Cours logique - Mémo n°4

Problème SAT et algorithme DPLL

Emmanuel Coquery

1 Formes conjonctives et disjonctives

Notation: Si $A_1, \ldots, A_n$ sont des formules, on utilise les notations suivantes :

$$\bigwedge_{i=1}^{n} A_i = A_1 \wedge \ldots \wedge A_n \quad \text{et} \quad \bigvee_{i=1}^{n} A_i = A_1 \vee \ldots \vee A_n$$

Si $n = 0$:

$$\bigwedge_{i=1}^{0} A_i = \top \quad \text{et} \quad \bigvee_{i=1}^{0} A_i = \bot$$

Si $n = 1$:

$$\bigwedge_{i=1}^{1} A_i = A_1 \quad \text{et} \quad \bigvee_{i=1}^{1} A_i = A_1$$

Définition 1 *Un* littéral *est une formule atomique ou la négation d'une formule atomique*

Définition 2 *Une* formule conjonctive *est une formule de la forme :* $\bigwedge_{i=1}^{m} \bigvee_{j=1}^{m} A_i^j$ *où les* A_i^j *sont des littéraux.*

Définition 3 *Une* formule disjonctive *est une formule de la forme :* $\bigvee_{i=1}^{m} \bigwedge_{j=1}^{m} A_i^j$ *où les* A_i^j *sont des littéraux.*

Attention à une pas confondre *formule conjonctive* et *conjonction* (c.-à.-d. une formule de la forme $\bigwedge_{i=1}^{n} A_i$, où les A_i peuvent être n'importe quelle formule). De même, il ne faut pas confondre *formule disjonctive* et *disjonction*. Cependant, on peut remarquer qu'une formule conjonctive est un cas particulier de conjonction (en fait c'est une conjonction de disjonctions de littéraux).

Notation: Lorsque l'on écrit une conjonction de formules, on pourra omettre la formule $\top$ (car $A \wedge \top \equiv A$). De même, lorsque l'on écrit une disjonction de formules, on pourra omettre la formule $\bot$.

Propriété 1 *Pour toute formule* A*, il existe une formule conjonctive* A' *et une formule disjonctive* A'' *telles que* $A \equiv A' \equiv A''$

Preuve: On comme par éliminer de A les symboles $\Rightarrow$ et $\Leftrightarrow$ en utilisant les équivalences suivantes :

$B \Rightarrow C \equiv \neg B \vee C$

$B \Leftrightarrow C \equiv (B \Rightarrow C) \wedge (C \Rightarrow B)$

On obtient alors une formule équivalente à A et qui ne contient que $\wedge$, $\vee$, $\neg$, $\top$, $\bot$ et des variables propositionnelles. Dans la suite de la preuve, on supposera que A ne contient que $\wedge$, $\vee$, $\neg$, $\top$, $\bot$ et des variables propositionnelles (c.-à.-d. que $\Rightarrow$ et $\Leftrightarrow$ on déjà été élmininés).

On procède ensuite par induction sur A pour montrer l'existence de A' et A'' :

- Si A est un littéral, alors elle est déjà en forme conjonctive et en forme disjonctive.
- Si $A = B \vee C$, alors, par induction, il existe des formules conjonctives B' et C' et des formules disjonctives B'' et C'' telles que $B' \equiv B'' \equiv B$ et $C' \equiv C'' \equiv C$. On peut remarquer que $A'' = B'' \vee C''$ est une formule disjonctive équivalente à A.

  Construisons à présent une formule conjonctive équivalente à A. Pour cela, on remarque tout d'abord que $B' = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n} B_i'^j$ et $C' = \bigwedge_{k=1}^{p} \bigvee_{l=1}^{q} C_k'^l$. En appliquant les règles de distributivité sur la formule $A' \vee B'$, on obtient :

$$A' = \bigwedge_{i=1}^{m} \bigwedge_{k=1}^{p} (\bigvee_{j=1}^{n} B_i'^j \vee \bigvee_{l=1}^{q} C_k'^l)$$

  On peut remarquer que A' est une formule conjonctive équivalente à A.
- Si $A = B \wedge C$, la preuve est similaire au cas précédent.
- Si $A = \neg B$, alors par hypothèse d'induction, il existe $B' = \bigwedge_{i=1}^{n} \bigvee_{j=1} B_i'^j$ avec $B' \equiv B$. En utilisant les lois de De Morgan, on peut faire entrer la négation à l'intérieur des $\wedge$ et des $\vee$, puis éliminer les doubles négations (de la forme $\neg\neg B_j'^i$). On obtient alors la formule disjonctive $A'' = \bigvee_{i=1}^{n} \bigwedge_{j=1} A_i''^j$ avec $A_i''^j = C_i^j$ si $B_i^j = \neg C_i^j$ pour un certain C_j^i et $A_i''^j = \neg B_i^j$ sinon. On peut remarquer que $A'' \equiv A$. On peut également construire de manière similaire A' à partir de B'', où B'' est la forme disjonctive de B.

□

La transformation utilisée dans la preuve précédente peut introduire une formule conjonctive de taille exponentielle en présence de nombreuses disjonctions. En relâchant la contrainte d'équivalence, on peut obtenir une formule de taille linéaire en fonctionde la taille de la formule initiale.

Définition 4 *Soit A et B deux formules. Ces formules sont dites* équisatisfiables *si A est satisfiable si et seulement si B l'est.*

Propriété 2 *Soit A une formule. Il existe une formule conjonctive B telle que :*

1. *A et B sont équisatisfiables*
2. *La taille de B est linéaire en fonction de la taille de A.*

Preuve: On montre par induction que l'on peut introduire la formule B de la forme $p \wedge C$ où p n'apparaît pas dans C et C est une formule conjonctive telle que pour toute interprétation I, on peut trouver I' telle que $[C]_{I'} = [p \Leftrightarrow A]_{I'}$ et $I_{|V(p \Leftrightarrow A)} = I'_{|V(p \Leftrightarrow A)}$. La deuxième partie implique que $[p \Leftrightarrow A]_{I'} = [p \Leftrightarrow A]_I$.

Il est aisé de vérifier que B et A sont équisatisfiables : si A est satisfiable, alors il suffit de prendre v comme valeur pour p. Si A n'est pas satisfiable, soit I une interprétation quelconque. Si $I(p) = f$, alors $[B]_I = f$. Sinon, $I(p) = v$. Comme A est insatisfiable, $[A]_I = f$, et donc $[p \Leftrightarrow A]_I = f$, d'où $[B]_I = f$.

Montrons à présent l'existence de B par induction sur A.

- Si $A = \bot$, on pose $B = p \wedge \neg p$, où p est une variable fraîche [1]
- Si $A = \top$, on pose $B = p \wedge p$ où p est fraîche.

1. c'est-à-dire une variable qui n'est pas dans l'ensemble des variables considérées jusqu'ici

- Si $A = p$, on pose $B = p$
- Si $A = \neg A'$, alors soit $B' = p' \wedge C'$ obtenue par induction sur A'. Soit p une variable fraîche. On pose $B = p \wedge (\neg p \vee \neg p') \wedge (p \vee p') \wedge C'$.
- Si $A = A' \vee A''$, alors soient $B' = p' \wedge C'$ et $B'' = p'' \wedge C''$ obtenues par induction à partir de B et C respectivement. Soit p une variable fraîche. On pose $B = p \wedge (\neg p \vee p' \vee p'') \wedge (p \vee \neg p') \wedge (p \vee \neg p'') \wedge C' \wedge C''$.
- Si $A = A' \wedge A''$, alors soient $B' = p' \wedge C'$ et $B'' = p'' \wedge C''$ obtenues par induction à partir de B et C respectivement. Soit p une variable fraîche. On pose $B = p \wedge (p \vee \neg p' \vee \neg p'') \wedge (\neg p \vee p') \wedge (\neg p \vee p'') \wedge C' \wedge C''$.
- Si $A = A' \Rightarrow A''$, alors soient $B' = p' \wedge C'$ et $B'' = p'' \wedge C''$ obtenues par induction à partir de B et C respectivement. Soit p une variable fraîche. On pose $B = p \wedge (\neg p \vee \neg p' \vee p'') \wedge (p \vee p') \wedge (p \vee \neg p'') \wedge C' \wedge C''$.
- Si $A = A' \Leftrightarrow A''$, alors soient $B' = p' \wedge C'$ et $B'' = p'' \wedge C''$ obtenues par induction à partir de B et C respectivement. Soit p une variable fraîche. On pose $B = p \wedge (\neg p \vee \neg p' \vee p'') \wedge (\neg p \vee p' \vee \neg p'') \wedge (p \vee p' \vee p'') \wedge (p \vee \neg p' \vee \neg p'') \wedge C' \wedge C''$.

□

Définition 5 *Une* clause *est une disjonction de littéraux.*

Une formule conjonctive est donc une conjonction de clauses.

Par la suite, on identifiera chaque clause avec toute autre clause obtenue en permutant ses littéraux, en éliminant des littéraux déjà présents dans la clause u en éliminant $\bot$ et $\neg\top$ de la clause. Par exemple, on identifiera $r \vee \neg q \vee r \vee \bot \vee \neg p$ avec $\neg p \vee \neg q \vee r$.

La clause vide (c.-à-d. une disjonction vide de littéraux, donc la formule $\bot$) sera également notée $\square$.

2 Problème SAT

Le problème SAT consiste à déterminer si une formule est satisfiable. Ce problème est dit NP-Complet : il peut être résolu en temps polynomial par une machine non-déterministe. Autrement dit, on sait déterminer en temps polynomial si une "solution candidate" (ici une interprétation) permet d'affirmer que la formule est satisfiable.

Le principe des algorithmes de test de satisfiabilité complets par énumération consiste à vérifier la valeur de vérité de la formule par rapport aux 2^n interprétations possibles [2] pour les n variables de la formule.

L'ensemble de ces interprétations constitue *l'espace de recherche* du problème. Les algorithmes par énumération cherchent ainsi à explorer intelligemment cet espace de recherche en tentant de minimiser l'impact de l'exponentielle. Ainsi certains algorithme essaient de prévoir l'impact d'un choix partiel des valeurs pour les variables, par exemple en propageant les valeurs des variables affectées dans la valeur de vérité de la formule pour voir si cela ne la rend pas directement (in)satisfiable.

Remarque: L'algorithme consistant à tenter de dériver le séquent $A \vdash$ pour montrer l'insatisfiabilité d'une formule ne procède pas par exploration de l'espace des interprétations. Il n'est cependant pas à l'abris d'un explosion exponentielle en temps d'exécution, par exemple si A est une forme conjonctive.

2. si on se restreint à des interprétations deux à deux différentes sur au moins une des variables de la formule

2.1 Arbre de recherche

Une approche classique pour énumérer les différentes interprétations de l'espace de recherche associé à une formule A consiste à parcourir un *arbre de recherche* :

- Chaque noeud interne de l'arbre correspond à une variable.
- Chaque feuille correspond à une interprétation I, éventuellement partielle, pour laquelle on connaît la valeur de vérité $[A]_I$. Plus précisément, cette valeur est calculable en temps polynomial dans la taille de A.
- Chaque noeud interne [3] possède deux fils, un correspondant à des interprétations où la variable prend la valeur vrai, l'autre où elle prend la valeur faux.
- Quelque soit la branche, elle ne contient pas deux noeuds correspondant à la même variable.

Remarque: Le parcours complet de l'arbre de recherche n'est typiquement nécessaire que si la formule n'est pas satisfiable, auquel cas $[A]_I = f$ pour toutes les feuilles. Si la formule est satisfiable, on peut s'arrêter sur la première feuille telle que $[A]_I = v$.
Remarque: Il existe de nombreux arbres de recherche pour une formule A donnée. En pratique, l'arbre de recherche n'est pas fixé à l'avance mais est construit (virtuellement) au fur et à mesure de son exploration.

Par la suite, on appellera *affectation courante* pour un noeud de l'arbre de recherche l'affectation des valeurs aux variables correspondant à la branche qui va de la racine au noeud en question.

3 DPLL

L'algorithme DPLL, du nom de ses auteurs Davis, Putnam, Logemann et Loveland constitue la base des solveurs SAT modernes. Cet algorithme s'applique sur une formule en forme conjonctive, c'est-à-dire sur un ensemble de clauses.

3.1 Propagation unitaire

La propagation unitaire est un moyen de prendre en compte l'affectation d'une variable à une valeur selon les deux principes suivants :

1. éliminer des clauses les littéraux faux vis-à-vis de l'affectation courante des valeurs aux variables ;
2. pour les clauses unitaires, c'est-à-dire les clauses possédant un seul littéral, forcer l'affectation de la variable de façon à rendre la clause vraie.

L'algorithme DPLL va appliquer la propagation unitaire à chaque noeud de l'arbre de recherche. De plus, si certaines variables se voient affectées une valeur lors de ce processus, la propagation unitaire leur est également appliquée, jusqu'à ce que plus aucune variable ne se voit affectée de valeur par ce biais.

3.2 Algorithme DPLL

Fonction DPLL(A) :
  Appliquer la propagation unitaire, de manière itérative
    jusqu'à obtention d'un point fixe
  Si A ne contient que des clauses satisfaites
    renvoyer SAT
  Si A contient la clause vide
    renvoyer $UNSAT$
  Choisir une variable I
  Renvoyer $DPLL(A \cup \{I\}) \vee DPLL(A \cup \{\neg I\})$

3. y compris la racine

En pratique, on ne duplique pas l'ensemble A pour y ajouter une clause (lors de l'appel récursif), ni pour en supprimer une (lors de la propagation unitaire). On garde simplement trace des modifications de façon à pouvoir les défaire. L'action de défaire les modifications est appelée *retour arrière* ou *backtrack*.

Pour calculer le point fixe de la propagation unitaire, on peut maintenir une liste de littéraux à affecter à vrai, initialisée avec des littéraux des clauses unitaires. Chaque fois qu'une clause unitaire est déduite, si le littéral est déjà affecté à faux, on peut renvoyer UNSAT, sinon on l'ajoute à la liste s'il n'est pas déjà affecté à vrai. On retire ensuite un littéral de la liste, que l'on affecte à vrai et on propage sa valeur dans les clauses déduisant ainsi éventuellement d'autres clauses unitaires. On itère cette dernière étape jusqu'à obtenir une liste vide.

3.3 Indexation des clauses

Afin d'optimiser la propagation unitaire, on peut indexer les clauses par les littéraux qu'elles contiennent, i.e. associer à chaque littéral la liste des clauses qui contiennent ce littéral. Ainsi lorsqu'un littéral est affecté à faux, on obtient directement la liste des clauses à tester.

Cette optimisation peut être poussée plus loin : en effet, tant qu'il reste au moins deux variables non affectées dans une clause, celle-ci ne peut être réduite à une clause unitaire par propagation de la valeur d'un seul littéral. On peut ainsi se contenter de ne mettre chaque clause que dans la liste de deux littéraux. Ces deux littéraux sont appelés *littéraux surveillés* pour cette clause. Lorsqu'un des deux littéraux, que l'on appelra l_1 est affecté, il y a deux possibilités : s'il est affecté à vrai alors la clause est satisfaite et on ne fait rien. Sinon, la clause n'est satisfiable que si un autre littéral peut être affecté à vrai. On examine alors la clause : si elle contient un littéral affecté à vrai, elle est satisfaite et on ne fait rien. Si tous ses littéraux sont affectés à faux, renvoyer UNSAT. S'il ne reste qu'un seul autre littéral, l_2 non affecté (par définition l_2 est l'autre littéral surveillé), la clause devient unitaire et on ajoute ce littéral à la liste de propagation. Sinon, il reste au moins un littéral l'_1 non affecté qui n'est pas surveillé pour cette clause. l'_1 devient alors le nouveau littéral surveillé de la clause en remplacement au littéral l_1.