universität innsbruck

SS 2024 lecture 2

Constraint Solving

René Thiemann and Fabian Mitterwallner

based on a previous course by Aart Middeldorp

Outline

1. **Summary of Previous Lecture**
2. **Conflict Graphs**
3. **NP-Completeness of SAT**
4. **SAT Reductions**
5. **Further Reading**

Theorem

propositional formula φ is valid $\iff$ $\neg\varphi$ is unsatisfiable

Definitions

- literal is atom p or negation $\neg p$ of atom
- clause is disjunction of literals
- conjunctive normal form (CNF) is conjunction of clauses
- disjunctive normal form (DNF) is disjunction of conjunctions of literals

Theorem

$\forall$ formula φ $\exists$ CNF ψ $\exists$ DNF χ such that $\varphi \equiv \psi \equiv \chi$

Remark

Tseitin's transformation is linear-time translation to equisatisfiable CNF

Definition (Abstract DPLL)

- states $M \parallel F$ consist of list M of (possibly annotated) non-complementary literals and CNF F
- transition rules
  - unit propagate $\qquad M \parallel F, C \vee l \implies M\, l \parallel F, C \vee l$

    if $M \vDash \neg C$ and l is undefined in M
  - pure literal $\qquad M \parallel F \implies M\, l \parallel F$

    if l occurs in F and l^c does not occur in F and l is undefined in M
  - decide $\qquad M \parallel F \implies M\, l^d \parallel F$

    if l or l^c occurs in F and l is undefined in M

Definition (Abstract DPLL, cont'd)

- fail $\qquad M \parallel F, C \quad \Longrightarrow \quad$ fail-state

  if $M \models \neg C$ and M contains no decision literals

- backtrack $\qquad M\, l^{d}\, N \parallel F, C \quad \Longrightarrow \quad M\, l^{c} \parallel F, C$

  if $M\, l^{d}\, N \models \neg C$ and N contains no decision literals

- backjump $\qquad M\, l^{d}\, N \parallel F, C \quad \Longrightarrow \quad M\, l' \parallel F, C$

  if $M\, l^{d}\, N \models \neg C$ and $\exists$ clause $C' \vee l'$ such that

  backjump clause

  - $F, C \models C' \vee l'$
  - $M \models \neg C'$
  - l' is undefined in M
  - l' or l'^{c} occurs in F or in $M\, l^{d}\, N$

Definition

basic DPLL $\mathcal{B}$ consists of transition rules unit propagate, decide, fail, backjump

Theorem

- there are no infinite derivations $\parallel F \Longrightarrow_{\mathcal{B}} S_1 \Longrightarrow_{\mathcal{B}} S_2 \Longrightarrow_{\mathcal{B}} \cdots$
- if $\parallel F \Longrightarrow_{\mathcal{B}} S_1 \Longrightarrow_{\mathcal{B}} \cdots \Longrightarrow_{\mathcal{B}} S_n \not\Longrightarrow_{\mathcal{B}}$ then
  1. $S_n =$ fail-state if and only if F is unsatisfiable
  2. $S_n = M \parallel F'$ only if F is satisfiable and $M \vDash F$

Outline

1. Summary of Previous Lecture
2. **Conflict Graphs**
3. NP-Completeness of SAT
4. SAT Reductions
5. Further Reading

Problem: How to obtain backjump clauses

- backjump $M \overset{d}{l} N \parallel F, C \implies M\, l' \parallel F, C$

  if $M \overset{d}{l} N \vDash \neg C$ and ... (some more conditions; involves finding a backjump clause)
- situation: complicated looking rule; unclear how to obtain backjump clause
- solution
  - store information of applied rules (unit propagate, decide, ...) in conflict graph
  - cuts in conflict graphs separate conflict node from current decision literal and literals at earlier decision levels
  - cuts that correspond to unique implication points (UIPs) generate backjump clauses

click to access overlay version of slides for example and explanation of conflict graph, unique implication point, etc.

Remarks

- computed clauses are clauses that correspond to cut in conflict graph, a set of edges that separate conflict node from current decision literal and literals at earlier decision levels
- clause is computed by negating all literals that are a source of an edge in the cut
- clauses corresponding to UIPs are backjump clauses
- UIPs always exist (last decision literal)
- backjumping with respect to last UIP amounts to backtracking
- when applying backjump rule, backjump clause is used to update conflict graph
- most SAT solvers use backjump clause corresponding to 1st UIP

Observation

adding backjump clauses to clause database (learning) helps to prune search space

- learn $\quad M \parallel F \implies M \parallel F, C$

  if $F \vDash C$ and each atom of C occurs in F or in M

Observation

restarts are useful to avoid wasting too much time in parts of search space without satisfying assignments

- restart $\quad M \parallel F \implies \parallel F$

Final Remarks

- restarts do not compromise completeness if number of steps between consecutive restarts strictly increases
- modern SAT solvers additionally incorporate
  - heuristics for selecting next decision literal
  - special data structures that allow for efficient unit propagation

Outline

1. Summary of Previous Lecture
2. Conflict Graphs
3. **NP-Completeness of SAT**
4. SAT Reductions
5. Further Reading

Definitions

- P is class of decision problems that can be solved in polynomial time by deterministic Turing machine
- NP is class of decision problems that can be solved in polynomial time by non-deterministic Turing machine
- decision problem A is NP-hard if every NP problem B is polynomial-time reducible to A
- decision problem A in NP-complete if it is NP-hard and in NP

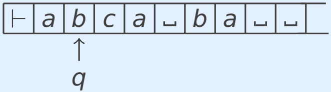
Famous Open Problem

P $=$ NP ?

Definition

non-deterministic TM (NTM) is 8-tuple $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \Delta, s, F)$ with

1. Q: finite set of states
2. Σ: input alphabet
3. $\Gamma \supseteq \Sigma$: tape alphabet
4. $\vdash \in \Gamma - \Sigma$: left endmarker
5. $\sqcup \in \Gamma - \Sigma$: blank symbol
6. $\Delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R\}}$: transition function
7. $s \in Q$: start state
8. $F \subseteq Q$: final states



such that

$$\forall p \in F \quad \forall a \in \Gamma : \quad \Delta(p, a) = \varnothing$$

$$\forall p \in Q \quad \forall (q, b, d) \in \Delta(p, \vdash) : \quad b = \vdash \text{ and } d = R$$

Definitions

- configuration: element of $Q \times \{ y\sqcup^\omega \mid y \in \Gamma^* \} \times \mathbb{N}$
- start configuration on input $x \in \Sigma^*$: $(s, \vdash x \sqcup^\omega, 0)$
- next configuration relation is binary relation $\xrightarrow[M]{1}$ defined as:

$$(p, z, n) \xrightarrow[M]{1} \begin{cases} (q, z', n-1) & \text{if } (q, b, L) \in \Delta(p, z_n) \\ (q, z', n+1) & \text{if } (q, b, R) \in \Delta(p, z_n) \end{cases}$$

  with

  - z_n: n-th symbol of z
  - z': string obtained from z by substituting b for z_n (at position n)

- $\xrightarrow[M]{n} = (\xrightarrow[M]{1})^n \quad \forall n \geqslant 0$ $\qquad \xrightarrow[M]{*} = \bigcup_{n \geqslant 0} \xrightarrow[M]{n}$
- $x \in \Sigma^*$ is accepted by M if $(s, \vdash x \sqcup^\omega, 0) \xrightarrow[M]{*} (q, y, n)$ for some $q \in F$, y, n

Theorem (Cook-Levin)

SAT is NP-complete

Lemma

SAT is in NP

Proof Sketch

- use non-deterministic ability of NTM to guess truth assignment
- verify in polynomial time whether it is satisfying assignment

Theorem

SAT is NP-hard

Proof

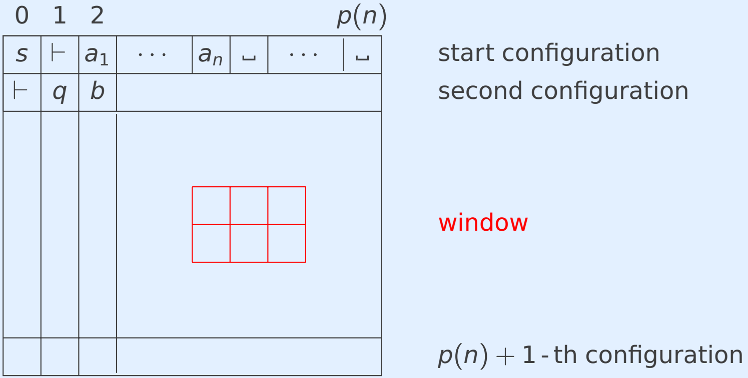
- let A be arbitrary decision problem in NP
- task: define polynomial-time reduction from A to SAT
- (language encoding of) A is accepted by NTM $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \Delta, s, F)$ that runs in polynomial time
- $\exists$ polynomial $p(n)$ such that M halts in at most $p(n)$ steps for any input x of length n
- given input x, we construct CNF formula $\varphi_M(x)$ of polynomial size such that

$$M \text{ accepts } x \quad \Longleftrightarrow \quad \varphi_M(x) \text{ is satisfiable}$$

- assumption (WLOG): $\alpha \xrightarrow[M]{1} \alpha$ for every halting configuration α

Proof (cont'd)

- every computation of M on x can be recorded in $(p(n)+1) \times (p(n)+1)$ sized table containing successive configurations



- properties of accepting table can be encoded in formula $\varphi_M(x)$

Proof (cont'd)

- variables $\langle i, j, a\rangle$ for all $0 \leqslant i, j \leqslant p(n)$ and $a \in \Gamma \cup Q$

  $\langle i, j, a\rangle$ is true if cell at position (i, j) contains symbol a

- $\varphi_M(x) \;=\; \varphi_{\text{cell}} \;\wedge\; \varphi_{\text{start}} \;\wedge\; \varphi_{\text{move}} \;\wedge\; \varphi_{\text{accept}}$

- φ_{cell}

$$\bigwedge_{i,j}\left[\bigvee_{a}\langle i,j,a\rangle \;\wedge\; \bigwedge_{a \neq b}\left(\neg\langle i,j,a\rangle \vee \neg\langle i,j,b\rangle\right)\right]$$

- φ_{start} for input $x = a_1 \cdots a_n$

$$\langle 0,0,s\rangle \;\wedge\; \langle 0,1,\vdash\rangle \;\wedge\; \langle 0,2,a_1\rangle \;\wedge\; \cdots \;\wedge\; \langle 0,n+1,a_n\rangle \;\wedge\; \langle 0,n+2,\sqcup\rangle \;\wedge\; \cdots \;\wedge\; \langle 0,p(n),\sqcup\rangle$$

- φ_{accept}

$$\bigvee_{i,j}\bigvee_{q \in F}\langle i,j,q\rangle$$

Proof (cont'd)

- φ_{move} $\quad \displaystyle\bigwedge_{0 \leqslant i < p(n)} \; \bigwedge_{0 \leqslant j < p(n)-1} \varphi^{i,j}_{\text{window}}$

- $\varphi^{i,j}_{\text{window}}$

$$\bigvee_{\substack{\begin{array}{|c|c|c|}\hline a_1 & a_2 & a_3 \\ \hline b_1 & b_2 & b_3 \\ \hline\end{array}\\ \text{is legal window}}} \begin{array}{l} \langle i,j,a_1\rangle \wedge \langle i,j+1,a_2\rangle \wedge \langle i,j+2,a_3\rangle \wedge \\ \langle i+1,j,b_1\rangle \wedge \langle i+1,j+1,b_2\rangle \wedge \langle i+1,j+2,b_3\rangle \end{array}$$

Example

suppose $\Delta(p,a) = \{(q,b,R)\}$ and $\Delta(p,b) = \{(p,c,L),(q,a,R)\}$

a	p	b
p	a	c

~~b~~	~~a~~	~~b~~
~~b~~	~~c~~	~~b~~

a	a	p
a	a	b

a	b	a
a	b	a

~~p~~	~~b~~	~~a~~
~~c~~	~~b~~	~~a~~

b	a	b
c	a	b

~~b~~	~~q~~	~~b~~
~~q~~	~~b~~	~~q~~

Outline

1. Summary of Previous Lecture
2. Conflict Graphs
3. NP-Completeness of SAT
4. **SAT Reductions**
5. Further Reading

SAT Variations

- 3SAT: every clause has (at most) 3 literals
- 2SAT: every clause has (at most) 2 literals

Theorem

- 3SAT is NP-complete
- 2SAT is solvable in polynomial time

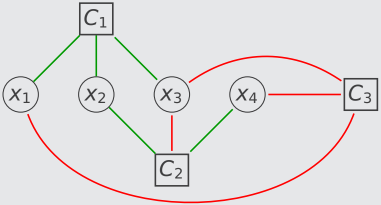
Planar 3SAT

instance is 3SAT formula φ whose incidence graph is planar

- φ with clauses $\mathcal{C} = \{C_1, \dots, C_m\}$ over variables $\mathcal{V} = \{x_1, \dots, x_n\}$
- bipartite graph $(\mathcal{C} \cup \mathcal{V}, \mathcal{E})$ with $\mathcal{E}$ containing edge $C_i - x_j$ if and only if C_i contains x_j or $\neg x_j$

Example

CNF $\varphi = \{\underbrace{\{x_1, x_2, x_3\}}_{C_1}, \underbrace{\{x_2, \neg x_3, x_4\}}_{C_2}, \underbrace{\{\neg x_1, \neg x_3, \neg x_4\}}_{C_3}\}$



planar 3SAT instance

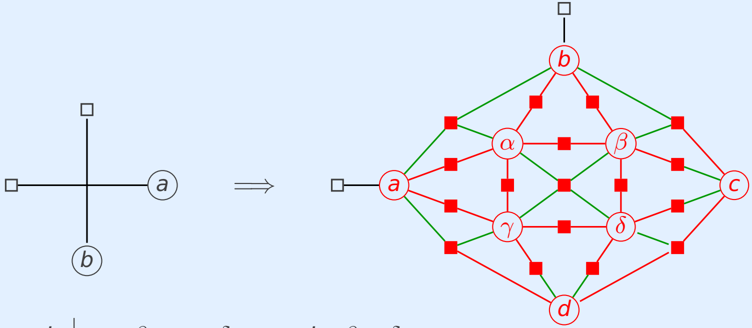
Theorem (Lichtenstein 1982)

planar 3SAT is NP-complete

Remark

planar 3SAT is often used in reductions to show NP-hardness of particular problems

Main Idea (Crossover Gadget)



6 new variables

17 new clauses

■ $a \vee \gamma \vee \neg d$

claim: $c = a$ and $d = b$

a	b	α	β	γ	δ	c	d	$\beta \vee \delta$
0	0	1	0	0		0	0	
0	1							
1	0	0	1	0	0	1	0	1
1	1							

Outline

1. Summary of Previous Lecture
2. Conflict Graphs
3. NP-Completeness of SAT
4. SAT Reductions
5. **Further Reading**

Kröning and Strichmann

- Section 2.2

Further Reading

- Stephen A. Cook
  The Complexity of Theorem-Proving Procedures
  Proc. 3rd ACM SToC, pp. 151 – 158, 1971

Further Viewing

- Erik Demaine
  Algorithmic Lower Bounds: Fun with Hardness Proofs
  MIT OpenCourseWare, 2014

Important Concepts

- 2SAT
- 3SAT
- conflict graph
- crossover gadget
- cut
- incidence graph
- learning
- NP
- NP-hard
- NP-complete
- P
- planar 3SAT
- reduction
- restart
- unique implication point