GRASP: A Search Algorithm for Propositional Satisfiability

EE878C Homework #2
2002/11/1
KAIST, EECS ICS Lab
Lee, Dongsoo
Sat in a Nutshell

- Given a Boolean formula, find a variable assignment such that the formula evaluates to 1, or prove that no such assignment exists.

\[ F = (a + b)(a' + b' + c) \]

- For \( n \) variables, there are \( 2^n \) possible truth assignments to be checked.

- NP-Complete problem.
**Problem Representation**

- **Conjunctive Normal Form**
  - \( F = (a+b)(a'+b'+c) \)
  - Simple representation (more efficient data structures)

- **Logic circuit representation**
  - Circuits have structural and direction information

- **Circuit – CNF conversion is straightforward**

\[
\begin{align*}
  d &\equiv (a + b) \\
  (a + b + d') \\
  (a' + d) \\
  (b' + d) \\
\end{align*}
\]

\[
\begin{align*}
  e &\equiv (c \cdot d) \\
  (c' + d' + e) \\
  (d + e') \\
  (c + e') \\
\end{align*}
\]
**DLL Algorithm**

- **Davis, Logemann and Loveland**

- Basic framework for many modern SAT solvers
- Also known as DPLL for historical reasons
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a' + c' + d)
(b' + c' + d)
(a' + b + c')
(a' + b' + c)
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)
Basic DLL Procedure - DFS

\[
\begin{align*}
(a' + b + c) \\
(a + c + d) \\
(a + c + d') \\
(a + c' + d) \\
(a + c' + d') \\
(b' + c' + d) \\
(a' + b + c') \\
(a' + b' + c)
\end{align*}
\]

\[\text{Decision} \iff a\]
Basic DLL Procedure - DFS

\[
\begin{align*}
(a' + b + c) \\
(a + c + d) \\
(a + c + d') \\
(a + c' + d) \\
(a + c' + d') \\
(b' + c' + d) \\
(a' + b + c') \\
(a' + b' + c)
\end{align*}
\]
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a' + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)

Implication Graph

Conflict!
Basic DLL Procedure - DFS

Implication Graph

Conflict!
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)

 Backtrack
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)

\[0 \quad 1 \quad \text{Forced Decision}\]

Conflict!
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(b' + c' + d)
(a' + b + c')
(a' + b' + c)
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c + d)
(a' + b + c')
(a' + b' + c)

Decision

Conflict!
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c' + d')
(a + c' + d)
(a' + b + c')
(b' + c' + d)
(a' + b' + c)

Diagram:

```
  a
   /
  b
 /  \
0  1
  /
 c c
0 1 0
```

← Backtrack
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)

 Forced Decision

Conflict!
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)

⇔ Forced Decision
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c' + d')
(a + c' + d)
(b' + c' + d)
(a' + b + c')
(a' + b' + c)
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)

Conflict!
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)

Backtrack
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a=1</th>
<th>(a' + b' + c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b=1</td>
<td>c=1</td>
</tr>
</tbody>
</table>

< Forced Decision>
Basic DLL Procedure - DFS

(a' + b + c)
(a + c + d)
(a + c + d')
(a + c' + d)
(a + c' + d')
(b' + c' + d)
(a' + b + c')
(a' + b' + c)

Diagram:

- a
  - b
    - c
    - c
      - d
      - d
    - d
  - b
    - 1
      - 0
      - 1
    - 0
      - 1
    - 1
  - c
    - 0
      - 1
      - 1
    - 0
      - 1
    - 1

Values:
- a=1
- b=1
- c=1
- d=1
Basic DLL Procedure - DFS

\[(a' + b + c)\]
\[(a + c + d)\]
\[(a + c + d')\]
\[(a + c' + d)\]
\[(b' + c + d)\]
\[(a' + b + c')\]
\[(a' + b' + c)\]
Implications and Boolean Constraint Propagation

- **Implication**
  - A variable is forced to be assigned to be True or False based on previous assignments.

- **Unit clause rule (rule for elimination of one literal clauses)**
  - An unsatisfied clause is a **unit clause** if it has exactly one unassigned literal.
    - The unassigned literal is implied because of the unit clause.

- **Boolean Constraint Propagation (BCP)**
  - Iteratively apply the unit clause rule until there is no unit clause available
  - Workhorse of DLL based algorithms.
GRASP

- Marques-Silva and Sakallah [SS96, SS99]
- Incorporates conflict driven learning and non-chronological backtracking
- Practical SAT instances can be solved in reasonable time
- Bayardo and Schrag’s ReISAT also proposed conflict driven learning [BS97]
Conflict Driven Learning and Non-chronological Backtracking

\[ x_1 + x_4 \]
\[ x_1 + x_3' + x_8' \]
\[ x_1 + x_8 + x_{12} \]
\[ x_2 + x_{11} \]
\[ x_7' + x_3' + x_9 \]
\[ x_7' + x_8 + x_9' \]
\[ x_7 + x_8 + x_{10'} \]
\[ x_7 + x_{10} + x_{12'} \]
Conflict Driven Learning and Non-chronological Backtracking

\[ x_1 + x_4 \]
\[ x_1 + x_3' + x_8' \]
\[ x_1 + x_8 + x_{12} \]
\[ x_2 + x_{11} \]
\[ x_7' + x_3' + x_9 \]
\[ x_7' + x_8 + x_9' \]
\[ x_7 + x_8 + x_{10'} \]
\[ x_7 + x_{10} + x_{12'} \]

\[ x_1 = 0 \]
Conflict Driven Learning and Non-chronological Backtracking

\[ x_1 + x_4 \]
\[ x_1 + x_3' + x_8' \]
\[ x_1 + x_8 + x_12 \]
\[ x_2 + x_{11} \]
\[ x_7' + x_3' + x_9 \]
\[ x_7' + x_8 + x_9' \]
\[ x_7 + x_8 + x_{10'} \]
\[ x_7 + x_{10} + x_{12'} \]

\[ x_1 = 0, \ x_4 = 1 \]
Conflict Driven Learning and Non-chronological Backtracking

\[ x_1 + x_4 \]
\[ x_1 + x_3' + x_8' \]
\[ x_1 + x_8 + x_{12} \]
\[ x_2 + x_{11} \]
\[ x_7' + x_3' + x_9 \]
\[ x_7' + x_8 + x_9' \]
\[ x_7 + x_8 + x_{10'} \]
\[ x_7 + x_{10} + x_{12'} \]

\[ x_4 = 1 \]
\[ x_1 = 0 \]
\[ x_3 = 1 \]

\[ x_1 = 0, x_4 = 1 \]
\[ x_3 = 1 \]
Conflict Driven Learning and Non-chronological Backtracking

\[ x_1 + x_4 \]
\[ x_1 + x_3' + x_8' \]
\[ x_1 + x_8 + x_12 \]
\[ x_2 + x_11 \]
\[ x_7' + x_3' + x_9 \]
\[ x_7' + x_8 + x_9' \]
\[ x_7 + x_8 + x_10' \]
\[ x_7 + x_10 + x_12' \]
Conflict Driven Learning and Non-chronological Backtracking

\begin{align*}
x_1 + x_4 \\
x_1 + x_3' + x_8' \\
x_1 + x_8 + x_{12} \\
x_2 + x_{11} \\
x_{7'} + x_3' + x_9 \\
x_{7'} + x_8 + x_9' \\
x_7 + x_8 + x_{10'} \\
x_7 + x_{10} + x_{12'}
\end{align*}
Conflict Driven Learning and Non-chronological Backtracking

\[
\begin{align*}
x_1 + x_4 \\
x_1 + x_3' + x_8' \\
x_1 + x_8 + x_{12} \\
x_2 + x_{11} \\
x_7' + x_3' + x_9 \\
x_7' + x_8 + x_9' \\
x_7 + x_8 + x_{10}' \\
x_7 + x_{10} + x_{12}'
\end{align*}
\]

\[
\begin{align*}
x_1 &= 0, \ x_4 &= 1 \\
x_3 &= 1, \ x_8 &= 0, \ x_{12} &= 1 \\
x_2 &= 0
\end{align*}
\]
Conflict Driven Learning and Non-chronological Backtracking

\[ x_1 + x_4 \]
\[ x_1 + x_3' + x_8' \]
\[ x_1 + x_8 + x_{12} \]
\[ x_2 + x_{11} \]
\[ x_{7'} + x_3' + x_9 \]
\[ x_{7'} + x_8 + x_9' \]
\[ x_7 + x_8 + x_{10'} \]
\[ x_7 + x_{10} + x_{12'} \]
Conflict Driven Learning and Non-chronological Backtracking

\[ x_1 + x_4 \]
\[ x_1 + x_3' + x_8' \]
\[ x_1 + x_8 + x_{12} \]
\[ x_2 + x_{11} \]
\[ x_7' + x_3' + x_9 \]
\[ x_7' + x_8 + x_9' \]
\[ x_7 + x_8 + x_{10}' \]
\[ x_7 + x_{10} + x_{12}' \]

\[ x_4 = 1 \]
\[ x_1 = 0 \]
\[ x_3 = 1 \]
\[ x_7 = 1 \]
\[ x_{11} = 1 \]
\[ x_2 = 0 \]
\[ x_8 = 0 \]
\[ x_{12} = 1 \]

\[ x_1 = 0, x_4 = 1 \]
\[ x_3 = 1, x_8 = 0, x_{12} = 1 \]
\[ x_2 = 0, x_{11} = 1 \]
\[ x_7 = 1 \]
Conflict Driven Learning and Non-chronological Backtracking

\[ x_1 + x_4 \]
\[ x_1 + x_3' + x_8' \]
\[ x_1 + x_8 + x_{12} \]
\[ x_2 + x_{11} \]
\[ x_7' + x_3' + x_9 \]
\[ x_7' + x_8 + x_9' \]
\[ x_7 + x_8 + x_{10}' \]
\[ x_7 + x_{10} + x_{12}' \]

Diagram:
- \( x_1 = 0, x_4 = 1 \)
- \( x_3 = 1, x_8 = 0, x_{12} = 1 \)
- \( x_2 = 0, x_{11} = 1 \)
- \( x_7 = 1, x_9 = 0, 1 \)
Conflict Driven Learning and Non-chronological Backtracking

\[ \begin{align*}
x_1 + x_4 \\
x_1 + x_3' + x_8' \\
x_1 + x_8 + x_{12} \\
x_2 + x_{11} \\
x_7' + x_3' + x_9 \\
x_7' + x_8 + x_9' \\
x_7 + x_8 + x_{10'} \\
x_7 + x_{10} + x_{12'} \\
\end{align*} \]

\[ \begin{align*}
x_1 &= 0, \quad x_4 = 1 \\
x_3 &= 1, \quad x_8 = 0, \quad x_{12} = 1 \\
x_2 &= 0, \quad x_{11} = 1 \\
x_7 &= 1, \quad x_9 = 1 \\
x_3 = 1 \land x_7 = 1 \land x_8 = 0 \rightarrow \text{conflict} \\
\end{align*} \]
Conflict Driven Learning and Non-chronological Backtracking

\[ x_1 + x_4 \]
\[ x_1 + x_3' + x_8' \]
\[ x_1 + x_8 + x_{12} \]
\[ x_2 + x_{11} \]
\[ x_7' + x_3' + x_9 \]
\[ x_7' + x_8 + x_9' \]
\[ x_7 + x_8 + x_{10}' \]
\[ x_7 + x_{10} + x_{12}' \]

x1=0, x4=1
x3=1, x8=0, x12=1
x2=0, x11=1
x7=1, x9=1

x3=1 ∧ x7=1 ∧ x8=0 → conflict
Add conflict clause: x3'+x7'+x8
Conflict Driven Learning and Non-chronological Backtracking

\[ x_1 + x_4 \]
\[ x_1 + x_3' + x_8' \]
\[ x_1 + x_8 + x_{12} \]
\[ x_2 + x_{11} \]
\[ x_7' + x_3' + x_9 \]
\[ x_7' + x_8 + x_9' \]
\[ x_7 + x_8 + x_{10'} \]
\[ x_7 + x_{10} + x_{12'} \]

\[ x_3' + x_7' + x_8 \]

\[ x_1 \]
\[ x_3 \]
\[ x_2 \]
\[ x_7 \]

\[ x_1 = 0, x_4 = 1 \]
\[ x_1 = 0, x_3 = 1, x_8 = 0, x_{12} = 1 \]
\[ x_2 = 0, x_{11} = 1 \]
\[ x_7 = 1, x_9 = 1 \]

\[ x_3 = 1 \land x_7 = 1 \land x_8 = 0 \rightarrow \text{conflict} \]

Add conflict clause: \[ x_3' + x_7' + x_8 \]
Conflict Driven Learning and Non-chronological Backtracking

\[
\begin{align*}
&x1 + x4 \\
&x1 + x3' + x8' \\
&x1 + x8 + x12 \\
&x2 + x11 \\
&x7' + x3' + x9 \\
&x7' + x8 + x9' \\
&x7 + x8 + x10' \\
&x7 + x10 + x12' \\
&x3' + x8 + x7' \\
\end{align*}
\]

Backtrack to the decision level of \(x3=1\)  
\[x7 = 0\]
Conflict Clause

- Significantly prune the search space
  - Learned clause is useful forever
- Useful in generating future conflict clauses