Satisfiability Checking

SAT-Solving

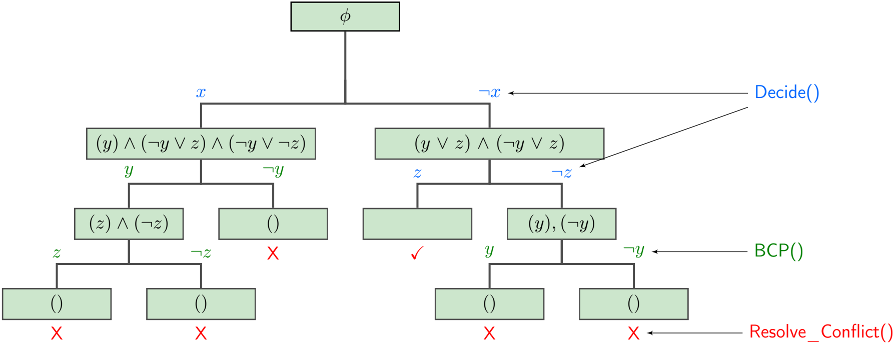
Prof. Dr. Erika Ábrahám

Theory of Hybrid Systems
Informatik 2

WS 10/11

Assume the CNF formula

$\phi \ : \ (x \lor y \lor z) \land (\neg x \lor y) \land (\neg y \lor z) \land (\neg x \lor \neg y \lor \neg z)$



A basic SAT algorithm

```
while (true)
 {
     if (!decide()) return SAT;
     while (!BCP())
          if (!resolve_conflict()) return UNSAT;
 }
```

Choose the next variable and value.
Return false if all variables are assigned.

Boolean Constraint Propagation. Return false if reached a conflict.

Conflict resolution and backtracking. Return false if impossible.

- Decision
- Boolean Constraint Propagation (BCP)
- Conflict resolution and backtracking

- Decision
- Boolean Constraint Propagation (BCP)
- Conflict resolution and backtracking

DLIS (Dynamic Largest Individual Sum) – choose the assignment that increases the most the number of satisfied clauses

- For a given variable x:
  - C_{xp} – # unresolved clauses in which x appears positively
  - C_{xn} - # unresolved clauses in which x appears negatively
  - Let x be the literal for which C_{xp} is maximal
  - Let y be the literal for which C_{yn} is maximal
  - If $C_{xp} > C_{yn}$ choose x and assign it TRUE
  - Otherwise choose y and assign it FALSE
- Requires $\mathcal{O}(\#\textit{literals})$ queries for each decision.

Jersolow-Wang method

Compute for every clause c and every literal l:

$$J(l) : \sum_{l \in c, c \in \phi} 2^{-|c|}$$

- Choose a literal l that maximizes $J(l)$.
- This gives an exponentially higher weight to literals in shorter clauses

- We will see other (more advanced) decision heuristics soon.
- These heuristics are integrated with a mechanism called learning of conflict clauses, which we will learn soon.

- Decision
- Boolean Constraint Propagation (BCP)
- Conflict resolution and backtracking

- A clause can be

satisfied:	at least one literal is satisfied
unsatisfied:	all literals are assigned but none are statisfied
unit:	all but one literals are assigned but none are satisfied
unresolved:	all other cases

- Example: $c = (x_1 \vee x_2 \vee x_3)$

x_1	x_2	x_3	c
1	0		satisfied
0	0	0	unsatisfied
0	0		unit
	0		unresolved

BCP: Unit clauses are used to imply consequences of decisions.

- Organize the search in the form of an implication graph
  - Each node corresponds to a variable assignment
  - Definition: Decision Level (DL) is the depth of the node in the decision tree.
  - Notation: $x = v@d$
    x is assigned to $v \in \{0, 1\}$ at the decision level d

Definition

An implication graph is a labeled directed acyclic graph $G(V, E)$, where

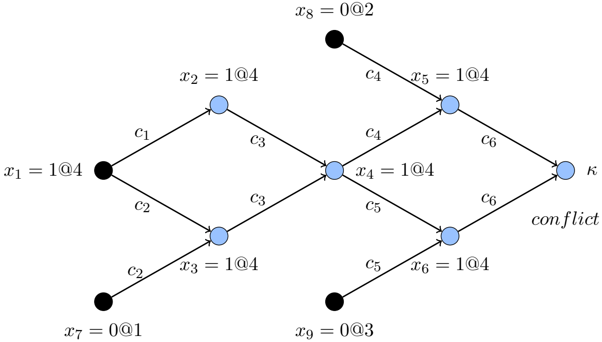
- V represents the literals of the current partial assignment.
  Each node is labeled with the literal that it represents and the decision level at which it entered the partial assignment.
- E with $E = \{(v_i, v_j) | v_i, v_j \in V, v_i \neq v_j, \neg v_i \in \text{Antecedent}(v_j)\}$ denotes the set of directed edges where each edge (v_i, v_j) is labeled with Antecedent(v_j).
- G can also contain a single conflict node labeled with κ and incoming edges $\{(v, \kappa) | \neg v \in c\}$ labeled with c for some conflicting clause c.

Current truth assignment:

$$\{x_7 = 0@1,\ x_8 = 0@2,\ x_9 = 0@3\}$$

Current decision assignment: $\{x_1 = 1@4\}$

$c_1 = (\neg x_1 \vee x_2)$
$c_2 = (\neg x_1 \vee x_3 \vee x_7)$
$c_3 = (\neg x_2 \vee \neg x_3 \vee x_4)$
$c_4 = (\neg x_4 \vee x_5 \vee x_8)$
$c_5 = (\neg x_4 \vee x_6 \vee x_9)$
$c_6 = (\neg x_5 \vee \neg x_6)$

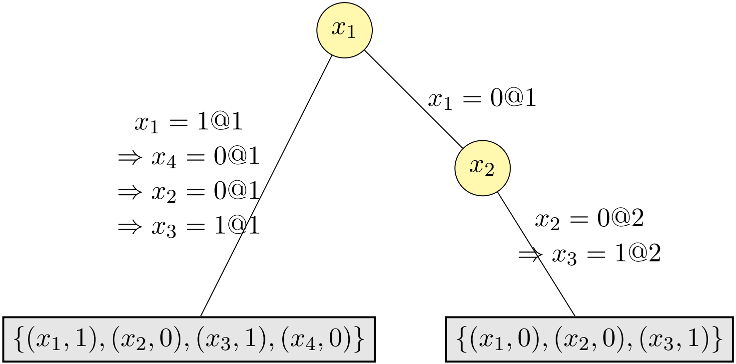


- For BCP, it would be a large effort to check for each propagation the value of each literal in each clause.
- One could keep for each literal a list of clauses in which it occurs.
- It is even enough to watch two literals in each clause such that either one of them is true or both are unassigned.
  If a literal l gets true, we check each clause in which $\neg l$ is a watch literal (which is now false).
    - If the other watch is true, the clause is satisfied.
    - Else, if we find a new watch position, we are done.
    - Else, if the other watch is unassigned, the clause is unit.
    - Else, if the other watch is false, the clause is conflicting.

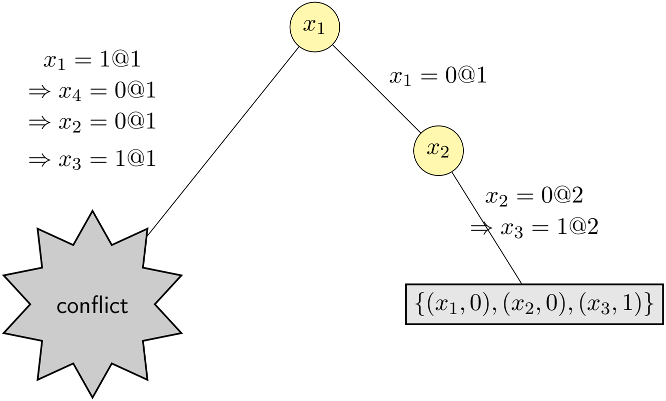
- Decision
- Boolean Constraint Propagation (BCP)
- Conflict resolution and backtracking
    - Basic backtracking
    - Non-chronological backtracking
    - Conflict-driven backtracking

$$c_1 = (x_2 \vee x_3)$$
$$c_2 = (\neg x_1 \vee \neg x_4)$$
$$c_3 = (\neg x_2 \vee x_4)$$

No backtrack in this example!



$$c_1 = (x_2 \vee x_3)$$
$$c_2 = (\neg x_1 \vee \neg x_4)$$
$$c_3 = (\neg x_2 \vee x_4)$$
$$c_4 = (\neg x_1 \vee x_2 \vee \neg x_3)$$



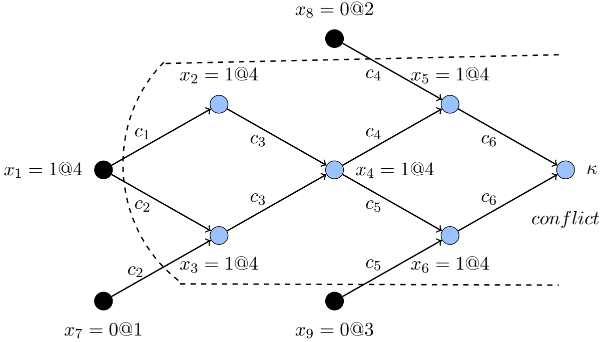
- Decision
- Boolean Constraint Propagation (BCP)
- Conflict resolution and backtracking
    - Basic backtracking
    - Non-chronological backtracking
    - Conflict-driven backtracking

Current truth assignment:

$$\{x_7 = 0@1, x_8 = 0@2, x_9 = 0@3\}$$

Current decision assignment: $\{x_1 = 1@4\}$

$c_1 = (\neg x_1 \vee x_2)$
$c_2 = (\neg x_1 \vee x_3 \vee x_7)$
$c_3 = (\neg x_2 \vee \neg x_3 \vee x_4)$
$c_4 = (\neg x_4 \vee x_5 \vee x_8)$
$c_5 = (\neg x_4 \vee x_6 \vee x_9)$
$c_6 = (\neg x_5 \vee \neg x_6)$



We learn the conflict clause $c_7 : (\neg x_1 \vee x_7 \vee x_8 \vee x_9)$

What to do now?

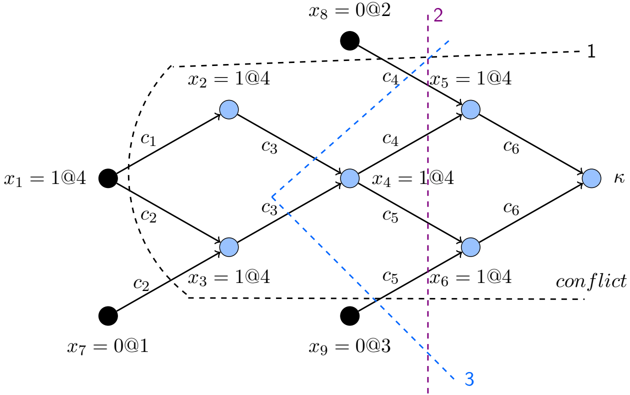
- Undo decision level 4.
- Propagate in the new clause c_7 at decision level 3.
- It leads to a new assignment at decision level 3.
- Propagate the newly assigned literals.

So the rule is:

- Backtrack to the largest decision level in the conflict clause,
- propagate in the learned clause, and
- propagate all new assignments.

- Decision
- Boolean Constraint Propagation (BCP)
- Conflict resolution and backtracking
  - Basic backtracking
  - Non-chronological backtracking
  - Conflict-driven backtracking

- Def: A conflict clause is any clause implied by the formula.
- Let L be a set of literals labeling nodes that form a cut in the implication graph, seperating the conflict node from the roots.
- Claim: $\vee_{l \in L} \neg l$ is a conflict clause.



1. $(x_8 \vee \neg x_1 \vee x_7 \vee x_9)$
2. $(x_8 \vee \neg x_4 \vee x_9)$
3. $(x_8 \vee \neg x_2 \vee \neg x_3 \vee x_9)$

⋮

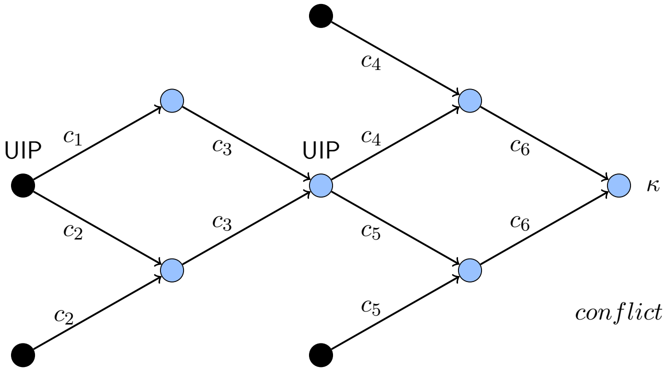
⋮

- How many clauses should we add?
- If not all, then which ones?
  - Shorter ones?
  - Check their influence on the backtracking level?
  - The most "influential"?

- Def: An asserting clause is a conflict clause with a single literal from the current decision level.

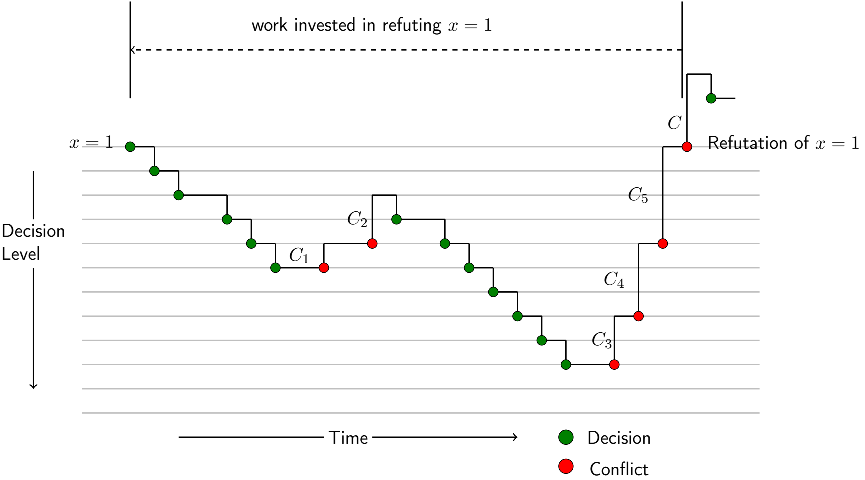
  Backtracking (to the right level) makes it a unit clause.
- Asserting clauses are those that force an immediate change in the search path.
- Modern solvers only consider asserting clauses.

- Definition: A Unique Implication Point (UIP) is an internal node in the implication graph such that all paths from the last decision to the conflict node go through it.
- The first UIP is the UIP closest to the conflict.



- So the rule is: backtrack to the second highest decision level *dl*, but do not erase it.
- This way the literal with the currently highest decision level will be implied at decision level *dl*.
- Question: What if the conflict clause has a single literal?
  - For example, from $(x \vee \neg y) \wedge (x \vee y)$ and decision $x = 0$, we learn the conflict clause (x).

Progress of a SAT solver



- The binary resolution is a sound (and complete) inference rule:

$$\frac{(\beta \vee a_1 \vee \ldots \vee a_n) \qquad (\neg\beta \vee b_1 \vee \ldots \vee b_m)}{(a_1 \vee \ldots \vee a_n \vee b_1 \vee \ldots \vee b_m)} \text{(Binary Resolution)}$$

- Example:

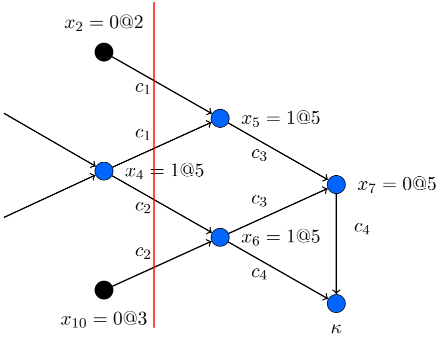
$$\frac{(x_1 \vee x_2) \qquad (\neg x_1 \vee x_3 \vee x_4)}{(x_2 \vee x_3 \vee x_4)}$$

What is the relation of resolution and conflict clauses?

Conflict clauses and resolution

- Consider the following example:

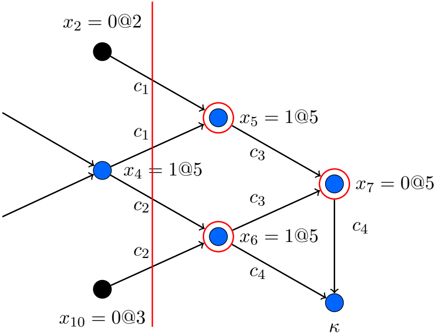
$c_1 = (\neg x_4 \vee x_2 \vee x_5)$
$c_2 = (\neg x_4 \vee x_{10} \vee x_6)$
$c_3 = (\neg x_5 \vee \neg x_6 \vee \neg x_7)$
$c_4 = (\neg x_6 \vee x_7)$
$\vdots \qquad \vdots$



- Conflict clause: $c_5 : (x_2 \vee \neg x_4 \vee x_{10})$

- Conflict clause: $c_5 : (x_2 \vee \neg x_4 \vee x_{10})$

$c_1 = (\neg x_4 \vee x_2 \vee x_5)$
$c_2 = (\neg x_4 \vee x_{10} \vee x_6)$
$c_3 = (\neg x_5 \vee \neg x_6 \vee \neg x_7)$
$c_4 = (\neg x_6 \vee x_7)$
$\vdots \qquad \vdots$



- Assigment order: x_4, x_5, x_6, x_7
  - T1 = Res$(c_4, c_3, x_7) = (\neg x_5 \vee \neg x_6)$
  - T2 = Res(T1,$c_2, x_6) = (\neg x_4 \vee \neg x_5 \vee x_{10})$
  - T3 = Res(T2,$c_1, x_5) = (x_2 \vee \neg x_4 \vee x_{10})$

```
procedure analyze_conflict() {
  if (current_decision_level = 0) return false;
  cl := current_conflicting_clause;
  while (not stop_criterion_met(cl)) do {
    lit   := last_assigned_literal(cl);
    var   := variable_of_literal(lit);
    ante  := antecedent(var);
    cl    := resolve(cl, ante, var);
  }
  add_clause_to_database(cl);
  return true;
}
```

Applied to our example:

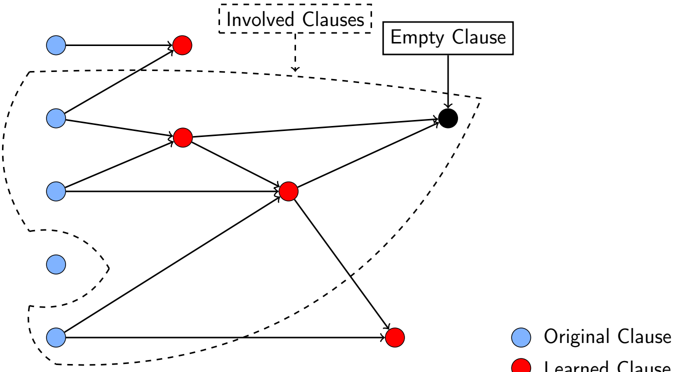
name	*cl*	*lit*	*var*	*ante*
c_4	$(\neg x_6 \vee x_7)$	x_7	x_7	c_3
	$(\neg x_5 \vee \neg x_6)$	$\neg x_6$	x_6	c_2
	$(\neg x_4 \vee x_{10} \vee \neg x_5)$	$\neg x_5$	x_5	c_1
c_5	$(\neg x_4 \vee x_2 \vee x_{10})$			

Definition

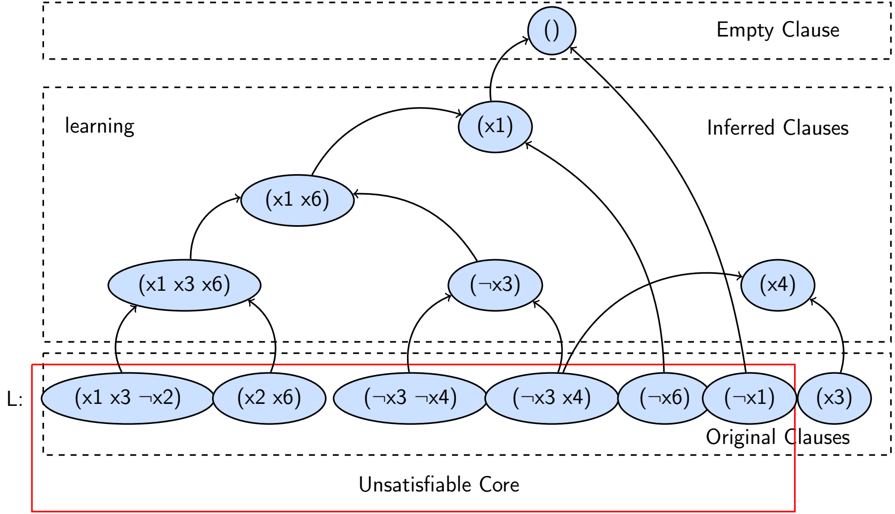
An unsatisfiable core of an unsatisfiable CNF formula is an unsatisfiable subset of the original set of clauses.

- The set of all original clauses is an unsatisfiable core.
- The set of those original clauses that were used for resolution in conflict analysis during SAT-solving (inclusively the last conflict at decision level 0) gives us an unsatisfiable core which is in general much smaller.
- However, this unsatifiable core is still not always minimal (i.e., we can remove clauses from it still having an unsatisfiable core).

A resolution graph gives us more information to get a minimal unsatisfiable core.



Resolution graph: Example



Theorem

It is never the case that the solver enters decision level dl again with the same partial assignment.

Proof.

Define a partial order on partial assignments: $\alpha < \beta$ iff either α is an extension of β or α has more assignments at the smallest decision level at that α and β do not agree.
BCP decreases the order, conflict-driven backtracking also. Since the order always decreases during the search, the theorem holds. □

Back to decision heuristics...

- Decision
- Boolean Constraint Propagation (BCP)
- Conflict resolution and backtracking
  - Basic backtracking
  - Non-chronological backtracking
  - Conflict-driven backtracking

- VSIDS(Variable State Independent Decaying Sum)
- Gives priority to variables involved in recent conflicts.
- "Involved" can have different definitions. We take those variables that occur in clauses used for conflict resolution.

1. Each variable in each polarity has a counter initialized to 0.
2. We define an increment value (e.g., 1).
3. When a conflict occurs, we increase the counter of each variable, that occurs in at least one clause used for conflict resolution, by the increment value.
   Afterwards we increase the increment value (e.g., by 1).
4. For decisions, the unassigned variable with the highest counter is chosen.
5. Periodically, all the counters and the increment value are divided by a constant.

Decision heuristics - VSIDS (cont'd)

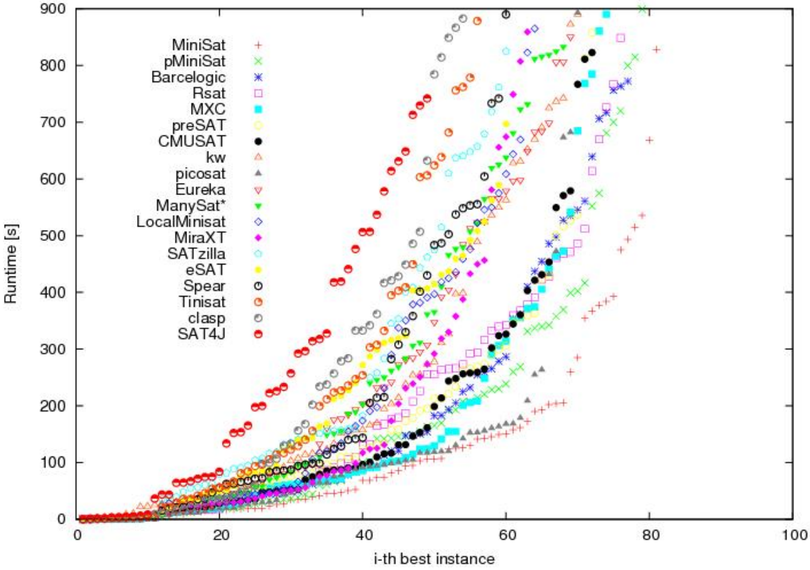
- Chaff holds a list of unassigned variables sorted by the counter value.
- Updates are needed only when adding conflict causes.
- Thus - decision is made in constant time.

VSIDS is a 'quasi-static' strategy:

- static because it doesn't depend on current assignment
- dynamic because it gradually changes. Variables that appear in recent conflicts have higher priority.

  This strategy is a conflict-driven decision strategy.

"...employing this strategy dramatically (i.e., an order of magnitude) improved performance..."



taken from http://baldur.iti.uka.de/sat-race-2008/analysis.html