**Simple Data Structures in DP Implementation**

- Variables: 1, 2, 3, -1, -2, -3
- Clauses: -21, -1 2, -2 -1, -1 -23, 1, 2, 3, -3 2

**BCP Implementation Details**

- Each variable is marked as *unassigned*, *false*, or *true* \( \{X, 0, 1\} \)

- No explicit resolution:
  - When a literal is assigned, visit all clauses where its negation occurs.
  - Find those clauses which have all but one literal assigned to false.
  - Assign remaining non-false literal to *true* and continue.

- Decision:
  - Heuristically find a variable that is still unassigned.
  - Heuristically determine phase for assignment of this variable.

**More Implementation Details**

- **Decision level** is the depth of recursive calls (= #nested decisions).

- The **trail** is a stack to remember order in which variables are assigned.

- For each decision level, the old trail height is saved on the **control stack**.

- Undeleting assignments in backtracking:
  - Get old trail height from control stack.
  - Unassign all variables up to the old trail height.

**BCP Example**

- Variables: \(X, 1, 2, 3, 4, 5\)
- Clauses: \(-1 -2, -1 2, 1 -2, 3 -1 -2, -3 1, -3 2\)
**Decision Heuristics**

- **static heuristics:**
  - one linear order determined before solver is started
  - usually quite fast, since only calculated once
  - can also use more expensive algorithms

- **dynamic heuristics**
  - typically calculated from number of occurrences of literals (in unsatisfied clauses)
  - rather expensive, since it requires traversal of all clauses (or more expensive updates in BCP)
  - recently, second order dynamic heuristics (Chaff)

**Cut Width Heuristics**

- view CNF as a graph:
  - clauses as nodes, edges between clauses with same variable

- a cut is a set of variables that splits the graph in two parts

- recursively find short cuts that cut of parts of the graph

- static or dynamically order variables according to the cuts

---

**Partial Assignment and Partial Control**

- Assign
  - decision level
  - Control
  - Trail

- Variables
  - Assignment
  - Clauses

- BCP
  - decision level
  - Control
  - Trail

- Variables
  - Assignment
  - Clauses

---

**Assumption**

- assume no occurrences of $1, 2, -1, -2$ on the right side
**Cut Width Algorithm**

```c
int sat(CNF cnf)
{
    SetOfVariables cut = generate_good_cut(cnf);
    CNF assignment, left, right;

    left = cut_off_left_part(cut, cnf);
    right = cut_off_right_part(cut, cnf);

   forall_assignments(assignment, cut)
    {
        if (sat(apply(assignment, left)) && sat(apply(assignment, right)))
            return 1;
    }

    return 0;
}
```

**Cut Width Heuristics cont.**

- resembles cuts in circuits when CNF is generated with Tseitin transformation
- ideally cuts have constant or logarithmic size . . .
  - for instance in tree like circuits
  - so the problem is reconvergence: the same signal / variable is used multiple times
- . . . then satisfiability actually becomes polynomial (see exercise)

**CNF in Horn Form**

A clause is called **positive** if it contains a positive literal.

A clause is called **negative** if all its literals are negative.

A clause is a **Horn** clause if contains at most one positive literal.

CNF is in **Horn Form** iff all clauses are Horn clause (Prolog without negation)

Order assignments point-wise: $\sigma \leq \sigma'$ iff $\sigma(x) \leq \sigma'(x)$ for all $x \in V$

Horn Form with only positive clauses has minimal satisfying assignment.

Minimal satisfying assignment is obtained by BCP (polynomial).

A Horn Form is satisfiable iff the minimal assignments of its positive part satisfies all its negative clauses as well.

**DP and Horn Form**

- CNF in Horn Form: use above specialized fast algorithm
- non Horn: split on literals which occurs positive in non Horn clauses
  - actually choose variable which occurs most often in such clauses
- this gradually transforms non Horn CNF into Horn Form
- main heuristic in SAT solver SATO
- **Note**: In general, BCP in DP prunes search space by avoiding assignments incompatible to minimal satisfying assingment for the Horn part of the CNF.
Other popular Decision Heuristics

- Dynamic Largest Individual Sum (DLIS)
  - fastest dynamic first order heuristic (eg GRASP solver)
  - choose literal (variable + phase) which occurs most often
  - ignore satisfied clauses
  - requires explicit traversal of CNF (or more expensive BCP)

- Look-forward heuristics (eg SATZ solver)
  - do trial assignments and BCP for all unassigned variables (both phases)
  - if BCP leads to conflict, force toggled assignment of current trial decision
  - skip trial assignments implied by previous trial assignments
    (removes a factor of $|V|$ from the runtime of one decision search)