Quelques algorithmes pour la démonstration automatique accompagnés d'une implantation simpliste

DEA Logique et fondements de l'informatique
Notes de cours

11 décembre 2006

I - Le calcul des propositions

On utilisera les connecteurs $\neg$, $\wedge$, $\vee$, $\Rightarrow$ et $\Leftrightarrow$, les deux constantes propositionnelles T et F ainsi qu'un ensemble (infini dénombrable) de variables propositionnelles $\mathcal{P}$. Une *assignation*, ou une *distribution* est une fonction σ de $\mathcal{P}$ dans $\{\mathsf{T}, \mathsf{F}\}$. L'assignation σ est étendue de façon standard aux formules. La sémantique des connecteurs est donnée par :

$\sigma(\neg A) = \mathsf{T}$,	si $\sigma(A) = \mathsf{F}$	$\sigma(\neg A) = \mathsf{F}$,	si $\sigma(A) = \mathsf{T}$
$\sigma(A \wedge B) = \mathsf{T}$,	si $\sigma(A) = \mathsf{T}$ et $\sigma(B) = \mathsf{T}$	$\sigma(A \wedge B) = \mathsf{F}$,	si $\sigma(A) = \mathsf{F}$ ou $\sigma(B) = \mathsf{F}$
$\sigma(A \vee B) = \mathsf{T}$,	si $\sigma(A) = \mathsf{T}$ ou $\sigma(B) = \mathsf{T}$	$\sigma(A \vee B) = \mathsf{F}$,	si $\sigma(A) = \mathsf{F}$ et $\sigma(B) = \mathsf{F}$
$\sigma(A \Rightarrow B) = \mathsf{T}$,	si $\sigma(A) = \mathsf{F}$ ou $\sigma(B) = \mathsf{T}$	$\sigma(A \vee B) = \mathsf{F}$,	si $\sigma(A) = \mathsf{F}$ et $\sigma(B) = \mathsf{T}$
$\sigma(A \Rightarrow B) = \mathsf{T}$,	si $\sigma(A) = \sigma(B)$	$\sigma(A \vee B) = \mathsf{F}$,	si $\sigma(A) \neq \sigma(B)$

On dit qu'une formule A est *réalisable*, ou encore *satisfaisable*, s'il existe une assignation σ telle que $\sigma(A) = \mathsf{T}$.
On dit qu'une formule A est *valide*, ou encore qu'elle est une *tautologie*, si pour toute assignation σ, $\sigma(A) = \mathsf{T}$.

On examinera six méthodes de décision :

1. Méthode des tableaux matriciels (algorithme de Quine)
2. Méthode de tableaux sémantiques (Beth, Hinkita, Smullyan)
3. Méthode des connexions (Bibel, Wallen)
4. Méthode des *"if-expressions"* (Boyer, Moore)
5. Méthode de réduction à la forme normale (Hilbert, Ackermann)
6. Méthode de résolution (Robinson)

§ 1. Tableaux matriciels

C'est la méthode bien connue des tables de vérité qui consiste à assigner successivement les valeurs T et F à chacunes des variables propositionnelles. Quine a amélioré cette coûteuse méthode en proposant l'application de règles de simplification lors de la substitution des constantes aux variables propositionnelles. Une meilleure performance (en moyenne) est obtenue en susbtituant d'abord à la variable ayant le plus d'occurence dans la formule.

(1.a) Règles de simplifications

On écrit les régles sous la forme $A \longrightarrow A'$, où A et A' sont deux formules. On choisit A et A' de telle sorte

que $A \Leftrightarrow A'$ soit valide. Ce qui fonde la validité des règles (théorème de remplacement). De plus, A' est une formule de taille strictement inférieure à A, ce qui assure la terminaison du processus de simplification. Les règles proposées par Quine sont :

$\neg\mathsf{F}$	$\longrightarrow$	T	$\neg\mathsf{T}$	$\longrightarrow$	F	$\neg\neg A$	$\longrightarrow$	A		–	
$\mathsf{T} \wedge A$	$\longrightarrow$	A	$A \wedge \mathsf{T}$	$\longrightarrow$	A	$\mathsf{F} \wedge A$	$\longrightarrow$	F	$A \wedge \mathsf{F}$	$\longrightarrow$	F
$\mathsf{T} \vee A$	$\longrightarrow$	T	$A \vee \mathsf{T}$	$\longrightarrow$	T	$\mathsf{F} \vee A$	$\longrightarrow$	A	$A \vee \mathsf{F}$	$\longrightarrow$	A
$\mathsf{T} \Rightarrow A$	$\longrightarrow$	A	$\mathsf{F} \Rightarrow A$	$\longrightarrow$	T	$A \Rightarrow \mathsf{T}$	$\longrightarrow$	T	$A \Rightarrow \mathsf{F}$	$\longrightarrow$	$\neg A$
$\mathsf{T} \Leftrightarrow A$	$\longrightarrow$	A	$A \Leftrightarrow \mathsf{T}$	$\longrightarrow$	A	$\mathsf{F} \Leftrightarrow A$	$\longrightarrow$	$\neg A$	$A \Leftrightarrow \mathsf{F}$	$\longrightarrow$	$\neg A$

On dit qu'une règle est *applicable* à une formule si celle-ci a l'une des formes de l'un des membres gauches des régles.

(1.b) Un algorithme

On appelle S la fonction de simplification vérifiant, pour toute formule A :

$S(A) = A$, si aucune règle n'est applicable à A
$A \longrightarrow S(A)$, sinon

Etant donnée une formule A on veut itérer l'application des règles de simplification jusqu'à saturation. On définit pour cela la fonction suivante S^* telle que

Si $A = \neg A_1$ et si $A'_1 = S^*(A_1)$ alors
$S^*(A) = S(A'_1)$
Sinon, si $A = A_1 \star A_2$ avec $\star \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$, si $A'_1 = S^*(A_1)$ et si $A'_2 = S^*(A_2)$ alors
$S^*(A) = S(A'_1 \star A'_2)$

Soit alors la fonction T définie sur une formule A et un ensemble de variables propositionnelles X.

Si $A = \mathsf{T}$ ou $A = \mathsf{F}$ ou $X = \emptyset$ alors
$T(A, X) = A$
Sinon, si $X = \{x\} \cup X'$ et si $A' = S(S^*(A[\mathsf{T}/x]) \wedge S^*(A[\mathsf{F}/x]))$ alors
$T(A, X) = T(A', X')$

Si X est l'ensemble des variables propositionnelles de la formule A et si $T(A, X) = \mathsf{T}$ alors A est une tautologie.

§ 2. Les tableaux sémantiques

La méthode des tableaux matriciels est une méthode *synthétique* : elle procède de l'assignation des valeurs de vérité aux variables propositionnelles pour en déduire la valeur de vérité des formules. La méthode des tableaux sémantique est *analytique* : elle procède de l'assignation d'une valeur de vérité aux formules pour en déduire une, ou plusieurs, assignations de valeur de vérité aux variables propositionnelles.
Pour démontrer qu'une formule est une tautologie en utilisant une méthode analytique, on cherche à construire une assignation réalisant la négation de cette formule. Si toute tentative de construction de l'assignation échoue, c'est que la formule est bien une tautologie.

(2.a) Règles d'analyse

On se donne deux symboles $^\top$ et $^\bot$ qui nous serviront à marquer les formules. On notera $A^\top$ et $A^\bot$ et on dira que ce sont des *formules marquées*. Intuitivement, $A^\top$ signifie *"vérifier A"* et $A^\bot$, *"falsifier A"*. Les règles d'analyse de la méthode des tableaux sémantiques sont également des règles de *simplification* : elles font décroître la taille des formules à analyser. Il existe deux sortes de règles : les règles de *prolongement*,

dites aussi *règles* α ; les règles de *branchement*, dites aussi *règles* β. Nous présentons les règles sous forme arborescente.

$(\neg A)^\top$ $\downarrow$ $A^\bot$	$(A \wedge B)^\top$ $\downarrow$ $A^\top$ $\downarrow$ $B^\top$	$(A \vee B)^\top$ $\swarrow\ \searrow$ $A^\top\ \ B^\top$	$(A \Rightarrow B)^\top$ $\swarrow\ \searrow$ $A^\bot\ \ B^\top$	$(A \Leftrightarrow B)^\top$ $\swarrow\ \searrow$ $A^\top\ \ A^\bot$ $\downarrow\ \ \downarrow$ $B^\top\ \ B^\bot$
$(\neg A)^\bot$ $\downarrow$ $A^\top$	$(A \wedge B)^\bot$ $\swarrow\ \searrow$ $A^\bot\ \ B^\bot$	$(A \vee B)^\bot$ $\downarrow$ $A^\bot$ $\downarrow$ $B^\bot$	$(A \Rightarrow B)^\bot$ $\downarrow$ $A^\top$ $\downarrow$ $B^\bot$	$(A \Leftrightarrow B)^\bot$ $\swarrow\ \searrow$ $A^\top\ \ A^\bot$ $\downarrow\ \ \downarrow$ $B^\bot\ \ B^\top$

On peut extraire de l'arbre obtenu par itération de l'application des règles une relation, appelée *pré-distribution*, entre $\mathcal{P}$ et $\{\mathsf{T}, \mathsf{F}\}$ qui, étant donnée une branche de l'arbre d'analyse, associe T à la variable p si $p^\top$ figure dans l'arbre et F si $p^\bot$ y figure. On obtient sur chaque branche de l'arbre une assignation de valeur de vérité pour une variable propositionnelle p lorsque ne figurent pas sur une même branche à la fois $p^\top$ et $p^\bot$. Si aucune des branches de l'arbre obtenu ne permet de déduire une assignation, c'est que la formule à la racine de l'arbre n'est pas satisfaisable.

On peut obtenir la correction de la méthode des tableaux sémantiques en interprétant les formules marquées comme des séquents et les règles d'analyse comme des règles du calcul des séquents. Ainsi un arbre construit selon les règles d'analyse sémantique est un arbre de dérivation du calcul des séquents. A $A^\top$ est interprété par $\vdash A$ et $A^\bot$, par $A \vdash$. A chaque règle d'analyse, on associe

$$\begin{array}{c} (\neg A)^\top \\ \downarrow \\ A^\bot \end{array} \qquad \frac{A \vdash}{\vdash \neg A} \qquad \begin{array}{c} (\neg A)^\bot \\ \downarrow \\ A^\top \end{array} \qquad \frac{\vdash A}{\neg A \vdash}$$

$$\begin{array}{c} (A \wedge B)^\top \\ \downarrow \\ A^\top \\ \downarrow \\ B^\top \end{array} \qquad \frac{\vdash A \quad \vdash B}{\vdash A \wedge B} \qquad \begin{array}{c} (A \wedge B)^\bot \\ \swarrow\ \searrow \\ A^\bot\ \ B^\bot \end{array} \qquad \frac{A, B \vdash}{A \wedge B \vdash}$$

$$\begin{array}{c} (A \vee B)^\top \\ \swarrow\ \searrow \\ A^\top\ \ B^\top \end{array} \qquad \frac{\vdash A, B}{\vdash A \vee B} \qquad \begin{array}{c} (A \vee B)^\bot \\ \downarrow \\ A^\bot \\ \downarrow \\ B^\bot \end{array} \qquad \frac{A \vdash \quad B \vdash}{A \vee B \vdash}$$

$$\begin{array}{c} (A \Rightarrow B)^\top \\ \swarrow\ \searrow \\ A^\bot\ \ B^\top \end{array} \qquad \frac{A \vdash B}{\vdash A \Rightarrow B} \qquad \begin{array}{c} (A \Rightarrow B)^\bot \\ \downarrow \\ A^\top \\ \downarrow \\ B^\bot \end{array} \qquad \frac{\vdash A \quad B \vdash}{A \Rightarrow B \vdash}$$

$$\begin{array}{c} (A \Leftrightarrow B)^\top \\ \swarrow\ \searrow \\ A^\top\ \ A^\bot \\ \downarrow\ \ \downarrow \\ B^\top\ \ B^\bot \end{array} \qquad \frac{A \vdash B \quad B \vdash A}{\vdash A \Leftrightarrow B} \qquad \begin{array}{c} (A \Leftrightarrow B)^\bot \\ \swarrow\ \searrow \\ A^\top\ \ A^\bot \\ \downarrow\ \ \downarrow \\ B^\bot\ \ B^\top \end{array} \qquad \frac{A, B \vdash \quad \vdash A, B}{A \Leftrightarrow B \vdash}$$

(2.b) Un algorithme

Le but essentiel de la méthode des tableaux sémantiques est d'utiliser les règles d'analyse des formules pour construire une distribution de valeurs de vérité. Les fonctions que nous allons définir sont une simplification en ce sens de la méthode des tableaux : on ne retiendra que la suite des variables marquées rencontrées lors de l'analyse.

Appelons R la fonction correspondant aux règles d'analyse. Elle est définie sur les formules marquées et retourne les formules marquées résultantes ainsi que le type, α ou β, de la règle. Appelons *pré-distribution* un ensemble de variables propositionnelles marquées.
Nous définissons mutuellement deux fonctions R^* et RR^* d'itération de R. La première prend en arguments une formule marquée M ainsi qu'une pré-distibution δ (éventuellement vide) et calcule l'ensemble des pré-distributions résultant de l'application itérée de R. La seconde prend en arguments une formule marquée M ainsi qu'un ensemble de pré-distributions Δ et calcule l'ensemble des pré-distributions résultant de l'application de R^* pour M et chacun des éléments δ de Δ.

Si une pré-distribution δ contient une variable marquée $p^\top$ (resp. $p^\bot$) on sait que l'on ne pourra obtenir une distribution de valeurs de vérité dés que l'on doit ajouter à δ la variable marquée $p^\bot$ (resp. $p^\top$) On se donne donc l'opération $\oplus$ telle que

$$\begin{array}{lcll} \delta \oplus p^\top & = & \emptyset, & \text{si } p^\bot \in \delta, \\ & = & \delta \cup \{p^\top\}, & \text{sinon, et} \\ \delta \oplus p^\bot & = & \emptyset, & \text{si } p^\top \in \delta, \\ & = & \delta \cup \{p^\bot\}, & \text{sinon} \end{array}$$

On définit alors mutuellement R^* et RR^* par

Si M est une variable propositionnelle (marquée) m alors
$R^*(M, \delta) = \{\delta \oplus m\}$
sinon :
si $R(M) = (\alpha, M_1)$ alors $R^*(M, \delta) = R^*(M_1, \delta)$;
si $R(M) = (\alpha, M_1, M_2)$ alors $R^*(M, \delta) = RR^*(M_2, R^*(M_1, \delta))$;
si $R(M) = (\beta, M_1, M_2)$ alors $R^*(M, \delta) = R^*(M_1, \delta) \cup R^*(M_2, \delta)$;
si $R(M) = (\beta, M_{11}, M_{12}, M_{21}, M_{22})$ alors
$R^*(M, \delta) = RR^*(M_{12}, R^*(M_{11}, \delta)) \cup RR^*(M_{22}, R^*(M_{21}, \delta))$
Et :
si $\Delta = \emptyset$ alors $RR^*(M, \Delta) = \emptyset$;
Si $\Delta = \Delta' \cup \{\emptyset\}$ alors $RR^*(M, \Delta) = RR^*(M, \Delta')$;
Si $\Delta = \Delta' \cup \delta$ et $\delta \neq \emptyset$ alors $RR^*(M, \Delta) = R^*(M, \delta) \cup RR^*(M, \Delta')$

Si $R^*(A^\bot, \emptyset) = \emptyset$ alors A est une tautologie.

§ 3. La méthode des connexions

Lorsque nous avons implémenté la méthode des tableaux sémantiques, nous nous sommes fait la remarque que seules, en fait, comptaient les variables propositionnelles. La méthode des connexions n'est rien d'autre que la mise en forme de cette remarque. Aussi se définit-elle plutôt comme un parcours de la formule à démontrer afin d'en extraire ces connexions d'atomes (*ie* variables propositionnelles) *complémentaires*.
La complémentarité des atomes dans la méthode des tableaux nous était donnée par la propagation, à l'aide des règles d'analyse, de la valeur de vérité attendue des formules. Dans la méthode des connexions, on utilisera une notion de *polarité* des (sous)formules.

(3.a) Polarité

On parlera donc d'*occurrence positive* ou d'*occurrence négative* d'une sous-formule (au sens large) On notera $A^\top$ une occurence positive de A et $A^\bot$ une occurence négative. La table suivante résume la définition de la

polarité des constituants d'une formule.

$(\neg A)^\top$	$A^\perp$	
$(\neg A)^\perp$	$A^\top$	
$(A \wedge B)^\top$	$A^\top$	$B^\top$
$(A \wedge B)^\perp$	$A^\perp$	$B^\perp$
$(A \vee B)^\top$	$A^\top$	$B^\top$
$(A \vee B)^\perp$	$A^\perp$	$B^\perp$
$(A \Rightarrow B)^\top$	$A^\perp$	$B^\top$
$(A \Rightarrow B)^\perp$	$A^\top$	$B^\perp$

Comme la méthode des tableaux, la méthode des connexions est une méthode indirecte : elle cherche à falsifier la tautologie candidate. On attribue donc une polarité négative à la formule elle-même.

(3.b) Types et chemins

A chaque sous-formule polarisée, on attribue un *type* (α ou β) qui, comme dans la méthode des tableaux sémantiques, distingue les formules qui donneront lieu à une *prologation* ou à un *branchement* de chemins. Les variables propositionnelles ne recoivent pas de type. Les règles d'attribution d'un type aux formules de la méthode des connexions sont *mutatis mutandis* les règles d'analyse de la méthode des tableaux.

Le type des formules est utilisé pour construire, en suivant la structure syntaxique des formules, un ensemble des chemins reliant les variables propositionnelles.
On note $x.X$ la suite dont le premier élément est x et qui se poursuit par la suite X. On note $X.x$ la suite dont le dernier élément est x et qui commence par la suite X. On note $[x_1, .., x_n]$ la suite des éléments x_1, .., x_n et $[]$ la suite vide. Par abus de langage, on note également $X.Y$ la concaténation de la suite X avec la suite Y. On appelle *occurence* dans une formule A une suite d'entiers définie par :

$[]$ est une occurrence dans A ;
si ρ est une occurrence dans A alors $\rho.1$ est une occurence dans $\neg A$;
si ρ_1 est une occurence dans A_1 et ρ_2, une occurence dans A_2 alors $\rho_1.1$ et $\rho_2.2$ sont des occurences dans $A_1 \wedge A_2$, $A_1 \vee A_2$, $A_1 \Rightarrow A_2$ et $A_1 \Leftrightarrow A_2$.

Si ρ est une occurrence dans A, on note $A|_\rho$ la *sous formule de A d'occurence ρ* que l'on définit par :

si $\rho = []$ alors $A|_\rho = A$;
si $\rho = 1.\rho'$ et $A = \neg A'$ alors $A|_\rho = A'|_{\rho'}$;
si $A = A_1 \star A_2$ avec $\star \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}$ alors
  si $\rho = 1.\rho'$ alors $A|_\rho = A_1|_{\rho_1}$
  sinon, si $\rho = 2.\rho'$ alors $A|_\rho = A_2|_{\rho_2}$.

Si A et ses sous formules sont polarisées, on pourra parler du type de $A|_\rho$. Un *chemin* dans une formule A est un ensemble d'occurences dans A tel que

- $\{[]\}$ est un chemin ;
- si $P = \{\rho\} \cup P'$ est un chemin, si $A|_\rho$ est de type α alors $P' \cup \{\rho.1\}$ ou $P' \cup \{\rho.1, \rho.2\}$ sont des chemins selon que $\rho.2$ existe ou non ;
- si $P = \{\rho\} \cup P'$ est un chemin, si $A|_\rho$ est de type β alors $P' \cup \{\rho.1\}$ et $P' \cup \{\rho.2\}$ sont des chemins.

(3.c) Un algorithme

L'implantation que nous proposons est extrèment proche de celle (optimisée) que nous avons donnée de la méthode des tableaux sémantiques. Cependant, comme la méthode des connexions s'énonce directement comme méthode de construction de chemins entre variables propositionnelles, les fonctions obtenues sont plus simples.

On reprend, *mutatis mutandis*, la fonction R de la méthode des tableaux sémantiques. Nous définissons deux fonctions RR et RR^*. La première (RR) est définie sur un couple constitué d'une suite F de formules polarisées et d'un ensemble P de variables propositionnelles également polarisées (*ie* un chemin) ; elle sera chargée de déterminer le ou les prochains chemins. La seconde (RR^*) est définie sur une suite S de couples constitués d'une suite de formules polarisées et d'un ensemble P de variables propositionnelle polariées. Elle est chargée d'itérer RR sur chaque élément de son argument pour calculer une suite de chemins.

La fonction $RR(F, P)$ est définie par

Si $F = []$ alors $RR(F, P) = [(F, P)]$;
Si $F = m.F'$ et m une variable (polarisée) alors
  $R(F, P) = [(F', m \oplus P)]$
Sinon, $F = M.F'$:
  Si $R(M) = (\alpha, M_1)$ alors $RR(F, P) = [(M_1.F', P)]$;
  Si $R(M) = (\alpha, M_1, M_2)$ alors $RR(F, P) = [(M_1.M_2.F, P)]$;
  Si $R(M) = (\beta, M_1, M_2)$ alors $RR(F, P) = [(M_1.F, P), (M_2.F, P)]$;
  Ssi $R(M) = (\beta, M_1, M_2, M_3, M_4)$ alors $Rr(F, P) = [(M_1.M_2.F, P), (M_3.M_4.F, P)]$

La fonction $RR^*(S)$ est définie par

Si $S = []$ alors $RR^*(S) = []$;
Si $S = ([], P).S'$ alors $RR^*(S) = P.RR^*(S')$;
Si $S = (F, P).S'$ et $F \neq []$ alors
  Soit $S'' = RR(F, P)$;
    Si $S'' = []$ alors $RR^*(S) = RR^*(S')$
    Sinon $RR^*(S) = RR^*(S''.S')$

§ 4. Les «if-expressions»

Les méthodes que nous avons données jusqu'à présent n'opéraient pas de transformation de la formule à démontrer. Nous allons développer dans ce paragraphe et dans les suivants des méthodes nécessitant un traitement préalable ou une forme particulière des formules.

(4.a) Codage des connecteurs

On peut coder les connecteurs binaires du calcul des propositions en utilisant un seul connecteur ternaire : le *if-then-else* des langages de programmation. Nous noterons If ce connecteur et $\text{If}(A, B, C)$ son application. Sa sémantique est celle attendue :

$$\begin{aligned} \text{If}(\mathsf{T}, A, B) &\Leftrightarrow A \\ \text{If}(\mathsf{F}, A, B) &\Leftrightarrow B \end{aligned}$$

La traduction des connecteurs usuels est donnée par le tableau suivant

$\neg A$	$\text{If}(A, \mathsf{F}, \mathsf{T})$
$A \wedge B$	$\text{If}(A, B, \mathsf{F})$
$A \vee B$	$\text{If}(A, \mathsf{T}, B)$
$A \Rightarrow B$	$\text{If}(A, B, \mathsf{T})$
$A \Leftrightarrow B$	$\text{If}(A, B, \text{If}(B, \mathsf{F}, \mathsf{T}))$

(4.b) Forme canonique

On définit par induction sur les «if-expressions»une *forme canonique* :

(i) Les variables propositionnelles et les constantes sont en forme canonique.

(ii) Si p est une variable propositionnelle et c est une constantes, si A et B sont en forme canonique alors $\text{If}(p, A, B)$ et $\text{If}(c, A, B)$ sont en forme canonique.

En utilisant les équivalences définissant la sémantique des «if-expressions», ont peut définir une *forme canonique réduite*.

A toute «if-expressions» correspond une (unique) forme canonique réduite donnée par la fonction CR définie par :

$$\begin{array}{lcl} CR(p) & = & p \\ CR(c) & = & c \\ CR(\text{If}(\mathsf{T}, A, B)) & = & CR(A) \\ CR(\text{If}(\mathsf{F}, A, B)) & = & CR(B) \\ CR(\text{If}(p, A, B)) & = & \text{If}(p, CR(A), CR(B)) \\ CR(\text{If}(\text{If}(A, B, C), D, E)) & = & CR(\text{If}(A, \text{If}(B, D, E), \text{If}(C, D, E))) \end{array}$$

Rem : c'est un exercice intéressant que de prouver formellement la terminaison de la fonction CR.

(4.c) La méthode

La méthode de décision basée sur le codage des propositions par des «if-expressions» est analogue à celle que nous avons utilisée pour les tableaux matriciels : on substitue aux variables successivement les constantes T et F et on simplifie. La fonction S définie ci-dessous réalise cette méthode. On définit S sur les formes canoniques (non réduites à une variable propositionnelle).

Si $A \in \{\mathsf{T}, \mathsf{F}\}$ alors $S(A) = A$;
Si $A = \text{If}(p, A_1, A_2)$:
si $S(A_1[\mathsf{T}/p]) = \mathsf{T}$ alors $S(A) = S(A_2[\mathsf{F}/p])$
sinon, $S(A) = \mathsf{F}$

§ 5. Mise en forme normale

Une façon de vérifier la validité d'une proposition est de la traduire en forme *disjonctive* ou *conjonctive* normale. Une formule est en forme disjonctive normale si elle est composée d'une disjonction (n-aire) de conjonctions (n-aires) d'atomes ou de négation d'atomes. Une formule est en forme conjonctive normale si elle est composée d'une conjonction (n-aire) de disjonctions (n-aires) d'atomes ou de négation d'atomes. Nous ne traiterons que le cas de la mise en forme conjonctive normale.

(5.a) Forme conjonctive normale

Une formule en forme conjonctive normale est une tautologie si toute les disjonctions qui la compose sont des tautologies. Une disjonction est une tautologie si elle contient au moins un atome et sont opposé. Ceci nous donne une procédure de décision.

Toute formule admet une forme conjonctive normale équivalente que l'on obtient en utilisant les règles de transformation suivantes :

$A \Rightarrow B \longrightarrow \neg A \vee B$	$A \Leftrightarrow B \longrightarrow (A \wedge B) \vee (\neg A \wedge \neg B)$
Idempotence $\neg\neg A \longrightarrow A$	
Lois de de Morgan	
$\neg(A \vee B) \longrightarrow \neg A \wedge \neg B$	$\neg(A \vee B) \longrightarrow \neg A \vee \neg B$
Distributivité	
$A \vee (B \wedge C) \longrightarrow (A \vee B) \wedge (A \vee C)$	$(A \wedge B) \vee C \longrightarrow (A \vee C) \wedge (B \vee C)$

Chacune des règles fait décroître une quatité : les trois premières font disparaître un symbole ; les lois de de Morgan font décroître la hauteur des négations et les lois de distributivité celle des disjonctions.

(5.b) Un algorithme

Nous implémenterons la mise en forme conjonctive normale en deux passes :

1. supprimer les symboles $\Rightarrow$ et $\Leftrightarrow$ appliquer les lois d'idempotence et de de Morgan (*ie* mise en forme *négative normale*) ;
2. appliquer les lois de distributivité.

La fonction de mise en forme négative normale peut être une paraphrase des règles à utiliser.

La disjonction et la conjonction étant associatives, on peut considérer des connecteurs n-aires plutôt que simplement binaires. De plus, comme nous connaissons la forme des conjonctives normales, on peut même oublier les connecteurs pour représenter les formes conjonctives normales comme un ensemble (la conjonction) d'ensembles (les disjonctions) d'atomes polarisés (les variables ou leur négation). Nous appellerons *clause* un ensemble d'atomes polarisés.
Nous ne chercherons pas à construire la forme normale des propositions pour ensuite en tester la validité : nous éliminerons les clauses triviales au fur et à mesure de leur apparition.

Si c_1 et c_2 sont deux clauses, par abus, nous noterons $c_1 \oplus c_2$ l'ensemble $c_1 \cup c_2$ si c_1 et c_2 ne contiennent pas d'atomes opposés ou l'ensemble vide sinon. Définissons alors D_1 la fonction qui distribue une clause c sur un ensemble de clauses C :

Si $C = \emptyset$ alors $D_1(c, C) = \emptyset$
sinon, $C = \{c'\} \cup C'$ et
  $D_1(c, C) = D_1(c, C')$, si $c \oplus c' = \emptyset$;
  $D_1(c, C) = (c \oplus c') \cup D_1(c, C')$, sinon.

On définit ensuite D_2 la fonction d'itération de D_1 qui distribue les clauses d'un ensemble C_1 sur un ensemble C_2, à savoir :

$D_2(C_1, C_2) = \bigcup_{c \in C_1} D_1(c, C_2)$

Appelons enfin D la fonction qui applique la distributivité sur une forme négative normale ; c'est à dire, la fonction qui transforme une forme négative normale A en un ensemble de clauses.

Si $A = p$, avec p variable alors $D(A) = \{\{p^\top\}\}$;
Si $A = \neg p$, avec p variable alors $D(A) = \{\{p^\bot\}\}$;
Si $A = A_1 \wedge A_2$ alors $D(A) = D(A_1) \cup D(A_2)$;
Si $A = A_1 \vee A_2$ alors $D(A) = D_2(D(A_1), D(A_2))$.

Si $D(\neg A) = \emptyset$ alors A est une tautologie.

§ 6. Résolution

La méthode de preuve par résolution consiste à chercher à réfuter une forme conjonctive normale que l'on appellera plutôt ici : *forme clausale*. On appelle *litéral* une variable propositionnelle ou la négation d'une variable propositionnelle (que l'on note alors $\overline{p}$). Une clause est donc un ensemble d'ensembles de litéraux. A vrai dire, ces ensembles peuvent admettre des répétions d'élément : on parlera donc plus proprement de *multi-ensembles*.
La réfutation est recherchée par application de la règle justement dite de *résolution*.

(5.a) Règle de résolution

Soient deux clauses $C_1 = \{C_1', p\}$ et $C_2 = \{C_2', \overline{p}\}$ la r'egle de résolution consiste à engendrer un *résolvant* à partir de clauses telles que C_1 et C_2. Le résolvant de la paire (C_1, C_2) est l'union de C_1' et C_2'. Si C_1 est réduite à p et C_2 à $\neg p$ alors leur résolvant est la *clause vide* notée $\bot$ (ou encore, la constante F). On notera $R(C_1, C_2)$ l'ensemble des résolvants de (C_1, C_2)

La validité de la règle de résolution repose sur la validité de l'équivalence

$$(C_1 \wedge C_2) \Leftrightarrow (C_1' \vee p) \wedge (C_2' \vee \overline{p}) \wedge (C_1' \vee C_2')$$

La méthode utilisant la règle de résolution est une méthode indirecte : on cherche à réfuter la négation de la tautologie candidate.
Pour prouver A, la marche à suivre est la suivante :

1. mettre $\neg A$ sous forme clausale, disons $C_1 \wedge .. \wedge C_n$;
2. poser $E_0 = \{C_1, .., C_n\}$;
3. puis $E_{n+1} = \bigcup R(C_i, C_j)$, pour $C_i \in \bigcup_{k\in 0..n} E_k$ et $C_j \in E_n$

Si $\bot \in \bigcup_{n\in I\!N}$ alors E_0 est inconsistant (et donc $\neg A$ n'est pas réalisable).

La méthode par résolution est, en général, grandement améliorée en éliminant les clauses triviales contenant deux litéraux complémentaires ainsi que les clauses redondantes.

(5.b) Mise en forme clausale

La méthode de résolution est en général utilisée lorsque le problème à résoudre est déjà en forme clausale (conjonctive normale) Cependant, il est possible de traiter des problèmes qui ne sont énoncés en forme clausale sans pour autant avoir à les reformuler en forme conjonctive normale (ce qui donne en soi une procédure de décision).

En effet, on peut, par un artifice, associer à toute formule A une formule $\tilde{A}$, en forme clausale, telle que $\tilde{A}$ soit équivalente à A. Plus précisemment, il suffit d'avoir qui si $\tilde{A}$ est valide alors A l'est aussi.
On associe à chaque formule non atomique A une variable propositionelle p_A. Si A est une variable p alors $p_A = p$.On définit alors, par induction sur A, l'ensemble $\mathcal{C}(A)$, en posant, pour chaque sous formule $A_1 \star A_2$ ou $\neg A_1$ de A, les équivalences $p_{A_1\star A_2} \Leftrightarrow p_{A_1} \star p_{A_2}$ ou $p_{\neg A_1} \Leftrightarrow \neg p_{A_1}$ mises sous forme clausale.

$$\begin{array}{rcl}
\mathcal{C}(p) & = & \emptyset \\
\mathcal{C}(\neg A) & = & \mathcal{C}(A) \cup \{\{\overline{p}_A, \overline{p}_{\neg A}\}, \{p_A, p_{\neg A}\}\} \\
\mathcal{C}(A \vee B) & = & \mathcal{C}(A) \cup \mathcal{C}(B) \cup \{\{\overline{p}_A, p_{A\vee B}\}, \{p_B, p_{A\vee B}\}, \{\overline{p}_{A\vee B}, p_A, p_B\}\} \\
\mathcal{C}(A \wedge B) & = & \mathcal{C}(A) \cup \mathcal{C}(B) \cup \{\{\overline{p}_{A\wedge B}, p_A\}, \{\overline{p}_{A\wedge B}, p_B\}, \{\overline{p}_A, \overline{p}_B, p_{A\wedge B}\}\} \\
\mathcal{C}(A \Rightarrow B) & = & \mathcal{C}(A) \cup \mathcal{C}(B) \cup \{\{\overline{p}_A, p_B, \overline{p}_{A\Rightarrow B}\}, \{p_A, p_{A\Rightarrow B}\}, \{\overline{p}_B, p_{A\Rightarrow B}\}\} \\
\mathcal{C}(A \Leftrightarrow B) & = & \mathcal{C}(A) \cup \mathcal{C}(B) \cup \{\{\overline{p}_A, p_B, \overline{p}_{A\Leftrightarrow B}\}, \{p_A, \overline{p}_B, \overline{p}_{A\Leftrightarrow B}\}, \{\overline{p}_A, \overline{p}_B, p_{A\Leftrightarrow B}\}, \{p_A, p_B, p_{A\Leftrightarrow B}\}\}
\end{array}$$

(5.c) Un algorithme

Nous utiliserons les symboles $\bot$ pour signaler qu'une clause contradictoire a été engendrée et $\top$ pour signaler qu'une clause triviale a été rencontrée.

Soit l un litéral et C une clause. On appelle R_0 la fonction définie par

$$\begin{array}{rcll}
R_0(l, C) & = & \bot, & \text{si } C = \{\overline{l}\} \\
R_0(l, C) & = & C', & \text{si } \overline{l} \in C \qquad \text{lcll} \\
 & = & \emptyset, & \text{sinon.}
\end{array}$$

On définit alors R, la fonction R qui tente de résoudre deux clauses C_1 et C_2 :

Si $C_1 = \emptyset$ alors $R(C_1, C_2) = \emptyset$
sinon, $C_1 = \{l\} \cup C_1'$,
  soit $C_2' = R_0(l, C_2)$:
    si $C_2' = \bot$ alors $R(C_1, C_2) = \bot$;
    si $C_2' = \emptyset$ alors $R(C_1, C_2) = \emptyset$;
    sinon, si il existe l' tel que $\{l', \overline{l'}\} \subseteq C_1' \cup C_2'$ alors $R(C_1, C_2) = \top$;
      sinon $R(C_1, C_2) = C_1' \cup C_2'$.

Soit Γ un ensemble de clauses ne contenant ni clause contradictoire, ni clause triviale. Soit C une clause. On définit R^* la fonction d'itération de R pour C sur Γ :

Si $\Gamma = \emptyset$ alors $R^*(C, \Gamma) = \emptyset$
sinon, $\Gamma = \{C'\} \cup \Gamma'$,

soit alors $C'' = R(C, C')$:
si $C'' = \bot$ alors $R^*(C, \Gamma) = \bot$;
si $C'' = \top$ alors $R^*(C, \Gamma) = R^*(C, \Gamma')$;
si $C'' = \emptyset$ alors $R^*(C, \Gamma) = R^*(C, \Gamma')$;
sinon $R^*(C, \Gamma) = \{C''\} \cup R^*(C, \Gamma')$.

On définit alors RR^* sur deux ensembles de clauses Γ_1 et Γ_2

Si $\Gamma_1 = \emptyset$ alors $RR^*(\Gamma_1, \Gamma 2) = \emptyset$
sinon, $\Gamma_1 = \{C\} \cup \Gamma'_1$,
soit alors $\Gamma'_2 = R^*(C, \Gamma_2)$:
si $\Gamma'_2 = \bot$ alors $RR^*(\Gamma_1, \Gamma 2) = \bot$;
sinon, $RR^*(\Gamma_1, \Gamma 2) = \Gamma'_2 \cup RR^*(\Gamma'_1, \Gamma 2)$

On définit enfin T sur deux ensembles de clauses Γ_1 et Γ_2

Soit $\Gamma'_1 = RR^*(\Gamma_1, \Gamma_2)$,
soit $\Gamma'_2 = RR^*(\Gamma_1, \Gamma_1)$:
si $\Gamma'_1 \cup \Gamma'_2 = emptyset$ alors $T(\Gamma'_1, \Gamma'_2) = \Gamma_1 \cup \Gamma_2$
sinon $T(\Gamma'_1, \Gamma'_2) = T(\Gamma_1 \cup \Gamma_2, \Gamma'_1 \cup \Gamma'_2)$.

Soit A une formule et Γ la forme clausale de $\neg A$, si $T(\emptyset, \Gamma) = \bot$ alors A est une tautologie.

II - Le calcul des prédicats

Termes, formule, validité

§ 1. Les tableaux sémantiques

On peut, pour le calcul des prédicats, reconduire la technique des tableaux sémantiques. L'ajout des quatification existentielles et universelles se traduit par l'ajout de deux nouveaux types pour les formules : δ et γ. Ces deux types de formule donnent toutes deux lieu à des règles de prolongement. Elles se différencient en ce qu'elles s'adressent à deux types de quantifiation : l'existentielle et l'universelle.

(1.a) Les règles d'analyse

En calcul des propositions, nous décomposions les formules en leurs sous-formules. Ici, en calcul des prédicats nous utiliserons, pour ce qui est des formules quatifiées, des *instances* plutôt que des sous-formules. Pour ce, nous serons amenés à enrichir le *vocabulaire* des notre langage de termes en introduisant de nouvelles constantes d'individus : on travaillera donc en fonction d'un vocabulaire courrant $\mathcal{V}$ contenant le *"domaine de Herbrand"* découvert jusqu'à présent.

Formules δ	
$\exists x.\Phi^\top$	$\forall x.\Phi^\bot$
$\downarrow$	$\downarrow$
$\Phi[c/x]^\top$	$\Phi[c/x]^\bot$
avec $c \notin \mathcal{V}$ et $\mathcal{V}$ est enrichi avec c	

Formules γ	
$\forall x.\Phi^\top$	$\exists x.\Phi^\bot$
$\downarrow$	$\downarrow$
$\Phi[z/x]^\top$	$\Phi[z/x]^\bot$
pour n'importe quel z	

(1.b) Du bon usage des règles

On remarquera que l'on peut utiliser *ad libitum* ces règles pour produire de nouvelles formules, ce qui n'était pas le cas pour les règles α et β du calcul propositionnel. Cependant, il sera inutile de créer plus d'une instance des formules δ (l'existence d'un témoin suffit aux existentielles) De même, il sera inutile d'engendrer

indéfiniment de nouvelles constantes pour instancier les formule γ, mieux vaudra utiliser le vocabulaire initial ainsi que celui issu du traitement des formules δ[1] On définira donc une *stratégie* d'application des règles.

Lorsque l'on itère le processus d'application des règles, plusieurs cas de figure pourront se produire :

1. on peut clore une branche contenant deux formules atomiques se signes opposés ;
2. ne restent à traiter que des formules de type γ qui ont déjà toutes été instanciées, dans ce cas, on trouve sur la branche une interprétation satisfaisant la formule de départ ;
3. aucun des deux cas précédant ne se produit et le processus se poursuit à l'infini, le calcul des prédicats est indécidable.

(1.c)

Termes/Formules.

§ 1. Unification

On dit que deux termes t et u sont *unifiables* si et seulement s'il existe une substitution σ telle que $\sigma(t) = \sigma(u)$. On appelle alors σ un *unificateur* de t et u. Le problème d'*unification* est celui qui consiste à chercher décider de l'unifiabilité de deux termes.

Le problème de l'unification est décidable (sur les termes du premier ordre). Mieux, si deux termes sont unifiables, il existe un *unificateur le plus général* (*mgu* en bon franglais) c'est à dire qu'il existe une substitution σ telle que $\sigma(t) = \sigma(u)$ et pour tout autre unificateur τ, il existe une substitution ρ telle que, pour toute variable x, $\tau(x) = \rho(\sigma(x))$ (on dit que τ est une *instance* de σ).

(1.a) Algorithmes

Si t est un terme, on note $\mathcal{V}(t)$ l'ensemble de ses variables. On a pris l'habitude de présenter les procédures d'unification par un ensemble de règles, que l'on notera $E \longrightarrow E'$, transformant un ensemble d'équation transformant un ensemble d'équations E en un nouvel ensemble E'. On itère alors l'application des règles jusqu'à saturation ou échec. Si E est un ensemble d'équations saturés alors tout élément de E est de la forme $x = t$ où x est une variable. Il y a deux causes d'échec au processus d'unification : ou bien E contient une équation de la forme $x = t$ et $x \in \mathcal{V}(t)$, on parle alors d'*"occur check"* ; ou bien E contient une équation de la forme $f(t_1, .., t_n) = g(u_1, .., u_m)$ et $f \neq g$, on parle alors de *"clash"*.

Nous donnons un premier ensemble de règles inspirés de l'algorithme original de robinson.

Delete	$\{t = t\} \cup E$	$\longrightarrow$	E	–
Check	$\{x = t\} \cup E$	$\longrightarrow$	occur-check	, si $x \in \mathcal{V}(t)$
Eliminate	$\{x = t\} \cup E$	$\longrightarrow$	$\{x = t\} \cup E[t/x]$	, si $x \notin \mathcal{V}(t)$ et $x \in \mathcal{V}(E)$
Decomp	$\{f(t_1, .., t_n) = f(u_1, .., u_n)\} \cup E$	$\longrightarrow$	$\{t_1 = u_1, .., t_n = u_n\} \cup E$	–
Clash	$\{f(t_1, .., t_n) = g(u_1, .., u_m)\} \cup E$	$\longrightarrow$	clash	, si $f \neq g$

Chaque règle fait décrître une quatité : la taille de l'ensemble (Delete), la somme de la taille des termes (Decomp) ou le nombre d'occurence d'une variable (Eliminate)

Le second ensemble de règles est inspiré d'une autre algorithme d'unification dû à Martelli et Montanari. Plutôt que de saturer un ensemble d'équation, cet algorithme construit une suite de couples (x, t) dits *liaison* de la variable x au terme t. Si $\sigma = [(x_1, t_1), .., (x_n, t_n)]$, on note $t\sigma$ le terme $t[x_1, t_1]..[x_n, t_n]$. Si x est une variable, on note $x \in \sigma$ pour $\exists (x, t) \in \sigma$
Les règles sont de la forme $E, \sigma \longrightarrow E', \sigma'$ où E et E' sont des ensembles d'équations et σ, σ' des suites de

[1] Le seul cas où une formule γ pourra donner lieu à la génération de nouveaux symbole est celui où le vocabulaire est resté vide alors qu'il n'existe d'autres formules que celles de type γ

liaisons.

Delete	$\{t = t\} \cup E, \sigma$	$\longrightarrow$	E, σ	–
Decomp	$\{f(t_1, .., t_n) = f(u_1, .., u_n)\} \cup E, \sigma$	$\longrightarrow$	$\{t_1 = u_1, .., t_n = u_n\} \cup E, \sigma$	–
Merge	$\{x_i = t\} \cup E, \sigma$	$\longrightarrow$	$\{t_i = t\} \cup E, \sigma$	, si $x_i \neq t$ et $(x_i, t_i) \in \sigma$
Bind	$\{x = t\} \cup E, \sigma$	$\longrightarrow$	$E, (x, t).\sigma$	, si $x \neq t$ et $x \notin \sigma$ et $x \notin \mathcal{V}(t\sigma)$

L'argument de terminaison est analogue à celui de l'ensemble de règles précédent. Si plus aucune règle n'est applicable et E est non vide alors l'unification échoue.