Problème SAT

"En informatique théorique, le problème SAT ou problème de satisfaisabilité booléenne est le problème de décision, qui, étant donnée une formule de logique propositionnelle, détermine s'il existe une assignation des variables propositionnelles qui rend la formule vraie. Ce problème est important en théorie de la complexité. Il a été mis en lumière par le théorème de Cook, qui est à la base de la théorie de la NP-complétude et du problème P = NP. Le problème SAT a aussi de nombreuses applications notamment en satisfaction de contraintes, planification classique, model checking, diagnostic, et jusqu'au configurateur d'un PC ou de son système d'exploitation : on se ramène à des formules propositionnelles et on utilise un solveur SAT." (Wikipédia).

Postulat

L'équation de logique propositionnelle est insatisfaite si nous avons deux clauses à variable(s) unique(s) et identique(s), et que ces 2 variables sont opposées ; ou bien si l'ensemble des clauses forment littéralement des combinaisons binaires ordonnées, ou désordonnées.

Dans tous les autres cas l'équation est satisfaite.

Deux clauses unitaires contrariées :

(a) & (!a)

Ou bien

(a|a) & (a|b) & (a|!b|c) & (!a|!a)

Combinaisons binaires littérales ordonnées :

(a|b|c) & (!a|b|c) & (a|!b|c) & (!a|!b|c) & (a|b|!c) & (!a|b|!c) & (a|!b|!c) & (!a|!b|!c)

Ici toutes les combinaisons binaires (a, b, c) sont littéralement présentes dans des clauses respectives, et de façon ordonnée.

Combinaisons binaires littérales désordonnées :

(a|b) & (e|b) & (a|!d) & (!a|c) & (!c| !b) & (!b|d) & (!a| !e)

Ici toutes les combinaisons binaires (a, b) sont présentes, mais tronquées dans des clauses séparées que l'on peut fusionner par la règle de résolution en logique propositionnelle :

(a|b) & (e|b) & (a|!d) & (!a|c) & (!c|!b) & (!b|d) & (!a|!e)

(a|b) & (!a|b) & (a|!b) & (!a|!b)

Ici nous passons de 7 clauses à 4, car 3 sont éliminées par « fusion » avec 3 autres, la première clause « (a|b) » n'ayant pas été fusionnée.

Les combinaisons désordonnées sont la principale difficulté pour un algorithme qui tenterait de résoudre des problèmes SAT par cette méthode, car il s'agit de faire le test avec tous les jeux de variables possibles.

Exemple de jeux de variables possibles avec les combinaisons binaires littérales associées :

(a|!b|c)		(a|b|!c)		(!a|b|c|!d)		(d)	
a	1	a	1	a	0	d	1
b	0	b	1	b	1		
c	1	c	0	c	1		
a/b	10	a/b	11	d	0		
a/c	11	a/c	10	a/b	01		
b/c	01	b/c	10	a/c	01		
a/b/c	101	a/b/c	110	a/d	00		
				b/c	11		
				b/d	10		
				c/d	10		
				a/b/c	011		
				a/b/d	010		
				a/c/d	010		
				b/c/d	110		
				a/b/c/d	0110		

Bleu : Il me manque 00.

Vert : Il me manque 000, 001, 010, 100 et 111.

Vert clair : Il me manque 00.

Gris : Il me manque 00.

Orange : Il me manque toutes les combinaisons à 4 bits sauf une.

Jaune : Combinaisons binaires uniques sans réelles importances.

Rouge : la formule serait insatisfaite si nous avions la clause (!d) quelque part dans l'équation, ce qui n'est pas le cas ici.

Pour chaque clause j'ai *2^length_clause - 1* jeux de variables.

Equation de complexité algorithmique : *Nb_clauses * 2^length_clause * 2*

L’astuce consiste donc à utiliser la règle de résolution logique pour chaque jeu de variables.

Le programme informatique va se décomposer en 3 fonctions principales :

- ✓ Une fonction « sort » qui créé le tableau T décrit à la page précédente. Il s’agit d’un tri des clauses de l’équation d’origine avec lequel on associe à chaque jeu de variables son empreinte binaire basée sur la négativité ou sur la positivité de chaque variable :
  Pour (a|!b) le jeu de variables a/b donne « 10 » en binaire, « 1 » pour « a » qui n’est pas négativé, et « 0 » pour « !b » qui est négativé.
  Fonction qui appelle la fonction « sat » et qui s’arrête là si le retour est faux, dans le cas contraire la fonction appelle « reslog ».

- ✓ Une fonction « sat » capable de déterminer l’insatisfaisabilité d’une équation où 2 clauses unitaires seraient contrariées, ou bien d’une combinaison binaire ordonnée, à partir du tableau T.

- ✓ Une fonction « reslog » capable d’effectuer la règle de résolution en logique propositionnelle pour chaque jeu de variables de l’équation, à partir du tableau T.
  Fonction qui rappelle la fonction « sort » une seule fois, après avoir créé une nouvelle équation par résolution logique.

Programme de test en Javascript ici : https://susoft.fr/sat/sat11.html

Source : https://susoft.fr/sat/sat11.js

Le programme testé ici est comparé à un SAT solveur par bruteforce classique, la source contient les deux méthodes de résolution, les fonctions intéressantes sont nommées ci-dessus.