Licence Informatique, semestre 5 — 2019–20
Eléments de logique pour l'informatique (Info 315) — 7 octobre 2019
`http://www.lri.fr/~paulin/Logique`

Feuille 5 - Calcul des séquents et résolution propositionnelle

Exercice 1 *Séquents valides.*

Donner pour chacun des séquents suivants la formule associée. Dire si les séquents sont valides.

1. $(p \vee q \Rightarrow r), \neg(p \wedge z) \vdash \neg p, \neg z$
2. $\vdash (r \wedge s), q$
3. $(p \vee q) \Rightarrow r, \neg(p \vee z) \vdash$
4. $\vdash$
5. $\perp \vdash$

Exercice 2 *Preuve de séquents.*

On se donne les séquents suivants
— $S_1 : r, s \vdash (r \Rightarrow (\neg p \Rightarrow q)) \wedge ((q \vee \neg p) \Rightarrow r)$
— $S_2 : \vdash (p \Rightarrow q) \Rightarrow (p \wedge \neg q) \Rightarrow (r \wedge \neg r)$
— $S_3 : q, r \Rightarrow p \vdash \neg(p \Rightarrow q) \wedge r, \neg q \Rightarrow \neg r, p \wedge q \wedge (\neg p \vee q)$

Pour chacun des séquents précédents :

1. Ecrire la formule logique associée.
2. En utilisant le système G, déterminer si le séquent est valide. Que peut-on en conclure pour la formule logique associée ? Si la formule n'est pas valide, donner une interprétation dans laquelle la formule est évaluée à faux.

Exercice 3 *Correction des règles du système G.*

Les règles pour la négation dans le système G sont :

$$\frac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta} \qquad \frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A}$$

Montrer que ces deux règles sont correctes et réversibles, c'est-à-dire qu'une interprétation rend vrai la conclusion (séquent "sous la barre") si et seulement si elle rend vrai la ou les prémisses (séquent "au-dessus" de la barre).

Exercice 4 On s'intéresse ici à des formules qui contiennent le symbole d'équivallence $p \Leftrightarrow q$. On rappelle que $p \Leftrightarrow q$ est défini comme $(p \Rightarrow q) \wedge (q \Rightarrow p)$. Dans la suite x, y et z désignent trois variables propositionnelles.

1. Donner la forme normale conjonctive des formules $(x \Leftrightarrow y)$ et $\neg(x \Leftrightarrow y)$.
2. Construire la table de vérité de la formule $(x \Leftrightarrow (y \Leftrightarrow z))$. Cette formule est-elle valide ? satisfiable ?
3. On considère maintenant des séquents avec des formules contenant le symbole d'équivalence. Soit les deux règles dans lesquelles Γ et Δ sont des ensembles de formules et p et q sont des formules :

$$\frac{\Gamma, p \vdash \Delta, q \qquad \Gamma, q \vdash \Delta, p}{\Gamma \vdash \Delta, p \Leftrightarrow q} \qquad \frac{\Gamma, p, q \vdash \Delta \qquad \Gamma \vdash \Delta, p, q}{\Gamma, p \Leftrightarrow q \vdash \Delta}$$

   (a) Montrer que ces deux règles sont correctes (si une interprétation satisfait les deux prémisses alors elle satisfait la conclusion de la règle).
   (b) Montrer qu'elles sont inversibles (si une interprétation satisfait la conclusion de la règle alors elle satisfait les deux prémisses).
   (c) Construire en utilisant ces règles un arbre de preuve pour le séquent $\vdash x \Leftrightarrow (y \Leftrightarrow (x \Leftrightarrow y))$.

(d) La formule $x \Leftrightarrow (y \Leftrightarrow (x \Leftrightarrow y))$ est-elle valide ?

Exercice 5 Soient E_1 et E_2 les ensembles de formules suivants :

$$E_1 = \{p \Leftrightarrow q, \neg(p \Rightarrow q) \vee (\neg(q \Rightarrow p))\} \qquad\qquad E_2 = \{(p \Rightarrow q), \neg q, \neg(q \Rightarrow p)\}$$

1. Construire pour E_1 puis pour E_2, les ensembles de clauses associés.
2. Chercher une réfutation pour chacun de ces ensembles.

Exercice 6 On considère les formules
— $A \stackrel{\text{def}}{=} (x \Leftrightarrow z) \wedge y$
— $B \stackrel{\text{def}}{=} (y \Leftrightarrow z) \Rightarrow x$

1. Mettre les formules A et $\neg B$ en forme clausale.
2. En utilisant la résolution, montrer que la formule B est conséquence logique de A.

Exercice 7 On se place dans le calcul propositionnel. On suppose que l'on dispose des programmes suivants :
— `deductionG` qui étant donné une formule A du calcul propositionnel renvoie vrai si et seulement si on peut construire dans le système G un arbre de dérivation complet dont la racine est le séquent avec la formule A comme conclusion ;
— `refutation` qui étant donné un ensemble de clauses propositionnelles E renvoie vrai si et seulement si on peut déduire par résolution à partir de E la clause vide $\perp$;
— `sat` qui étant donné un ensemble de clauses propositionnelles E renvoie vrai si et seulement si on peut construire une interprétation qui rend vraies l'ensemble des clauses de E ;
— `clausale` qui étant donnée une formule A renvoie l'ensemble des clauses correspondant à sa forme normale conjonctive.

1. Proposer trois solutions différentes pour implémenter une fonction `valide`(A) qui teste si la formule A est valide en utilisant respectivement les fonctions `deductionG`, `refutation` et `sat`.
2. Faire la même chose pour implémenter une fonction `satisfiable`(A) qui teste si la formule A est satisfiable et `insatisfiable`(A) qui teste si la formule A est insatisfiable.

Exercice 8 Nous rappelons la règle de résolution où C et C' sont des clauses, p une variable propositionnelle :

$$\frac{C \vee \neg p \quad C' \vee p}{C \vee C'}$$

On rappelle que les clauses sont vues comme des ensembles de littéraux et donc lorsque l'on écrit $C \vee C'$ on ne garde qu'une seule instance de chaque littéral. Si la clause contient une variable propositionnelle et sa négation elle se simplifie en $\top$. Pour avoir moins d'étapes dans une preuve on propose d'utiliser la nouvelle règle suivante :

$$\frac{C \vee \neg p \vee \neg q \quad C' \vee p \vee q}{C \vee C'}$$

où q est une variable propositionnelle différente de p.
Cette règle est-elle correcte ? Pourquoi ?

Exercice 9 *SAT-solver élémentaire.*

La procédure DPLL (du nom de ses inventeurs Davis, Putnam, Logemann et Loveland) permet de dire si une formule en forme normale conjonctive (représentée par un ensemble de clauses) est satisfiable. Pour cela, l'algorithme cherche à construire une interprétation qui rend la formule vraie.
L'algorithme DPLL travaille sur des problèmes de la forme : $I \gg \Delta$ avec I une interprétation partielle des variables propositionnelles de la forme $\{x_1 \mapsto b_1; \ldots; x_n \mapsto b_n\}$ (que l'on peut aussi représenter comme un ensemble de littéraux x_i si $b_i = V$ et $\neg x_i$ si $b_i = F$) et Δ un ensemble de clauses $C_1, \ldots, C_n$. On note Δ, C le problème qui contient toutes les clauses de Δ plus la clause C. Le but de l'algorithme est de construire une

interprétation qui étend I, c'est-à-dire de la forme $\{x_1 \mapsto b_1; \ldots; x_n \mapsto b_n; x_{n+1} \mapsto b_{n+1} \ldots x_p \mapsto b_p\}$ et qui rend vraies toutes les clause C_i. Les règles sont les suivantes :

Il y a deux cas triviaux. Le premier cas s'il n'y a plus de clauses, l'interprétation I répond au problème. Le second cas si l'une des clauses C_i est la clause vide alors le problème n'a pas de solution. Cela correspond aux deux règles :

$$\textsc{Success}\frac{}{I \gg \emptyset} \qquad \textsc{Conflict}\frac{}{I \gg \Delta, \bot}$$

Par ailleurs on peut simplifier les problèmes : lorsque l'interprétation I fixe la valeur de la variable x, alors les clauses C qui contiennent un litteral l de la forme x ou $\neg x$ se simplifient :

— si $I(l) = V$ alors la clause est trivialement vraie dans cette interprétation et peut être supprimée ;

— si $I(l) = F$ alors on peut retirer le littéral l de la clause.

Ceci est représenté par les deux règles suivantes :

$$\textsc{Bcp}_V\frac{I \gg \Delta \qquad I(l) = V}{I \gg \Delta, (l \vee C)} \qquad \textsc{Bcp}_F\frac{I \gg \Delta, C \qquad I(l) = F}{I \gg \Delta, (l \vee C)}$$

Maintenant si aucune des règles précédentes ne s'applique, alors il faut préciser l'interprétation en choisissant une valeur pour une nouvelle variable. Un cas simple est si une clause ne contient qu'un seul littéral alors pour que la clause soit vraie il faut choisir la valeur de la variable pour rendre ce littéral vrai. Ceci s'exprime par les règles suivantes :

$$\textsc{Assume}_V\frac{I + \{x \mapsto V\} \gg \Delta \qquad x \notin I}{I \gg \Delta, x} \qquad \textsc{Assume}_F\frac{I + \{x \mapsto F\} \gg \Delta \qquad x \notin I}{I \gg \Delta, \neg x}$$

Si maintenant, il n'y a aucune clause réduite à un littéral, alors il n'y a pas de choix évident. On va donc choisir une variable qui n'a pas encore de valeur et explorer les deux branches : celle où cette variable est interprétée par V et celle où elle est interprétée par F.

$$\textsc{Unsat}\frac{I + \{x \mapsto V\} \gg \Delta \qquad I + \{x \mapsto F\} \gg \Delta \qquad x \notin I}{I \gg \Delta}$$

On part d'un ensemble de clauses Δ et d'une interprétation vide. On construit un arbre en utilisant les règles précédentes et on cherche à arriver à une feuille de succès de la forme $I \gg \emptyset$. L'interprétation I est une solution de notre problème, elle rend vraie l'ensemble des clauses Δ.

Si toutes les feuilles de l'arbre correspondent à des conflits alors l'ensemble de clauses initial n'est pas satisfiable.

1. Appliquer l'algorithme précédent aux formules
   (a) $(\neg p \vee \neg q \vee \neg r) \wedge (\neg p \vee q \vee r) \wedge (p \vee q \vee \neg r)$
   (b) $(p \vee r) \wedge (\neg q \vee p) \wedge \neg p \wedge \neg r$
2. Justifier que la construction de l'arbre par les règles DPLL est toujours finie (trouver une quantité qui diminue le long des branches de l'arbre).
3. Montrer que les règles de DPLL sont correctes. C'est-à-dire que si un arbre DPLL de racine $I \gg \Delta$ contient une feuille de succès $J \gg \emptyset$ alors l'interprétation J étend l'interprétation I (c'est-à-dire que $I(x) = J(x)$ pour toutes les variables x qui apparaissent dans I) et l'interprétation J rend vraies toutes les clauses de Δ.
4. En déduire un méthode pour trouver un modèle d'une formule.
5. A votre avis, la méthode est-elle complète, c'est-à-dire s'il existe un modèle est-on sûr de le trouver ? Peut-on trouver avec cette méthode tous les modèles d'une formule ?
6. Peut-on utiliser cette méthode pour prouver la validité d'une formule ?