kHORN

Joseph Lenormand, Félix Yvonnet

April 2023

1 Introduction

On cherche à proposer un sat solver dans le cas simplifié des clauses de Horn décrit ci-après. Pour cela, nous allons proposer un algorithme en temps $O(n*m)$, avec m le nombre de clauses et n le nombre total de littéraux dans la formule, testant la satisfiabilité d'une formule en forme Horn-NF.

On appelle une clause de Horn une formule logique n'utilisant, sous sa forme normale, que la disjonction ($\vee$) et la négation ($\neg$), et contenant sous ces conditions au plus une clause positive (sans négation). Elles peuvent donc s'écrire $\neg p_1 \vee \cdots \vee \neg p_n \vee q$. Cela correspond intuitivement aux implications. On dit qu'une formule et sous la forme normale de Horn, noté Horn-NF, si c'est une conjonction de clauses de Horn.

2 Algorithme de résolubilité

2.1 Algorithme

function kHornSolver(clauses: List[List[Int]])
  $VarVraies \leftarrow EmptyList$
  loop
    if *clauses* vide **then**
      return Vrai
    end if
    current $\leftarrow$ plus petite clause de *clauses*
    if $|current| = 0$ **then**
      return Faux
    else if $|current| = 1$ **then**
      if $current[0] = \bot$ **then**
        return $Faux$
      else
        ajoute $current[0]$ à $VarVraies$
        supprime toutes les occurences de $\neg current[0]$ dans les autres éléments de *clauses*

```
                supprime toutes les clauses contenant current[0]
            end if
        else
            return Vrai
        end if
    end loop
end function
```

2.2 Preuves

2.2.1 terminaison

Remarquons rapidement que cet algorithme se termine. En effet le nombre total de variables libres décroit strictement à chaque itération de la boucle.

2.2.2 correction

La variable "*clauses*" est la liste des clauses de Horn représentées au format DIMACS où une première liste représente la conjonction des clauses de Horn qui sont elles mêmes représentées par une liste d'entier représentant la disjontion entre littéraux. Remarquons que toute clause de Horn est dans l'un des cas listés ci-après, où $p_1, \cdots, p_n, q$ sont des propositions (possiblement constantes égales à $\top$ ou $\bot$) :

1. une unique clause q
2. une disjonction de négation: $\neg p_1 \vee \cdots \vee \neg p_n$ avec $n \geq 1$
3. le cas global: $\neg p_1 \vee \cdots \vee \neg p_n \vee q$ avec $n \geq 1$

Or les cas 2 et 3 sont satisfiables en prenant $\forall i, p_i = \bot$. Il ne reste donc qu'à fixer q à $\top$ dans toutes les formules de type (1) pour savoir si on trouve des contradictions.

2.2.3 complexité

temporelle: Notons m le nombre de conjonctions dans la formule de Horn et n le nombre total de littéraux dans la formule.
On a donc au plus m littéraux positifs dans la formule.
Or à chaque tour de boucle soit on est dans un état terminal, soit on supprime toutes les clauses contenant le littéral dont on vient de fixer la valeur à $\top$. Le nombre de littéraux positifs est donc strictement décroissant à chaque itération de la boucle. On fait donc au plus m tour de boucle.
Finalement, à l'intérieur de la boucle, toutes les opérations consistent en de la recherche ou de la suppression dans une liste, ce qui se fait en temps linéaire, plus quelques opérations en temps constant.
Ainsi la complexité est en $O(m * n)$.

spatiale: On garde les mêmes définitions pour m et n.
La seule structure qu'on utilise en plus de celles données en argument est la liste $VarVraies$, qui contient au plus m littéraux.
Ainsi, on a une complexité spatiale en $O(m)$.

2.3 Nombre minimal de variables à évaluer à $\top$

Cet algorithme vérifie par ailleurs la propriété demandée en question 4., le nombre minimal de variables à évaluer à $\top$ pour satisfaire la formule correspond à la longueur finale de $VarVraies$ si le résultat envoyé est true.
En effet, le programme étant correct, fixer toutes les variables de $VarVraies$ à $\top$ suffit à vérifier la formule donnée.
Si, par l'absurde, il existe une solution avec strictement moins de variables évaluées à $\top$, alors en particulier, il existe $q \in VarVraies$, telle que q est évaluée à $\bot$ dans ladite solution optimale.
Prenons $\tilde{q}$ la première clause mise à $\top$ dans notre algorithme, telle qu'elle est à $\bot$ dans la solution optimale.
Toutes les variables évaluées à $\top$ avant elle dans l'algorithme sont aussi évaluées à $\top$ dans la solution optimale.
Donc une des clauses, en faisant les simplifications de notre algorithme, est de la forme $\tilde{q}$, puisque $\tilde{q}$ a été traitée après l'évaluation à $\top$ des variables la précédant dans $VarVraies$.
Dès lors, la solution optimale évalue cette clause de la conjonction à $\bot$, soit toute la formule à $\bot$, absurde.
D'où le fait que cet algorithme répond aussi à la question 4.