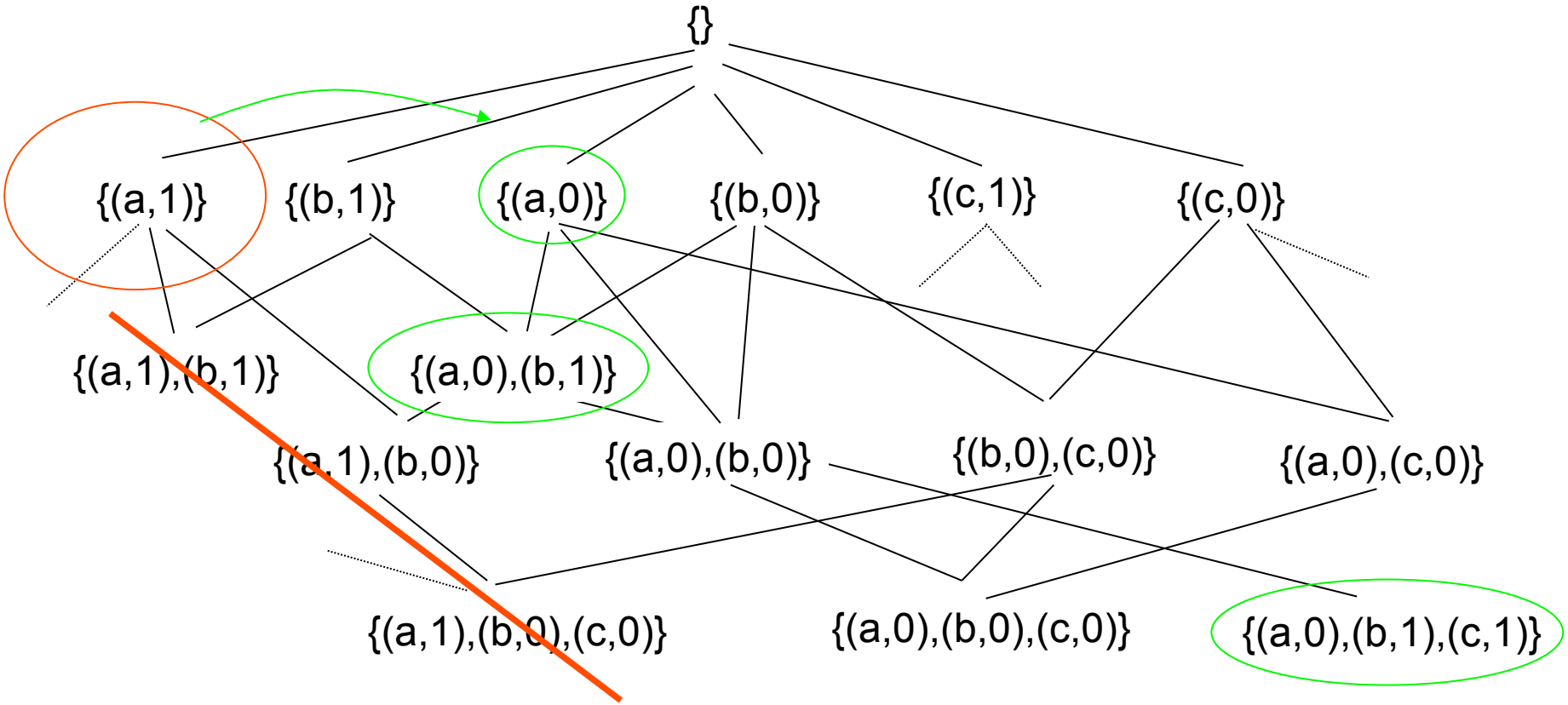
Programme

- Introduction
- Le langage de la LP (syntaxe)
- La sémantique de la LP
- Équivalence logique et Substitution
- Conséquence logique
- Formes normales et clausale
- Méthode de résolution
- Méthode des tableaux
- **Méthode de Davis et Putnam**
- Initiation à la logique des prédicats

Algorithme de Davis-Putnam-Logemann-Loveland

- On s'intéresse à la satisfiabilité d'une fbf sous forme conjonctive : le problème SAT
- Il s'agit d'explorer le plus judicieusement possible l'espace des interprétations partielles à la recherche d'un modèle
  - Exemple : $(a \vee b) \wedge (\neg b \vee c) \wedge (\neg a)$

Espace des interprétations partielles



(a∨b)∧(¬b∨c) ∧(¬a)

Recherche DPLL

- Idée 1 : inutile de produire une interprétation complète si une interprétation partielle est suffisante pour conclure

  => choix d'un symbole propositionnel à valuer

- Idée 2 : tirer parti de tout ce qu'apporte une interprétation partielle

  => propagation

- Idée 3 : procéder par essai erreur

  => backtrack

Algorithme DPLL

Algo : DPLL
Données : F un ens. de clauses, I une interprétation (partielle)
Résultat : *vrai* (satisfiable) ou *faux* (insatisfiable)
Début
    Si toutes les clauses de C sont à 1 alors retourner vrai ;
    (s,v) ← choix d'un symbole s et d'une valeur v;
    conflit ← propagation (s,v);
    Si conflit alors
        I ← backtrack;
        Si I=∅ alors retourner faux;
    Sinon I ← I ∪ {(s,v)};
    DPLL(F, I);
Fin

La propagation

- Règle de la clause unitaire (BCP)
  - Une clause est dite unitaire si l'interprétation partielle courante laisse un de ses littéraux non valué et falsifie tous les autres.
- Règle des littéraux purs
  - Un littéral est dit pur dans l'interprétation partielle courante si toutes ses occurrences dans les clauses non encore satisfaites sont sous la même forme (soit positive, soit négative)
- Dans les 2 cas, la propagation consiste à satisfaire ce littéral

Le choix de la variable

- L'idée serait de maximiser le nombre de clauses satisfaites ou qui deviennent unitaires

  ➔ trop complexe

- Différentes heuristiques sont utilisées
  - Ex : compteur de conflits (on incrémente le cptr dès que la variable apparaît dans une clause insatisfaite) avec l'idée d'essayer de satisfaire en premier les clauses les plus contraintes
  - Aucune ne peut être démontré meilleure qu'une autre

Le backtrack

- Simple backtrack
  - On inverse seulement la valeur du dernier symbole interprété.
  - Garantit la complétude (tout l'espace est exploré)
- Backtrack avec sauts
  - On repart d'un autre point, par exemple en changeant une autre des valeurs précédemment associées à un symbole
    - Ex. WalkSat Algo
  - On inverse plusieurs valeurs à la fois
  - On repart du début sur une autre variable
  - Problème d'incomplétude !!

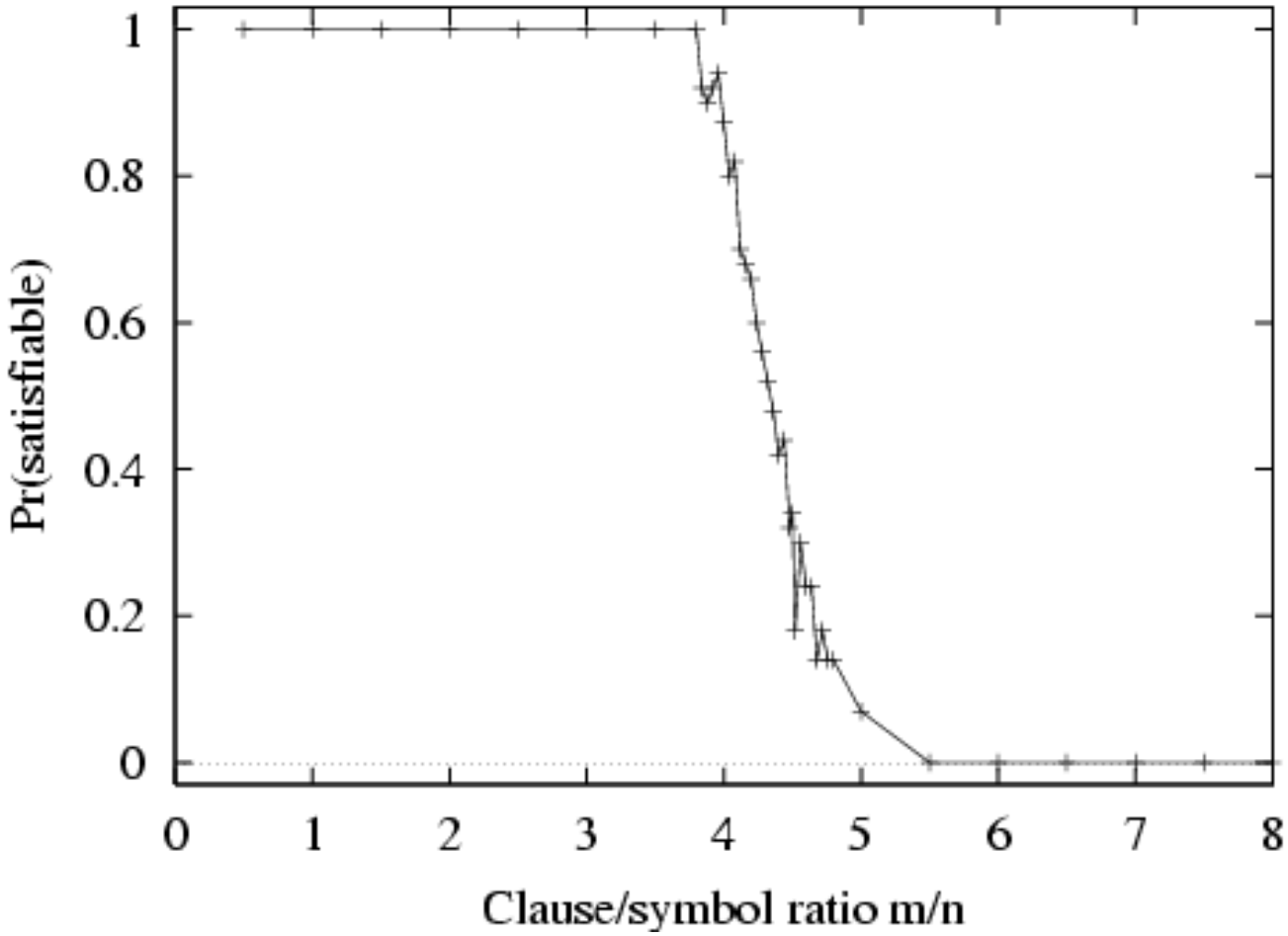
Simple DPLL

Algo : SimpleDPLL
Données : F un ens. de clauses, S un ens. de symboles, I une interprétation (partielle)
Résultat : *vrai* (satisfiable) ou *faux* (insatisfiable)
Début
    Si toutes les clauses de C sont à 1 alors retourner vrai ;
    Si une clause est à 1 retourner faux ;
    (s,v) ← recherche_pur_littéral ;
    Si s existe alors SimpleDPLL(F, S-{s}, I ∪ {(s,v)});
    (s,v) ← recherche_clause_unitaire ;
    Si s existe alors SimpleDPLL(F, S-{s}, I ∪ {(s,v)});
    s ← choisir(S);
    retourner (SimpleDPLL(F, S-{s}, I ∪ {(s,0)})
            ou SimpleDPLL(F, S-{s}, I ∪ {(s,1)}));
Fin

Dureté d'un Sat

- Un Sat “facile” est un problème sur lequel on obtient rapidement une réponse, donc soit :
  - Très satisfiable (beaucoup de modèles)
  - Très insatisfiable (beaucoup de contre-modèles)
- Un Sat “dur” est un problème intermédiaire (nb mod. = nb. contre-modèle) nécessitant de nombreux backtrack
- On peut caractériser ces problèmes en observant le comportement de l'algo pour différents problèmes
  - Soit *m* = nombre de clauses et *n* = nombre de symboles
  - 3-Sat “durs” ont un rapport *m/n* = 4.3

Dureté (n=50)


Pr(satisfiable)
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7
8
Clause/symbol ratio m/n

Temps moyen sur 100 3-SAT satisfiables (n=50)