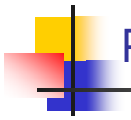
Answer Set Programming with Clause Learning

Jeffrey Ward and John S. Schlipt

Fall 2005 @ KLAB.CS.NMSU.EDU

Outlines

- Background.
- Conflict Clause Generation.
- Search Heuristics.
- The Experiments



Propositional CNF-SAT solvers

Davis –Putnam- Loveland- Logeman (DPLL)

```
srch4ModlExtndng (partlAssgn)
    while there exists a unit clause c
        let unitLit be the remaining literal in c
        partlAssgn := partlAssgn union {unitLit}
        if any clause has been falsified, return (* backtrack *)
    if partlAssgn is total (contains each variable or its negation)
        output 'SAT', output partlAssgn, and halt program
    else
        pick ltrl to guess next (i.e., branch on) - by a heuristic
        srch4ModlExtndng(partlAssgn union {ltrl})
        srch4ModlExtndng(partlAssgn union {not ltrl})
if partlAssgn is empty (* back at top level *) output 'unSAT'
```

Conflict Clause Generation

Since in SAT solver, inferences are derived through unit propagation, if x and ~ x are derived, it have been inferred since the latest choice made by another brancher.

When the solver encounters a contradiction:
it does a critical path analysis to choose a conflict clause.

If λ_0 is the last choice of the brancher, reconstruct the sequence of inferences used to infer each truth assignments since the last choice

Implication graph:

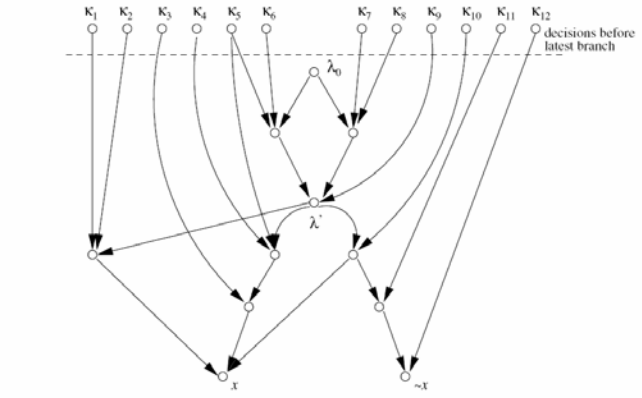
- The nodes are literals
- There is an edge from λ_1 literal to λ_2 if a clause $\{\lambda_2, \sim \lambda_1, \ldots\}$ was used to infer λ_2

Conflict Clause Generation (Cont..)

- There exist at least one directed path in the implication graph from λ_0 to ~x and at least one directed path in the implication graph from λ_0 to x.
- A node on all directed paths is called a unique implication point (UIP).
- Pick UIP λ' farthest from λ_0
- The clause contains λ' plus some literals that have been derived or guessed before the branching on λ_0 .

Conflict Clause Generation

Examples



Inference Rules

1) Forward inference.

If all the subgoals in a rule

a:- b_1 , . . . , b_k , not c_1 , . . . , not c_m

are true in the current assignment, infer a. Add edges from all b_i 's and ~c_j 's to a in the implication graph.

2) Kripke-Kleene negation (all rules canceled):

If every rule with head a has at least one subgoal negated in the current truth assignment, infer ~a. For each rule

a:- b_1 , . . . , b_k , not c_1 , . . . , not c_m

with head a, determine the canceling assignment, ~b_i or c_j , which was guessed or inferred first (at the earliest level of the backtracking search), and add an edge from that assignment to #a in the implication graph.

Inference Rules

3) Contraposition for true heads.

If atom a is true in the current truth assignment, and if every rule with head a except one has at least one sub-goal that is false in the current truth assignment, infer all the sub-goals of that remaining rule to be true.

Example

Only rules with a in their heads are:

a:- b, c, not d;

a:- e, f ;

a:- not g, h;

and that the current truth assignment contains a, d, ~e. Then ~g, h will be inferred. Add edges from each of a, d, ~e to each of ~g, h into the implication graph.

Inference Rules

4) Contraposition for false heads.

If an atom a is false in the current truth assignment and some rule

a:- b_1 , . . . , b_k

has every b_i except one true in the current truth assignment, infer that last b_i to be false.

Example:

The rule is a:- b, c, not d

and if ~a, b, c are in the truth assignment.

Infer d, and add edges from each of ~a, b, c to d.

Inference Rules

5) Well-founded negation (Smodels' AtMost):

Temporarily removing all satisfied and un-defined negative sub-goals in all rules of the program yields a Horn program. Compute its least model M ; the set of atoms false in M is unfounded; set these atoms to false in the current partial assignment.

Example

P contains the rules

a:- b; b:- c; c:- a; a:- d;

are the only rules with a, b, or c in their heads.

~d is in the current partial truth assignment.

Infer ~a, ~b, and ~c.

Add edges from ~d to ~a, from ~a to ~c, from ~c to ~b, and from ~b to ~a.

Note the implication graph contains a cycle.

Search Heuristics

- Each variable x has an “activity” count that counts the number of time that either x or ~x has been involved in producing a conflict.
- Choose the branching literal such that its activity count is maximized.

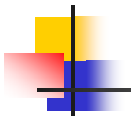
Experiment Results

Median and max secs on 11 random 3-coloring problems, uniform distribution of edges

Data set		Smodels		$Smodels_{cc}$		ASSAT		Cmodels		#Sat
vertices	edges	median	max	median	max	median	max	median	max	
400	900	3.4	163.0	1.9	197.2	0.3	88.3	0.3	88.2	10
400	950	99.5	511.8	89.8	1704.4	58.3	837.7	24.6	837.6	2
400	1000	20.7	134.3	16.0	68.4	3.7	14.3	4.0	13.1	0
500	1100	0.8	4.2	1.4	6.4	0.2	2.4	0.3	2.2	11
500	1150	196.0	356.6	697.9	2 abort	44.4	2 abort	188.3	2 abort	10
500	1200	2753.9	5 abort	>3600	8 abort	>3600	8 abort	>3600	8 abort	0-5

Median and max secs on 11 random 3-coloring problems, clumpy distribution of edges

Data set		Smodels		$Smodels_{cc}$		ASSAT		Cmodels		#Sat
vertices	edges	median	max	median	max	median	max	median	max	
10000	15150	256.1	1 abort	21.3	50.5	8.0	9.7	8.4	10.1	10
10000	17170	201.6	3 abort	19.2	21.7	8.5	11.4	10.1	11.8	7
10000	19190	>3600	8 abort	8.5	223.4	3.2	12.9	4.2	11.7	2



Complication introduced by Unfounded Set Inference Rule

As smodels utilize the Unfounded Set inference,
it is possible that:

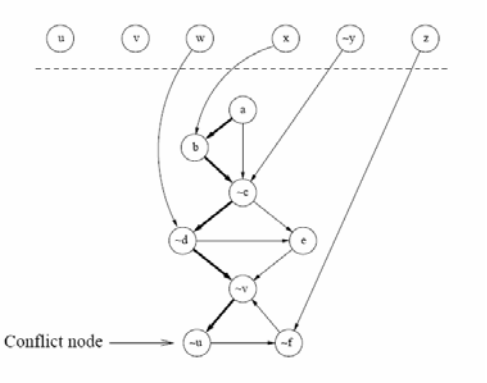
1) The implication graph have a circle.
2) Exist a conflict pair of node x and ~x where x was inferred prior to the current search level
3) There to be multiple pairs of conflicting literals appearing simultaneously.

Compute clause in smodels

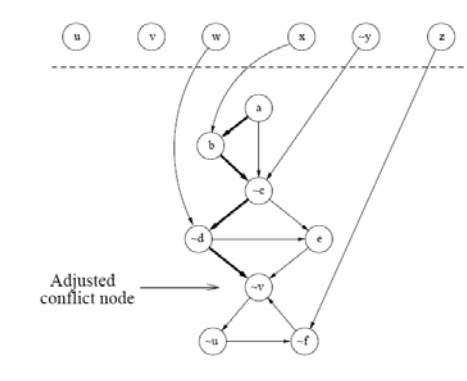
Implication graph G, choice node C, specified conflict node X

1) Compute path G that is an arbitrary acyclic path from C to X
2) Adjust conflict node selection
3) Create an additional edge to the adjusted conflict node
4) Compute Unique Implication Point
5) Traverse backward from the conflict node to find the clause

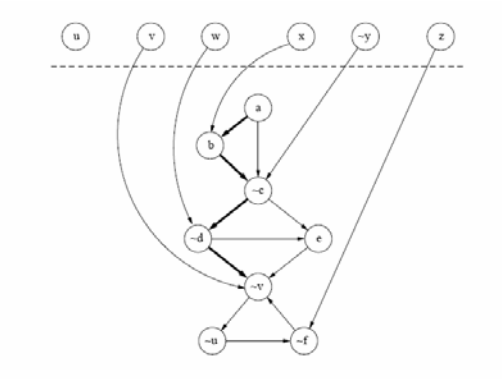
Compute clause in smodels



Compute clause in smodels



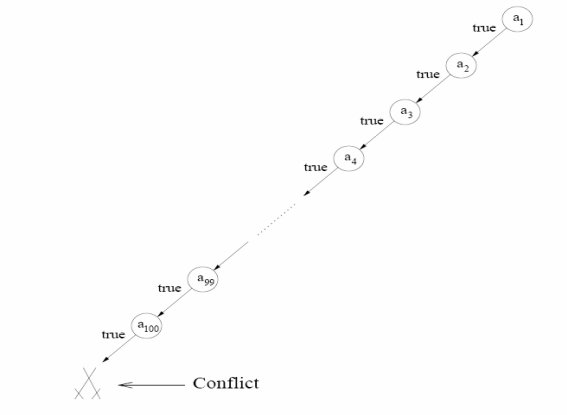
Compute clause in smodels



Using Conflict Clause

1) Backjumping
2) Serving as Additional Constraint
3) Search Heuristic
4) Restart from the root node periodically

Backjumping



Would you like to have any question?


Thank You!