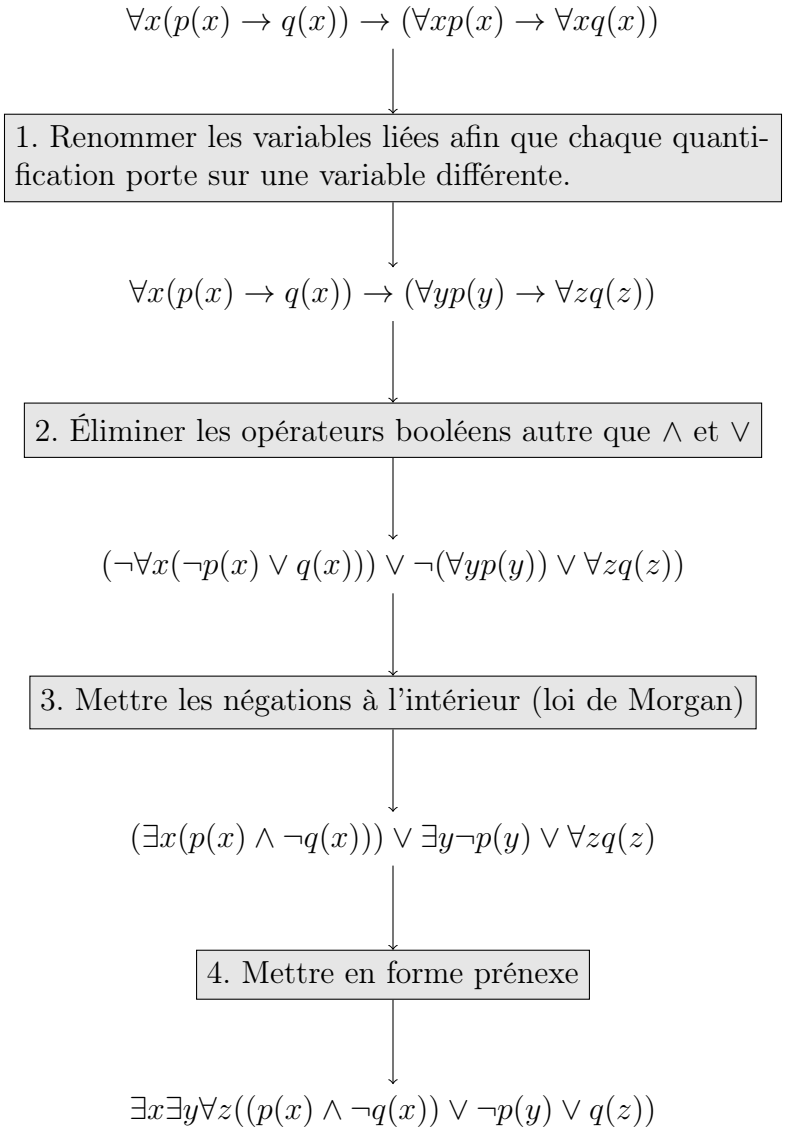
Définition 179 (préfixe et matrice)

préfixe $Q_1 x_1 \dots Q_n x_n$ ψ matrice

1. Attention, ce n'est pas un algorithme : on ne garantit pas la terminaison.

Proposition 180 ([DNRC01], p. 88, [BA12], p. 172)
Toute formule est équivalente à une formule prénexe.

Idée de la démonstration.
[([BA12], p. 174)]



■

11.2 Skolémisation ([BA12], p. 174) ([DNRC01], p. 89)

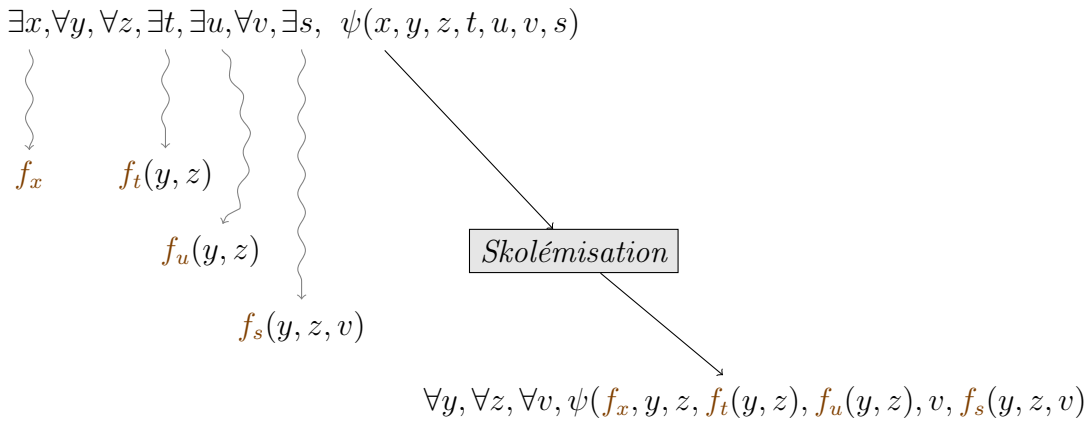
Définition 181 (formule prénexe universelle)
Une **formule universelle** est une formule close $\forall_1 x_1 \dots \forall_n x_n \psi$ où ψ est sans quantificateur.

Définition 182 (fonctions de Skolem, [DNRC01], p. 89-90)
Soit φ une formule prénexe.

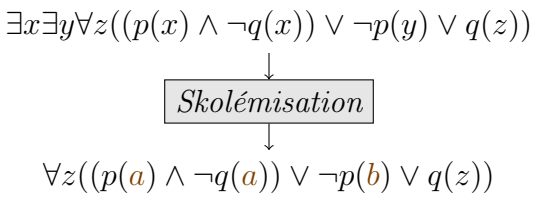
On appelle **fonction de Skolem** un symbole de fonction f_x d'arité n pour un quantificateur $\exists x$ qui dépend des n variables quantifiées universellement qui précède $\exists x$ dans φ.

Définition 183 (forme de Skolem, [DNRC01], p. 90)
La **forme de Skolem** d'une formule prénexe φ est la formule universelle obtenue en supprimant les $\exists x$ du préfixe de φ et en remplaçant chaque occurrence de x par le terme $f_x(\vec{y})$ où $\vec{y}$ sont les variables quantifiées universellement à gauche de $\exists x$.

Exemple 184



Exemple 185



Théorème 186 (de Skolem ([BA12], p. 172))
Une formule prénexe φ et sa forme de Skolem sont équisatisfiable. *Rem :* $\models formeSkolem(\varphi) \to \varphi$

11.3 Mise en forme clausale

Définition 187 (littéral, [DNRC01], p. 264)
Un **littéral** est une formule atomique ou la négation d'une formule atomique.

Définition 188 (forme normale conjonctive, [DNRC01], p. 87)
Une **forme normale conjonctive** est une conjonction de disjonctions de littéraux.

Proposition 189 ([DNRC01], p. 88)
Toute formule sans quantificateur est équivalente à une forme normale conjonctive.

Exemple 190

$$\forall z((p(a) \wedge \neg q(a)) \vee \neg p(b) \vee q(z))$$

$$\downarrow$$

Mise en forme clausale

$$\downarrow$$

$$\forall z((p(a) \vee \neg p(b) \vee q(z)) \wedge (\neg q(a) \vee \neg p(b) \vee q(z)))$$

11.4 Règles

Définition 191 (règle de résolution, [DNRC01], p. 265)

$$\text{res}\ \frac{(\ \psi \vee \ell_1 \vee \dots \ell_n\ ) \qquad (\ \neg\psi' \vee \ell'_1 \vee \dots \vee \ell'_k\ )}{(\ \ell_1\sigma \vee \dots \ell_n\sigma \vee \ell'_1\sigma \vee \dots \vee \ell'_k\sigma\ )}$$

où :
— ψ, $\neg\psi'$, $\ell_1, \dots, \ell_n, \ell'_1, \dots \ell'_k$ sont des littéraux ;
— les clauses $(\ \psi \vee \ell_1 \vee \dots \ell_n\ )$ et $(\ \neg\psi' \vee \ell'_1 \vee \dots \vee \ell'_k\ )$ n'ont pas de variables communes ; i.e. il faut renommer les variables avant (cf. Remarque p. 265 [DNRC01])
— ψ et ψ' sont unifiables et σ en est un unificateur principal.

Exemple 192 ([DNRC01], p. 265)

$$\textit{res}\ \frac{S(a, f(y)) \vee T(y) \qquad \neg S(x, f(x)) \vee R(x)}{T(a) \vee R(a)}$$

avec l'unificateur principal $[x := a, y := a]$.

Définition 193 (règle de contraction, [DNRC01], p. 265)

$$\text{contr}\ \frac{(\ \psi \vee \psi' \vee \ell_1 \vee \dots \ell_n\ )}{(\ \psi\sigma \vee \ell_1\sigma \vee \dots \ell_n\sigma\ )}$$

où :
— $\psi, \psi', \ell_1, \dots, \ell_n$ sont des littéraux ;
— ψ et ψ' sont unifiables et σ en est un unificateur principal.

Exemple 194 ([DNRC01], p. 265)

$$\textit{contr}\ \frac{S(y, f(x)) \vee S(a, f(y)) \ R(x)}{S(a, f(a)) \vee R(a)}$$

avec l'unificateur principal $[x := a, y := a]$.

Remarque 195 *Certains auteurs ([BA12], p. 196) ([LDR93], p. 93) présentent une seule règle de résolution qui fusionne la règle de résolution ci-dessus et la règle de contraction :*

$$\textit{super-res}\ \frac{(\ \psi_1 \vee \dots \vee \psi_i \vee \ell_1 \vee \dots \ell_n\ ) \qquad (\ \neg\psi'_1 \vee \dots \vee \neg\psi'_j \vee \ell'_1 \vee \dots \vee \ell'_k\ )}{(\ \ell_1\sigma \vee \dots \ell_n\sigma \vee \ell'_1\sigma \vee \dots \vee \ell'_k\sigma\ )}$$

si $\{\psi_1, \dots, \psi_i, \psi'_1, \dots \psi'_j\}$ *sont unifiables et où* σ *en est un unificateur principal.*

Exemple 196 *([BA12], p. 196)*

$$\textit{super-res}\ \frac{p(x) \vee p(y) \qquad \neg p(x') \vee \neg p(y')}{\bot}$$

avec l'unificateur principal $[y := x, x' := x, y' := x]$.

11.5 Exemples d'arbres de preuve

Exemple 197 ([DNRC01], p. 265)

$$\dfrac{\dfrac{\dfrac{\neg R(z,z) \vee R(z,a) \qquad \neg R(u,f(y)) \vee R(u,y)}{R(f(y),a) \vee R(f(y),y)}\, res}{R(f(a),a)}\, contr \qquad \dfrac{\dfrac{\dfrac{\neg R(x,a) \vee R(x,x) \qquad \neg R(v,f(y)) \vee \neg R(v,y)}{\neg R(f(y),a) \vee \neg R(f(y),y)}\, res}{\neg R(f(a),a)}\, contr}{\bot}\, res$$

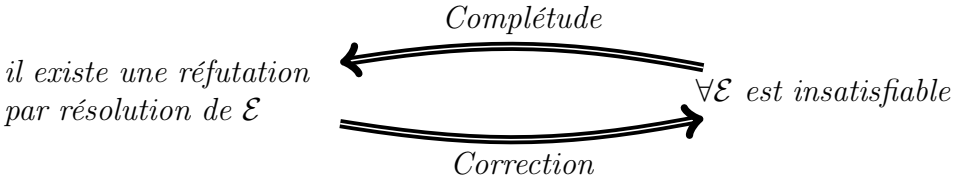
Exemple 198 ([BA12], p. 197-198)

1. $\neg p(x) \vee q(x) \vee r(x, f(x))$
2. $\neg p(x) \vee q(x) \vee r'(f(x))$
3. $p'(a)$
4. $p(a)$
5. $\neg r(a, y) \vee p'(y)$
6. $\neg p'(x) \vee \neg q(x)$
7. $\neg p'(x) \vee \neg r'(x)$
8. $\neg q(a)$ — *res(3, 6) avec* $[x := a]$
9. $q(a) \vee r'(f(a))$ — *res(2, 4) avec* $[x := a]$
10. $r'(f(a))$ — *res(8, 9) avec* $[]$
11. $q(a) \vee r(a, f(a))$ — *res(1, 4) avec* $[x := a]$
12. $r(a, f(a))$ — *res(8, 11) avec* $[]$
13. $p'(f(a))$ — *res(5, 12) avec* $[y := f(a)]$
14. $\neg r'(f(a))$ — *res(7, 13) avec* $[x := f(a)]$
15. $\bot$ — *res(10, 14) avec* $[]$

11.6 Correction et complétude

Notation 199 (clotûre universelle, [DNRC01], p. 21)
On note $\forall\varphi$ la formule close $\forall\vec{x}\varphi$ où $\vec{x}$ est une suite des variables libres dans φ.

Théorème 200 *Soit $\mathcal{E}$ une forme normale conjonctive.*



11.6.1 Démonstration de la correction

Idée de la démonstration.

$\boxed{\Rightarrow}$([CRK03], p. 270) ([BA12], p. 198) ([LDR93], p. 94-95)
Par contraposée, soit $\mathcal{M} \models \forall \mathcal{E}$. La démonstration est similaire au cas propositionnel :

Lemme 201 *si* $\mathcal{M} \models \forall C_1$, $\mathcal{M} \models \forall C_2$ *et* $res\dfrac{C_1 \quad C_2}{C}$ *alors* $\mathcal{M} \models \forall C$.

Lemme 202 *si* $\mathcal{M} \models \forall C_1$ *et* $contr\dfrac{C_1}{C}$ *alors* $\mathcal{M} \models \forall C$.

■

11.6.2 Modèles de Herbrand

Définition 203 (modèles de Herbrand)
Un **modèle de Herbrand** est un modèle $\mathcal{H} = (D, .^{\mathcal{H}})$ tel que :
— D = ensemble des termes clos ; (on suppose que l'on a au moins un symbole de constante)
— Pour tout symbole de fonction f d'arité n, on a :

$$\begin{array}{rccl} f^{\mathcal{H}} : & D^n & \to & D \\ & \vec{t} & \mapsto & f(\vec{t}). \end{array}$$

Proposition 204 ([LDR93], p. 99) *Soit φ une formule universelle.*

φ satisfiable ssi φ satisfiable dans un modèle de Herbrand.

Idée de la démonstration.
($\Leftarrow$) Immédiat. ($\Rightarrow$) Soit $\mathcal{M}$ un modèle tel que $\mathcal{M} \models \varphi$. On construit $\mathcal{H}$ avec pour tout symbole de prédicat p d'arité n :

$$\begin{array}{rccl} p^{\mathcal{H}} : & D^n & \to & \{0,1\} \\ & \vec{t} & \mapsto & \begin{cases} 1 & \text{si } \mathcal{M} \models p(\vec{t}) \\ 0 & \text{sinon} \end{cases} \end{array}$$

On démontre alors, par induction sur toute formule sans quantificateur $\psi(\vec{t})$, la propriété :

$$\mathcal{P}(\psi) : \mathcal{H}, [\vec{x} := \vec{t}] \models \psi \text{ ssi } \mathcal{M} \models \psi[\vec{x} := \vec{t}].$$

■

Définition 205 (théorie universelle)
Une théorie T est **universelle** si toutes les formules de T sont universelles.

Notation 206 *Soit T une théorie universelle. On note $Inst(T)$ l'ensemble des instances des formules de T obtenues par substitution de termes clos.*

Exemple 207 *Supposons qu'il n'y a qu'un seul symbole de constante c. $Inst(\{p(x)\} = \{p(c)\}$.*

Exemple 208 *Supposons qu'il y a deux symboles de constantes a et b.*
- $Inst(\{p(x)\} = \{p(a), p(b)\}$;
- $Inst(\{p(x) \wedge p(y)\} = \{p(a) \wedge p(a), p(a) \wedge p(b), p(b) \wedge p(a), p(b) \wedge p(b)\}$.

Exemple 209 *Supposons qu'il y a un symbole de constante c et un symbole de fonction f d'arité* 1.
- $Inst(\{p(x)\} = \{p(c), p(f(c), p(f^2(c))), \ldots\}$;
- $Inst(\{p(x) \wedge \neg p(f(y))\}) = \{ \begin{array}{l} p(c) \wedge \neg p(f(c)), \\ p(f(c)) \wedge \neg p(f(c)), \\ p(f(c)) \wedge \neg p(f^2(c)), \\ p(f^2(c)) \wedge \neg p(f^3(c)), \ldots\} \end{array}$.

On voit les formules closes comme des formules de la logique propositionnelle :

$$Inst(\{p(x) \wedge \neg p(f(y))\}) = \{ \begin{array}{l} \boxed{p(c)} \wedge \neg \boxed{p(f(c))}, \\ \boxed{p(f(c))} \wedge \neg \boxed{p(f(c))}, \\ \boxed{p(f(c))} \wedge \neg \boxed{p(f^2(c))}, \\ \boxed{p(f^2(c))} \wedge \neg \boxed{p(f^3(c))}, \ldots\} \end{array}$$

Théorème 210 ([LDR93], p. 99) *Soit T une théorie universelle.*
T insatisfiable ssi il existe $T' \subseteq Inst(T)$, T' fini t.q. T' insatisfiable en logique propositionnelle.

Idée de la démonstration.
($\Leftarrow$) Immédiat.

($\Rightarrow$)
T insatisfiable implique T n'a pas de modèle de Herbrand
implique $Inst(T)$ insatisfiable en logique prop.
implique qu'il existe $T' \subseteq Inst(T)$, T' fini, insatisfiable en logique prop.
d'après le **théorème de compacité de la logique prop.**

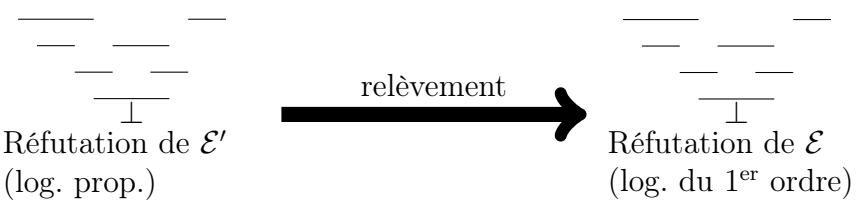
■

11.6.3 Démonstration de la complétude

Idée de la démonstration.

$\boxed{\Leftarrow}$([CRK03], p. 272) ([BA12], p. 199)

Supposons $\forall\mathcal{E}$ insatisfiable. D'après le théorème 210, il existe un ensemble fini de clauses $\mathcal{E}' \subseteq Inst(\mathcal{E})$, insatisfiable en logique prop. Par **complétude de la résolution en logique propositionnelle**, il existe une réfutation par résolution de $\mathcal{E}'$ en logique propositionnelle.



On démontre, par induction sur une preuve par résolution propositionnelle π', la propriété :

$\mathcal{P}(\pi')$: « Si π' est une preuve par résolution propositionnelle de C' pour $\mathcal{E}'$ alors il existe une clause C, une preuve par résolution du premier ordre de C pour $\mathcal{E}$, une substitution σ tel que $C' = C[\sigma]$ ».

— Cas de base. Si $C' \in \mathcal{E}'$, alors il existe $C \in \mathcal{E}$, σ tel que $C' = C[\sigma]$. D'où $\mathcal{P}(C')$.

— Cas inductif. Supposons $\mathcal{P}(C'_1)$ et $\mathcal{P}(C'_2)$. On montre $\mathcal{P}(\text{res-prop}\dfrac{\vdots\ C'_1 \quad \vdots\ C'_2}{C'})$ grâce à :

Lemme 211 *Soit C_1, C_2 des clauses sans variables communes, σ_1, σ_2 des substitutions. Si* res-prop$\dfrac{C_1[\sigma_1] \quad C_2[\sigma_2]}{C'}$ *alors*

il existe une substitution β et une clause C tel que super-res$\dfrac{C_1 \quad C_2}{C}$ *et $C' = C[\beta]$.*

Exemple 212 *([BA12], p. 199)*

$$\textit{res-prop}\ \frac{p(f(a)) \vee q(f(a)) \qquad \neg p(f(a)) \vee r(a)}{q(f(a)) \vee r(a)}$$

relèvement

$$\textit{super-res}\ \frac{p(x) \vee p(f(y)) \vee p(f(z)) \vee q(x) \qquad \neg p(f(u)) \vee \neg p(w) \vee r(u)}{q(f(u)) \vee r(u)}$$

■

Annexe A

Démonstration de la complétude du premier ordre

1. Témoins de Henkin. Soit Δ consistante. On considère $(\Sigma_n)_{n\in\mathbb{N}}$ la suite de signature et $(\Delta_n)_{n_\mathbb{N}}$ la suite de théories définies par :
   — $\Sigma_0 = \Sigma$;
   — $\Delta_0 = \Delta$;
   et
   — $\Sigma_{n+1} = \Sigma_n \cup \{c_{\psi(x)} \mid \psi(x) \text{ est une } \Sigma_n\text{-formule avec } x \text{ comme seule variable libre}\}$;
   — $\Delta_{n+1} := \Delta_n \cup \{\exists\psi(x) \to \psi(c_{\psi(x)}) \mid \psi(x) \text{ est une } \Sigma_n\text{-formule avec } x \text{ comme seule variable libre}\}$.
   Soit $\Delta'' := \bigcup_n \Delta_n$. Soit $\Sigma'' := \bigcup_n \Sigma_n$.

   Proposition 213 *Δ'' est une Σ''-théorie consistante.*

   Idée de la démonstration.
   On montre par récurrence sur n que Δ_n est consistante. ■

   Obtenir une théorie syntaxiquement complète. Soit $(\varphi_n)_{n\in\mathbb{N}}$ une énumération des Σ''-formules. On construit la suite de théories $(K_n)_{n\in\mathbb{N}}$ définie par :
   — $K_0 = \Delta''$;
   — $K_{n+1} = \begin{cases} K_n \cup \{\varphi_n\} & \text{si } K_n \nvdash_{DN} \varphi_n \text{ et } K_n \nvdash_{DN} \neg\varphi_n \\ K_n. \end{cases}$.
   On pose $\Delta' := \bigcup_{n\in\mathbb{N}} K_n$.

   Proposition 214 *Δ' est une Σ''-théorie :*
   — *consistante ;*
   — *syntaxiquement complète ;*
   — *telle que pour toute Σ''-formule $\psi(x)$, $\Delta' \vdash_{DN} (\exists x\psi(x) \to \psi(c_\psi))$.*

2. Le modèle $\mathcal{M}$ est bien défini.
3. On démontre le lemme 136 par induction sur φ. Soit la propriété
   $\mathcal{P}(\varphi)$: Soit $x_1, \ldots, x_n$ les variables libres de φ et soit $t_1, \ldots t_n$ des termes clos. On a : $\mathcal{M}[x_1 := [t_1], \ldots, x_n := [t_n]] \models \varphi$ ssi $\Delta' \vdash_{DN} \varphi(t_1, \ldots, t_n)$
   — On a :
   $$\begin{aligned} \mathcal{M}[x_1 := [t_1], \ldots, x_n := [t_n]] \models p(u_1, \ldots, u_k) \quad & \text{ssi } p^{\mathcal{M}}([v_1], \ldots, [v_k]) = 1 \\ & \text{ssi } \Delta' \vdash_{DN} \varphi(v_1, \ldots, v_k) \\ & \text{ssi } \Delta' \vdash_{DN} \varphi(u_1, \ldots, u_k)[x_1 := t_1, \ldots, x_n := t_n] \\ & \quad \text{où } v_i = u_i[x_1 := t_1, \ldots, x_n := t_n]. \end{aligned}$$
   d'où $\mathcal{P}(p(v_1, \ldots, v_k))$.

— Supposons $\mathcal{P}(\varphi)$.

$\mathcal{M}[x_1 := [t_1], \dots] \models \neg\varphi$ ssi $\mathcal{M}[x_1 := [t_1], \dots] \not\models \varphi$
ssi $\Delta' \nvdash_{DN} \varphi(\vec{t})$ par $\mathcal{P}(\varphi)$
ssi $\Delta' \vdash_{DN} \neg\varphi(\vec{t})$ car Δ' syntaxiquement complète

D'où $\mathcal{P}(\neg\varphi)$.

— $\vdots$

— Supposons $\mathcal{P}(\varphi)$.

$\mathcal{M}[x_1 := [t_1], \dots] \models \exists x\varphi$ implique il existe $t \in D$ tq $\mathcal{M}[x_1 := [t_1], \dots x := [t]] \models \varphi$
ssi $\Delta' \vdash_{DN} \varphi(\vec{t}, t)$ par $\mathcal{P}(\varphi)$
implique $\Delta' \vdash_{DN} \exists x, \varphi(\vec{t})$ car avec la règle $\exists_i$.

$\Delta' \vdash_{DN} \exists x, \varphi(\vec{t})$ implique $\Delta' \vdash_{DN} \varphi(\vec{t}, c_{\varphi(\vec{t},x)})$ car $\Delta' \vdash_{DN} \exists x, \varphi(\vec{t}) \to \varphi(\vec{t}, c_{\varphi(\vec{t},x)})$
$\mathcal{M}[x_1 := [t_1], \dots, x = c_{\varphi(\vec{t},x)}] \models \varphi$ par $\mathcal{P}(\varphi)$
ssi $\Delta' \vdash_{DN} \varphi(\vec{t}, t)$ par $\mathcal{P}(\varphi)$
implique $\mathcal{M}[x_1 := [t_1], \dots] \models \exists x\varphi$.

D'où $\mathcal{P}(\exists x, \varphi)$.

Annexe B

Tailles des représentations des fonctions booléennes

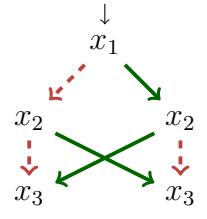
Théorème 215 *Considérons la fonction parité*

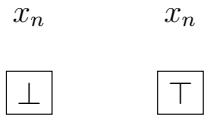
$$\begin{array}{rccc} parity: & \{0,1\}^n & \to & \{0,1\} \\ & (x_1,\ldots,x_n) & \mapsto & \sum_{i=1}^n x_i \ mod\ 2. \end{array}$$

1. *Il existe une formule de taille $O(n^2)$ pour parity ;*
2. *Il existe un circuit de taille $O(n)$ et de profondeur $O(\log n)$ pour parity ;*
   ([Sip06], Ex. 9.29)
3. *Il existe un BDD pour parity de taille $O(n)$;*
4. *Toute forme normale conjonctive (disjonctive) de parity est de taille $\geq 2^{n-1} \times n$;*
   ([Sav98], p. 84, Ex. 2.8)
5. *Tout circuit de parity de profondeur constante est de taille exponentielle en n.*
   ([AB09], p. 287 ; [Sav98], p. 450, Th. 9.7.4)

Idée de la démonstration.

1. Par induction, on définit
   — $\varphi_1(x_1) = x_1$;
   — Pour $n > 1$, $\varphi_n(x_1,\ldots,x_n) = (\varphi_{n/2}(x_1,\ldots,x_{n/2}) \wedge \neg\varphi_{n/2}(x_{n/2+1},\ldots,x_n)) \vee (\neg\varphi_{n/2}(x_1,\ldots,x_{n/2}) \wedge \varphi_{n/2}(x_{n/2+1},\ldots,x_n))$
   La taille $T(n)$ de φ_n vérifie $T(n) = 4T(n/2) + O(1)$ d'où $T(n) = O(n^2)$.
2. Même construction mais sans recopier les formules identiques.
3. On construit un BDD de la forme suivante :





4. Soit φ une forme normale disjonctive pour *parity*. Supposons qu'une clause ne parle pas d'un certain x_i. Soit une valuation ν qui rende cette clause vraie. En changeant la valeur de x_i dans ν, la formule reste vraie. Contradiction car φ est censé être une formule pour *parity*. Donc, pour tout $i \in \{1, \ldots, n\}$, chaque clause contient x_i ou $\neg x_i$. Ainsi, la DNF est une disjonction des conjunctions qui, elles, décrivent les valuations complètes qui rendent φ vrais. Il doit y en avoir au moins $2^n/2$.
5. Résultat difficile "$parity \notin AC^0$".

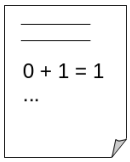
■

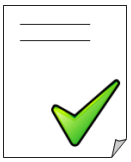
Annexe C

Grands théorèmes de logique

C.1 Indécidabilité de l'arithmétique

Théorème 216 ([CL93], p. 92) *Soit T est une théorie du premier ordre.*

Si 

T inclut PA *et* 
T consistante *alors le problème de T-validité est indécidable.*

Idée de la démonstration.
Par l'absurde, soit T-valide ? un algorithme qui décide le problème de T-validité.

fonction paradoxe(n)
  calculer φ_n la $n^{\text{ème}}$ formule
  retourner non T-valide ?($\varphi_n(n)$)

Il existe une formule $\psi(x)$ telle que (théoreme de représentation) pour tout entier n :
- $T \vdash \psi(n)$ si paradoxe(n) retourne vrai ;
- $T \vdash \neg\psi(n)$ si paradoxe(n) retourne faux.

Soit n_ψ le numéro de la formule $\psi(x)$.

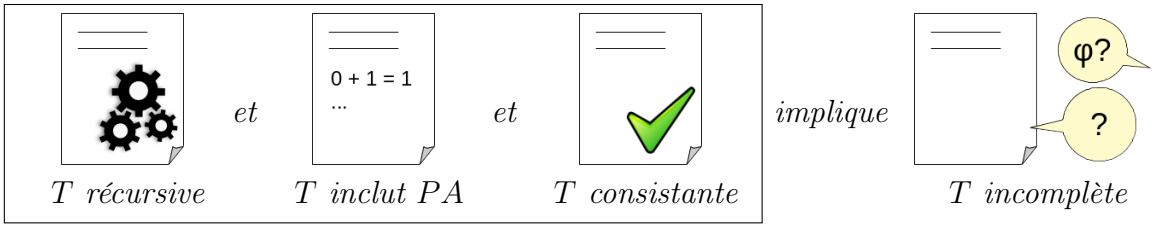
paradoxe(n_ψ) retourne vrai ssi $T \nvdash \psi(n_\psi)$ ssi paradoxe(n_ψ) retourne faux.

■

C.2 Théorème d'incomplétude de Gödel-Rosser

Définition 217 (théorie récursive)
T **récursive** si le problème suivant est décidable :
— entrée : une formule φ ;
— sortie : oui si $\varphi \in T$; non, sinon.

Théorème 218 (d'incomplétude de Gödel-Rosser, [CL93], p. 93) *Soit T une théorie de la logique du premier ordre. On a :*



Idée de la démonstration.
Si T était complète, alors voici un algorithme qui décide la T-validité :

```
fonction T-valide ?(φ)
    pour n := 0, 1, 2...
        pour toute preuve de longueur ≤ n
            Si c'est une preuve de φ, retourner vrai
            Si c'est une preuve de ¬φ, retourner faux
```

Contradiction avec le théorème 216.
■

Bibliographie

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[BA12] Mordechai Ben-Ari. *Mathematical logic for computer science*. Springer Science & Business Media, 2012.

[BJ87] George Boolos and Richard C. Jeffrey. *Computability and logic (2. ed.)*. Cambridge University Press, 1987.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[BN98] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.

[BOKR86] Michael Ben-Or, Dexter Kozen, and John Reif. The complexity of elementary algebra and geometry. *Journal of Computer and System Sciences*, 32(2) :251–264, 1986.

[Bus87] Samuel R. Buss. The boolean formula value problem is in ALOGTIME. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 123–131, 1987.

[BW96] Beate Bollig and Ingo Wegener. Improving the variable ordering of obdds is np-complete. *IEEE Trans. Computers*, 45(9) :993–1002, 1996.

[Can88] John Canny. Some algebraic and geometric computations in pspace. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 460–467. ACM, 1988.

[Car08] Olivier Carton. *Langages formels, calculabilité et complexité*, volume 101. Vuibert, 2008.

[CL93] René Cori and Daniel Lascar. Logique mathématique ii. fonctions récursives, théorème de gödel, théorie des ensembles, théorie des modèles, 1993.

[CRK03] Lascar D CORI R and JL Krivine. Logique mathématique, tome 1 : Calcul propositionnel, algèbre de boole, calcul des prédicats, coll. *Sciences Sup, Dunod*, 2003.

[DLL12] Stéphane Devismes, Pascal Lafourcade, and Michel Lévy. *Logique et démonstration automatique : introduction à la logique propositionnelle et à la logique du premier ordre*. Ellipses, 2012.

[DNRC01] René David, Karim Nour, Christophe Raffalli, and Pierre-Louis Curien. *Introduction à la logique : théorie de la démonstration : cours et exercices corrigés*. Dunod, 2001.

[DPV16] Sanjoy Dasgupta, Christos H Papadimitriou, and Umesh Virkumar Vazirani. Algorithms. 2016.

[Dup15] Jacques Duparc. *La logique pas à pas*. PPUR Presses polytechniques, 2015.

[EIS75] Shimon Even, Alon Itai, and Adi Shamir. On the complexity of time table and multicommodity flow problems. In *Foundations of Computer Science, 1975., 16th Annual Symposium on*, pages 184–193. IEEE, 1975.

[FFR74] Michael Jo Fischer, Michael J Fischer, and Michael O Rabin. Super-exponential complexity of presburger arithmetic. 1974.

[GLM01] Jean Goubault-Larrecq and Ian Mackie. *Proof theory and automated deduction*, volume 6. Springer Science & Business Media, 2001.

[Har09] John Harrison. *Handbook of practical logic and automated reasoning*. Cambridge University Press, 2009.

[KS08] Daniel Kroening and Ofer Strichman. *Decision procedures : an algorithmic point of view*. Springer Science & Business Media, 2008.

[Lal90] René Lalement. Logique, réduction, résolution, 1990.

[LDR93] Richard Lassaigne and Michel De Rougemont. Logique et fondements de l'informatique. *Hermes, Paris*, 1993.

[LS15] Romain Legendre and François Schwarzentruber. *Compilation : Analyse lexicale et syntaxique du texte à sa structure en informatique*. Reference Sciences. Ellipses, 2015.

[Mat03] Ju V Matijasevič. Enumerable sets are diophantine. In *Mathematical Logic In The 20th Century*, pages 269–273. 2003.

[Opp78] Derek C Oppen. A 2 22pn upper bound on the complexity of presburger arithmetic. *Journal of Computer and System Sciences*, 16(3) :323–332, 1978.

[Sav98] John E. Savage. *Models of computation - exploring the power of computing*. Addison-Wesley, 1998.

[Sip97] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

[Sip06] M. Sipser. *Introduction to the Theory of Computation*, volume 27. Thomson Course Technology Boston, MA, 2006.

[SM73] Larry J Stockmeyer and Albert R Meyer. Word problems requiring exponential time (preliminary report). In *Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 1–9. ACM, 1973.

[THY93] Seiichiro Tani, Kiyoharu Hamaguchi, and Shuzo Yajima. The complexity of the optimal variable ordering problems of shared binary decision diagrams. In *Algorithms and Computation, 4th International Symposium, ISAAC '93, Hong Kong, December 15-17, 1993, Proceedings*, pages 389–398, 1993.

[Tru97] John K Truss. *Foundations of mathematical analysis*. Oxford University Press, 1997.

Index