

Automatic Planning

Jörg Hoffmann

University of Innsbruck
STI

Masterstudium Informatik Vertiefungsmodul (WM 9)

- ▶ An Example
- ▶ Implication Graphs
- ▶ Nonchronological Backtracking
- ▶ Clause Learning
- ▶ Historical Remarks
- ▶ Isomorphy Inference



An Example

$$P = \{A, B, C, X_1, \dots, X_{12}\}$$

1. $\{\neg A, \neg B, C\}$
2. $\{\neg A, \neg B, \neg C\}$
3. $\{\neg A, B, C\}$
4. $\{\neg A, B, \neg C\}$
5. $\{X_1, \dots, X_{12}\}$
6. $\{\neg X_1, \dots, \neg X_{12}\}$

DPLL: Blackboard.



Automatic Planning 11. Clause Learning

- ▶ An Example
- ▶ Implication Graphs
- ▶ Nonchronological Backtracking
- ▶ Clause Learning
- ▶ Historical Remarks
- ▶ Isomorphy Inference



The state of search at some given time point during DPLL is a set of time-stamped literals: what literal was set to TRUE at what time (level in the search tree)? **Choice** and **implied** literals are distinguished.

Implication graphs capture this, along with the reasons for UP:

- ▶ Nodes are all time-stamped literals $l@t$
- ▶ Edges $(l@t, l'@t')$ indicate that l participated in a unit propagation deriving l'
- ▶ Each $l'@t'$ has k edges $(l_i@t_i, l'@t')$ where $\{\bar{l}_1, \dots, \bar{l}_k, l'\}$ is a clause
- ▶ This is a DAG (directed acyclic graph). *Why?*
- ▶ *What are the nodes with indegree 0?*

$$P = \{A, B, C, X_1, \dots, X_{12}\}$$

1. $\{\neg A, \neg B, C\}$
2. $\{\neg A, \neg B, \neg C\}$
3. $\{\neg A, B, C\}$
4. $\{\neg A, B, \neg C\}$
5. $\{X_1, \dots, X_{12}\}$
6. $\{\neg X_1, \dots, \neg X_{12}\}$

Implication graph of $\{A@1, X_1@2, \dots, X_{10}@11, B@12, C@12, \neg C@12\}$: Blackboard.



An Example

$$P = \{A, B, C, D, X_1, \dots, X_{12}\}$$

1. $\{\neg A, \neg B, C\}$
2. $\{\neg A, \neg B, \neg C\}$
3. $\{\neg B, D\}$
4. $\{\neg A, B, C\}$
5. $\{\neg A, B, \neg C\}$
6. $\{X_1, \dots, X_{12}\}$
7. $\{\neg X_1, \dots, \neg X_{12}\}$

Implication graph of $\{A@1, X_1@2, \dots, X_{10}@11, B@12, C@12, \neg C@12, D@12\}$: Blackboard.



Conflict Graphs

A variable v such that both $v \in N$ and $\neg v \in N$ is called **conflict variable**; v and $\neg v$ are called **conflict literals**.

A **conflict graph** C is a sub-graph of the implication graph that captures a conflict and its reasons:

- ▶ \perp is a node in C , and there is exactly one conflict variable whose literals are nodes in C
- ▶ All nodes in C have a path (in C) to \perp
- ▶ Every node l_j in C other than \perp either corresponds to a decision literal or has precisely the nodes l_1, \dots, l_k as predecessors where $\{\bar{l}_1, \dots, \bar{l}_k, l_j\} \in \Phi$

In words, we backchain, over the clauses, from one conflict variable until we reach choice literals.



$$P = \{A, B, C, X_1, \dots, X_{12}\}$$

1. $\{\neg A, \neg B, C\}$
2. $\{\neg A, \neg B, \neg C\}$
3. $\{\neg A, B, C\}$
4. $\{\neg A, B, \neg C\}$
5. $\{X_1, \dots, X_{12}\}$
6. $\{\neg X_1, \dots, \neg X_{12}\}$

Conflict graphs of $\{A@1, X_1@2, \dots, X_{10}@11, B@12, C@12, \neg C@12\}$:
Blackboard.

$$P = \{A, B, C, D, X_1, \dots, X_{12}\}$$

1. $\{\neg A, \neg B, C\}$
2. $\{\neg A, \neg B, \neg C\}$
3. $\{\neg B, D\}$
4. $\{\neg B, \neg D\}$
5. $\{\neg A, B, C\}$
6. $\{\neg A, B, \neg C\}$
7. $\{X_1, \dots, X_{12}\}$
8. $\{\neg X_1, \dots, \neg X_{12}\}$

Conflict graphs of
 $\{A@1, X_1@2, \dots, X_{10}@11, B@12, C@12, \neg C@12, D@12, \neg D@12\}$:
Blackboard.

Automatic Planning

11. Clause Learning

- ▶ An Example
- ▶ Implication Graphs
- ▶ **Nonchronological Backtracking**
- ▶ Clause Learning
- ▶ Historical Remarks
- ▶ Isomorphy Inference

Nonchronological Backtracking, I

“Analyze the reasons for a conflict, and backtrack to the last choice variable *responsible* for the conflict.”

- ▶ Switching the choice vars in between will not resolve the conflict
- ▶ Potentially exponential savings, see our example
- ▶ “Analyze reasons”: select a conflict graph, and collect the choice literals $I@t$; choose the one with highest t
- ▶ Sounds good, but it's not *that* easy ...

$$P = \{A, B, C, X_1, \dots, X_{12}\}$$

1. $\{\neg A, \neg B, C\}$
2. $\{\neg A, \neg B, \neg C\}$
3. $\{\neg A, B, C\}$
4. $\{\neg A, B, \neg C\}$
5. $\{X_1, \dots, X_{12}\}$
6. $\{\neg X_1, \dots, \neg X_{12}\}$

Conflict graphs of $\{A@1, X_1@2, \dots, X_{10}@11, B@12, C@12, \neg C@12\}$:
Blackboard.

$$P = \{A, B, C, D, X_1, \dots, X_{12}\}$$

1. $\{\neg A, \neg B, C\}$
2. $\{\neg A, \neg B, \neg C\}$
3. $\{\neg B, D\}$
4. $\{\neg B, \neg D\}$
5. $\{\neg A, B, C\}$
6. $\{\neg A, B, \neg C\}$
7. $\{X_1, \dots, X_{12}\}$
8. $\{\neg X_1, \dots, \neg X_{12}\}$

Conflict graphs of $\{A@1, X_1@2, \dots, X_{10}@11, B@12, C@12, \neg C@12, D@12, \neg D@12\}$:
Blackboard.

- ▶ Any conflict graph contains the choice literal assigned last
- ▶ Otherwise, the conflict would have occurred earlier
- ▶ So “nonchronological” always just flips the last choice literal???

$$P = \{A, B, C, X_1, \dots, X_{12}\}$$

1. $\{\neg A, \neg B, C\}$
2. $\{\neg A, \neg B, \neg C\}$
3. $\{\neg A, B, C\}$
4. $\{\neg A, B, \neg C\}$
5. $\{X_1, \dots, X_{12}\}$
6. $\{\neg X_1, \dots, \neg X_{12}\}$

Conflict graphs of $\{A@1, X_1@2, \dots, X_{10}@11, B@12, C@12, \neg C@12\}$ and $\{A@1, X_1@2, \dots, X_{10}@11, \neg B@12, C@12, \neg C@12\}$: Blackboard.

Say a conflict occurs for *both* truth values of variable v at level t :

- ▶ Let C_v be a conflict graph for $v@t$, $ch(C_v)$ be the choice literals; we have $v@t \in ch(C_v)$
- ▶ Let $C_{\neg v}$ be a conflict graph for $\neg v@t$; we have $\neg v@t \in ch(C_{\neg v})$
- ▶ Let $t^{max} := \max\{t' \mid l@t' \in ch(C_v) \cup ch(C_{\neg v}) \setminus \{v@t, \neg v@t\}\}$
- ▶ The choices $ch(C_v) \cup ch(C_{\neg v}) \setminus \{v@t, \neg v@t\}$ suffice to produce the dead-end at v
- ▶ We can backtrack to t^{max}

- ▶ An Example
- ▶ Implication Graphs
- ▶ Nonchronological Backtracking
- ▶ **Clause Learning**
- ▶ Historical Remarks
- ▶ Isomorphy Inference



Clause Learning, I

Conflict graphs encode logical implications:

$$\Phi \models \left(\bigwedge_{l@t \in ch(C)} l \implies \perp \right)$$

- ▶ If Φ holds then setting all choice literals in a conflict graph results in a conflict

This we can re-write to:

$$\Phi \models \bigvee_{l@t \in ch(C)} \bar{l}$$

- ▶ The negation of the choice literals is a valid clause!
- ▶ Adding this to the formula increases the power of UP



An Example

$$P = \{A, B, C, D, X_1, \dots, X_{12}\}$$

1. $\{\neg A, \neg B, C\}$
2. $\{\neg A, \neg B, \neg C\}$
3. $\{\neg B, D\}$
4. $\{\neg B, \neg D\}$
5. $\{\neg A, B, C\}$
6. $\{\neg A, B, \neg C\}$
7. $\{X_1, \dots, X_{12}\}$
8. $\{\neg X_1, \dots, \neg X_{12}\}$

Learned clauses of

$\{A@1, X_1@2, \dots, X_{10}@11, B@12, C@12, \neg C@12, D@12, \neg D@12\}$:
Blackboard.



- ▶ Say we obtain a conflict upon setting $l@t$
- ▶ We learn a clause $\{\bar{l}_1@t_1, \dots, \bar{l}_k@t_k, \bar{l}@t\}$ (since $l@t \in ch(C)$)
- ▶ l_1, \dots, l_k are all set by choices
- ▶ So $\bar{l}@t$ is enforced by UP!
- ▶ We get edges $(l_1@t_1, \bar{l}@t), \dots, (l_k@t_k, \bar{l}@t)$ in the implication graph
- ▶ $\bar{l}@t$ is “not a choice literal”

$$P = \{A, B, C, X_1, \dots, X_{12}\}$$

1. $\{\neg A, \neg B, C\}$
2. $\{\neg A, \neg B, \neg C\}$
3. $\{\neg A, B, C\}$
4. $\{\neg A, B, \neg C\}$
5. $\{X_1, \dots, X_{12}\}$
6. $\{\neg X_1, \dots, \neg X_{12}\}$

Learned clauses and conflict graphs of $\{A@1, X_1@2, \dots, X_{10}@11, B@12, C@12, \neg C@12\}$ and $\{A@1, X_1@2, \dots, X_{10}@11, \neg B@12, C@12, \neg C@12\}$: Blackboard.

With clause learning, we can elegantly define nonchronological backtracking:

- ▶ If the maximum time of the literals in a (any) learned clause is t^{max} , then it is valid to backtrack to t^{max}
- ▶ t^{max} can be lower than the last t iff the choice at t was just flipped due to a conflict
- ▶ This comes down to the same technique as we defined before
- ▶ More elegant formulation, learned clauses prune the future search space (“avoid similar errors”)

- ▶ An Example
- ▶ Implication Graphs
- ▶ Nonchronological Backtracking
- ▶ Clause Learning
- ▶ **Historical Remarks**
- ▶ Isomorphy Inference

- ▶ Clause learning originates from “dependency-directed backtracking” and “explanation-based (no-good) learning” techniques developed in the CSP community
- ▶ The distinguishing feature is that, here, the “no-good” is a clause, i.e., the exact same type of constraint as the rest of the problem
- ▶ While we just select $ch(C)$, much more can be done:
 - ▶ For any “cut” through C , where $ch(C)$ lies on one side and \perp lies on the other side, the literals on the left border of the cut constitute a valid clause
 - ▶ If the cut corresponds to a “unique implication point” then $l@t$ becomes flipped automatically as in our case
 - ▶ One has to take care to not learn *too many* clauses ...

Historical Remarks, III

Recent work [Beame&Kautz&Sabharwal, JAIR-2004] addresses the *reasons* for the efficiency of clause learning:

- ▶ DPLL is equivalent to a restricted form (“tree-like”) of resolution
- ▶ DPLL+clause learning (“with unlimited restarts”) is equivalent to resolution
- ▶ Resolution is exponentially more powerful than tree-like resolution!
- ▶ (Means there are formulas for which the best tree-like resolution proofs are exponentially larger)
- ▶ It seems that DPLL+clause learning provides a clever way of “arranging a resolution process”

The “best” SAT solver:

- ▶ Before 1991: DPLL
- ▶ 1991 - 1996: Hill-Climbing
- ▶ 1996 - today: Hill-Climbing in random instances, DPLL+clause learning in “structured”/“real-world” instances
- ▶ In particular, clause learning brought about a breakthrough in solving planning CNFs!
- ▶ (While FF-style planners are usually much better in finding plans, SATPLAN is best in finding *optimal* plans)

Automatic Planning
11. Clause Learning

- ▶ An Example
- ▶ Implication Graphs
- ▶ Nonchronological Backtracking
- ▶ Clause Learning
- ▶ Historical Remarks
- ▶ **Isomorphy Inference**

```

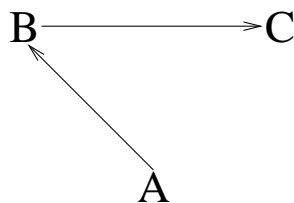
b := 0
while TRUE do
   $\Phi(b)$  := a CNF that is satisfiable iff
    there exists a plan with b steps
  if DPLL( $\Phi(b)$ ,  $\emptyset$ ) then
    output Plan encoded by satisfying assignment a
  endif
  b := b + 1
endwhile
    
```

We are considering a sequence of *rather similar* CNF formulas!

- ▶ Notations: facts P , actions A , with NOOPs A^N , initial state I , goals G , time steps $0 \leq t \leq b$
- ▶ Decision variables: p_t for all $0 \leq t \leq b$ and $p \in P$, a_t for all $0 \leq t \leq b - 1$ and $a \in A^N$
- ▶ Initial State Clauses: “specify initial values”
For all $p \in P$: $\{p_0\}$ if $p \in I$, $\{\neg p_0\}$ if $p \notin I$
- ▶ Goal Clauses: “specify goal values”
For all $g \in G$: $\{g_b\}$

- ▶ Action Precondition Clauses: “action implies its preconditions”
For all $0 \leq t \leq b - 1$, $a \in A^N$, $p \in pre(a)$: $\{\neg a_t, p_t\}$
- ▶ Action Interference Clauses: “do not apply interfering actions in the same time step”
For all $0 \leq t \leq b - 1$, $a, a' \in A^N$, $a \not\sim a'$: $\{\neg a_t, \neg a'_t\}$
[[$pre(a) \cup add(a) \cap del(a') \neq \emptyset$ or vice versa]]
- ▶ Fact Achievement Clauses: “fact implies disjunction of its achievers”
For all $1 \leq t \leq b$, $p \in P$: $\{\neg p_t\} \cup \{a_{t-1} \mid a \in A^N, p \in add(a)\}$

- ▶ Let $\phi = \{l_{t_1}^1, \dots, l_{t_k}^k\}$ be a precondition/fact achievement/action interference clause
... then ϕ appears also incremented $\phi_{++} := \{l_{t_1+1}^1, \dots, l_{t_k+1}^k\}$ and decremented $\{l_{t_1-1}^1, \dots, l_{t_k-1}^k\}$, if such t are available
- ▶ The only difference between $\Phi(b)$ and $\Phi(b + 1)$:
For all $g \in G$: $\{g_{b+1}\}$ instead of $\{g_b\}$
... and one more layer of precondition/fact achievement/action interference clauses



We can use the symmetries to “transfer” learned clauses! Termed “isomorphism inference” in the model checking context by [Strichman, CAV-2000, FMSD-2004].

The **dependency set** $DS(\phi)$ of a learned clause ϕ : the set of clauses ψ responsible for the edges in the respective conflict graph (if ψ is a learned clause itself, replace with $DS(\psi)$).

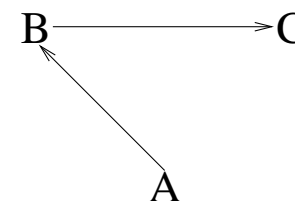
- ▶ Say $DS(\phi)$ contains *neither initial nor goal clauses*
- ▶ Then we can increment/decrement ψ up to the lowest/highest time step

$P = \{A, B, C\}, I = \{A\}, G = \{C\}, drAB = (\{A\}, \{B\}, \{A\}),$
 $drBC = (\{B\}, \{C\}, \{B\})$

Bound $b = 1, b = 2$ basic parallel encoding: Blackboard.

- ▶ Say ϕ is learned in $\Phi(b)$, and $DS(\phi)$ contains *no goal clauses*
- ▶ Then ϕ is also valid in $\Phi(b + 1)$

- ▶ Say ϕ is learned in $\Phi(b)$, and $DS(\phi)$ contains *no initial clauses*
- ▶ Then $\phi++$ is valid in $\Phi(b + 1)$



$P = \{A, B, C\}, I = \{A\}, G = \{C\}, drAB = (\{A\}, \{B\}, \{A\}),$
 $drBC = (\{B\}, \{C\}, \{B\})$

Bound $b = 1, b = 2$ isomorphism inference: Blackboard.

- ▶ [Strichman, CAV-2000, FMSD-2004] fails to observe that one can also transfer clauses dependent on the goal
- ▶ This is important! Without a goal, the CNFs are always gonna be satisfiable!
- ▶ In the planning context, there exists only a very preliminary implementation
- ▶ In general (not in the basic parallel encoding), matters are a little more complicated . . .
- ▶ . . . which is also why Strichman overlooked the goal-dependent transfers

- ▶ Isomorphy inference may even give us a useful **termination criterion** for the search:

Did we learn the empty clause independently of the initial state?

Didn't we learn any new clauses??

- ▶ This topic is *wide open* . . .

- ▶ J. Marques-Silva and K. Sakallah, *GRASP: A Search Algorithm for Propositional Satisfiability*, IEEE Transactions on Computers, 1999
- ▶ M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, *Chaff: Engineering an Efficient SAT Solver*, Proceedings of the 38th Design Automation Conference (DAC-2001)
- ▶ L. Zhang, C. Madigan, M. Moskewicz, C, and S. Malik, *Efficient Conflict-Driven Learning in a Boolean Satisfiability Solver*, Proceedings of the International Conference on Computer Aided Design 2001 (ICCAD-2001)
- ▶ P. Beame, H. Kautz, and A. Sabharwal, *Towards Understanding and Harnessing the Potential of Clause Learning*, JAIR, 2004

- ▶ O. Strichman, *Tuning SAT Checkers for Bounded Model Checking*, 12th International Conference on Computer Aided Verification (CAV-2000)
- ▶ O. Strichman, *Accelerating Bounded Model Checking of Safety Formulas*, Formal Methods in System Design, 2004