
A little bit more of SAT and Beyond

Albert Oliveras and Enric Rodríguez-Carbonell

Logic and Algebra in Computer Science

Session 7

Fall 2009, Barcelona



Departament de Llenguatges i Sistemes Informàtics

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Overview of the session

- Look-Ahead SAT Solvers
- Incomplete methods
- Max-SAT
- Pseudo-boolean and Cardinality Constraints
- QBF



Look-Ahead Based SAT Solvers

- Two concepts of CNF formulas:
 - Density \approx cost of unit propagation
 - Diameter \approx global connectivity of clauses
(larger diameter, more local clusters)
- CDCL SAT solvers behave well with:
 - High density
 - Large diameters (i.e. local clusters exist)
- Look-Ahead Based SAT solvers are good with formulas with:
 - Low density
 - Small diameters (i.e. all clauses connected)

Usually those are crafted/handmade problems (puzzles)



Look-Ahead Based SAT Solvers (2)

Overall architecture is a variant of basic DPLL:

DPLL (Formula \mathcal{F})

if $\mathcal{F} = \emptyset$ **then return** SAT

$\langle \mathcal{F}, x_{dec} \rangle := \text{LookAhead}(\mathcal{F})$

if $\square \in \mathcal{F}$ **then return** UNSAT

else if no x_{dec} is selected **then return** DPLL(\mathcal{F})

$b := \text{DirectionHeuristic}(x_{dec}, \mathcal{F})$

if DPLL($\mathcal{F}[x_{dec} = b]$) = SAT **then return** SAT

else return DPLL($\mathcal{F}[x_{dec} = \neg b]$)



Look-Ahead Based SAT Solvers (3)

```
LookAhead ( Formula  $\mathcal{F}$  )  
  
   $\mathcal{P} := \text{Preselect} (\mathcal{F})$  // select some vars  
  
  repeat  
    for all vars  $x \in \mathcal{P}$  do  
       $\mathcal{F} := \text{LookAheadReasoning} (\mathcal{F}, x)$   
      if  $\square \in \mathcal{F}[x = 0]$  and  $\square \in \mathcal{F}[x = 1]$  then  
        return  $\langle \mathcal{F}[x = 0]; * \rangle$   
      else if  $\square \in \mathcal{F}[x = 0]$  then  $\mathcal{F} := \mathcal{F}[x = 1]$   
      else if  $\square \in \mathcal{F}[x = 1]$  then  $\mathcal{F} := \mathcal{F}[x = 0]$   
      else  $H(x) := \text{DecHeuristic}(\mathcal{F}, \mathcal{F}[x = 0], \mathcal{F}[x = 1])$   
    end for  
  until no progress is made  
  
  return  $\langle \mathcal{F}; x \text{ with greatest } H(x) \rangle$ 
```



Overview of the session

- Look-Ahead SAT Solvers
- **Incomplete methods**
- Max-SAT
- Pseudo-boolean and Cardinality Constraints
- QBF



Incomplete methods

- Do not guarantee an answer
- Never classify formula as UNSAT, but may report models
- It is claimed they outperform CDCL on some applications
- There are also hybrid methods



Incomplete methods - GSAT

INPUT: CNF \mathcal{F}

PARAMETERS: integers MAX-FLIPS, MAX-TRIES

OUTPUT: a model for \mathcal{F} or FAIL

for $i=1$ **to** MAX-TRIES **do**

$\sigma :=$ a randomly generated assignment for \mathcal{F}

for $j=1$ **to** MAX-FLIPS **do**

if $\sigma \models F$ **then return** σ

$v :=$ a variable flipping resulting in the greatest
 decrease in the # of unsatisfied clauses

 Flip v in σ

return FAIL



Incomplete methods - Walksat

INPUT: CNF \mathcal{F}

PARAMETERS: MAX-FLIPS, MAX-TRIES and $p \in [0, 1]$

OUTPUT: a model for \mathcal{F} or FAIL

for $i=1$ **to** MAX-TRIES **do**

$\sigma :=$ a randomly generated assignment for \mathcal{F}

for $j=1$ **to** MAX-FLIPS **do**

if $\sigma \models F$ **then return** σ

$C :=$ unsatisfied clause chosen at random

if $\exists x \in C$ with break-count=0 **then** $v := x$

else

With prob. p

$v :=$ var in C chosen at random

With prob. $1 - p$

$v :=$ var in C with smallest break-count

Flip v in σ

return FAIL



Overview of the session

- Look-Ahead SAT Solvers
- Incomplete methods
- **Max-SAT**
- Pseudo-boolean and Cardinality Constraints
- QBF



Max-SAT

- **Max-SAT:**
Given a CNF, try to satisfy the maximum number of clauses
- **Weighted Max-SAT:**
Given a CNF where each clause has a cost, find an assignment that minimizes the sum of the cost of the falsified clauses
- **Partial Max-SAT:**
Given a CNF with *hard* and *soft* clauses, find an assignment that satisfies all hard clauses and the maximum number of soft clauses

In the following slides, we will consider Max-SAT



Max-Sat – Branch and Bound

MaxSAT (Formula \mathcal{F})

$\mathcal{F} := \text{simplifyFormula}(\mathcal{F})$

if $\mathcal{F} = \emptyset$ **or** \mathcal{F} only contains empty clauses **then**
 return # empty clauses

LB := #empty clauses + lowerBounding(\mathcal{F})

if $LB \geq UB$ **then return** UB

$x := \text{selectVar}(\mathcal{F})$

UB := min(UB, MaxSAT($\mathcal{F}[x = 0]$, UB))

return min(UB, MaxSAT($\mathcal{F}[x = 1]$, UB))



Max-Sat – Branch and Bound (2)

- Lower bounding can be computed using inference rules
- Example:

$$\left\{ \begin{array}{c} l_1 \\ \neg l_1 \vee \neg l_2 \\ l_2 \end{array} \right\} \Longrightarrow \left\{ \begin{array}{c} \square \\ l_1 \vee l_2 \end{array} \right\}$$

- Most of this inferences can be seen as particular instances of the complete Max-Resolution rule:

$$\left\{ \begin{array}{c} x \vee A \\ \neg x \vee B \end{array} \right\} \Longrightarrow \left\{ \begin{array}{c} A \vee B \\ x \vee A \vee \neg B \\ \neg x \vee \neg A \vee B \end{array} \right\}$$

Max-Sat – Core-base refinement

- Another approach is an abstraction-refinement one:
 - Consider all clauses as hard
 - If problem UNSAT, obtain core *Core* [abstract]
 - Convert each clause $C \in \text{Core}$ into $C \vee b$
 - Add constraint of the form $\sum b \leq 1$ [refine]
 - Iterate until problem is SAT (solution is #b's assigned to 1)
- This is a very naive presentation of the algorithm
- More sophisticated versions exist



Overview of the session

- Look-Ahead SAT Solvers
- Incomplete methods
- Max-SAT
- Pseudo-boolean and Cardinality Constraints
- QBF



Pseudo-Booleans and Cardinality Constraints

- SAT problem can be extended with constraints of the form:
 - $2x - 3y + 4z \leq 5$, where x, y, z are Boolean vars
(Pseudo-Boolean constraint)
 - $atmost(k, \{x_1, \dots, x_n\})$
 $atleast(k, \{x_1, \dots, x_n\})$
 $exactly(k, \{x_1, \dots, x_n\})$, where $k \in \mathbb{Z}$ and x_i are Boolean vars
(Cardinality constraints)
- There are two ways of dealing with them:
 - Encode them into SAT
 - Develop specialized tools



Overview of the session

- Look-Ahead SAT Solvers
- Incomplete methods
- Max-SAT
- Pseudo-boolean and Cardinality Constraints
- **QBF**



- QBF checks satisfiability of Quantified Boolean Formulas:

$$\forall x(\neg x \vee \exists z \exists u ((z \wedge \neg x) \vee (u \wedge z)))$$

- QBF is the prototypical PSPACE-complete problem
- Applications in planning, knowledge representation, formal methods,...
- However, QBF solvers are not as widely used as SAT solvers
- Again, two directions of research:
 - Reduction to SAT
 - Development of dedicated methods

Bibliography - Some further reading

- Armin Biere, Marijn Heule, Hans van Maaren, Toby Walsh (Eds.). *Handbook of Satisfiability*. Frontiers in Artificial Intelligence and Applications 185 IOS Press 2009, ISBN 978-1-58603-929-5

