SAT Solver

1) Présentation:

O sait que 1In3Sat est NP-complet, Dans la suite on a un "poc" pour un algo qui résout cette forme du problème.

On part d'une instance du problème dans la version 1In3saT.

On notera dans la suite !X la variable opposée à X et v l'operateur OU logique.

On utilisera dans la suite 1 ou True pour la valeur logique positive et 0 ou False pour la valeur négative.

Dans un graphe d'implication on définit les fils d'une variable comme toutes les variables qu'on peut attendre par un chemin d'implication y et z sont les fils de x si on le chemin x=>y=>z.

On appelle parents d'une variable toutes les variables dont elle est l'enfant.

Si on part d'une clause du problème 1In3Sat x v y v z =1 on a l' équivalence entre cette classe avec une valuation une variable True et deux False et la clause !x v !y v !z =1 avec une valuation deux True et une variable False.

Dans la suite on présentera une démarche pour résoudre la version 2In3Sat (exactement deux variables True) pour chaque clause.

2) Algo .

Si on a une clause:

x v y v z=1 (1)

avec deux variables sur trois de valuation True alors forcément on a

x v y=1 et x y z= 1 et y v z =1 (2).

Si une valuation est solution d'un système de clauses (1) alors elle est aussi solution du système de clauses (2).l'inverse est faux évidement. Mas la contraposée permet d'affirmer l'invalidé d'un système 2In3Sat . Si (2) est invalide (algo de Tarjan par exemple) alors (1) est invalide.

l'Idée de l'algorithme est qu' en partant d'une instance (1) on va fabriquer un système (2) et d'en déduire d'abord toutes les implications logiques en supposant qu'une valuation qui satisfasse le système existe .

La structure de (2) va imposer certaines assignations aux variables .

Par exemple :

a) Si le système (2) est valide et on a des composantes connexes alors toutes les variables sur une composante connexe sont identiques(même valeur).

b) Si dans une ligne d'implication on a une variable et son opposée alors forcément la première est False et l'autre est True (interdiction de 1==> 0)....

La démarche est la suivante :

1) 2-1)Analyse préalable :

a) Si une clause contient trois fois la même variable (x vx v x =1) alors le système (1) est invalide.

b) Si une clause contient deux variables identiques et de même « signature » (x v x v y =1) alors x= 1 et y=0.

c)) Si une clause contient deux variables identiques et de « signature » opposées (x v !x v y =1) alors y=1 et la clause peut être retirée du système.

d) si on a deux clauses avec deux variables identiques x et y on peut en déduire l'identité ou les valeurs des autres variables :

x v y x z =1 , x v y v r =1 => z=r .

x v y v z=1 , x v !y v r =1=> x=1 et z= !r

x v y v z=1 , !x v !y v r =1 => x = !y et z=r=1.

Une fois cette analyse préalable terminée et qu'on n'obtient pas une contradiction (x= !x) ou une assignation non valide aux variables d'une clause (x=1, y=1 et z=1) on passe à la phase suivante.

2-2) On génère le système 2Sat (2) comme expliqué en 1) et on le traite avec un algo de Tarjan.

a) Si (2) es invalide alors (1) est invalide .Fin.

b) Si (2) est valide on va travailler sur le graphe d'implication.

b-1) Si le graphe contient des cycles alors toutes les variables d'un cycle ont la même valeur et sont identiques.

b-2) Si une variable a deux fils opposés (x et !x) alors elle vaut 0.

b-3) si une variable est le fils et x et de !x alors elle vaut 1.

b-4) tous les fils de 1 sont 1 et tous les parents de 0 sont 0.

Les autres règles similaires sont implémentées dans le poc et relèvent de la logique élémentaire.

On applique plusieurs fois les étapes 1 et 2 jusqu'à la fin de détection de toute opportunité dans

le système et le graphe d'implication.

2-3) On procède à l'assignation aux variables qui restent les valeurs que donne l'algo de Tarjan (en remontant en suivant l'ordre topologique).

2-4) La solution qu'on a va subir un dernier traitement qui permet d'éliminer les valuations 111 des clauses et de ne garder que les valeurs 101, 110 et 011.

Ce traitement est fait dans les fonctions Handle et HandleUpperPart.

L'algo nous donne une valuation pour le système simplifié. Cette valuation et les informations tirés de l'analyse logique (FoundValues et eqalValues1 et eqalValues2) permettent de trouver un valuation solution du système initial.

3) Complexité:

Tous les traitements sont polynomiaux ,le seul point difficile est la fonction HandleUpperPart mais si on voit que l'algo après quelques tentatives attaque les arbres d'implications de la gauche ou de la droite , on se rend compte que l'algo est P.

1In3SAT est P.

4) Lancement du programme:

Il faut choisir une fois l'option N lors du lancement , le programme va générer une instance de 3sat et une instance de 2Sat qu'on va sauvegarder dans le répertoire

c:/Dev/Output.

Une fois l'analyse est faite (phase 1 et 2 de l'algo) on sauvegarde du nouveau les deux fichiers avec les extensions 'final'.

On peut relancer le programme sur un fichier 3Sat existant en choisissant l'option 'Y' au

démarrage.

Le fichier Result.txt contient une solution du système après simplification , une du système initial et les informations sur les variables (égalité).Ces informations permettent d'assigner des valuations aux variables que l'analyse logique a éliminé sans leur donner une valeur (pas présentes dans FoundValues).