Algorithms and Satisfiability

7. Satisfiability, Part I: Principles and Basic Algorithms

How to Think About What is True or False

Álvaro Torralba

AALBORG UNIVERSITET

Spring 2023

Thanks to Jörg Hoffmann for slide sources

Agenda

So far...

1. Dynamic Programming
2. Greedy Algorithms
3. Computational geometry algorithms: sweeping techniques
4. External-memory algorithms and data structures
5. Parallel algorithms
6. Amortized analysis

$\rightarrow$Techniques to make efficient algorithms and analyze their performance

What if an efficient algorithm does not exist?

What to do when you can't find an efficient algorithm?[1]

"I can't find an efficient algorithm, I guess I'm just too dumb."

In this lecture:

In this lecture we will study algorithms for:

1. Satisfiability: Given a Boolean formula, is it satisfiable?
   $\rightarrow$**NP**-complete

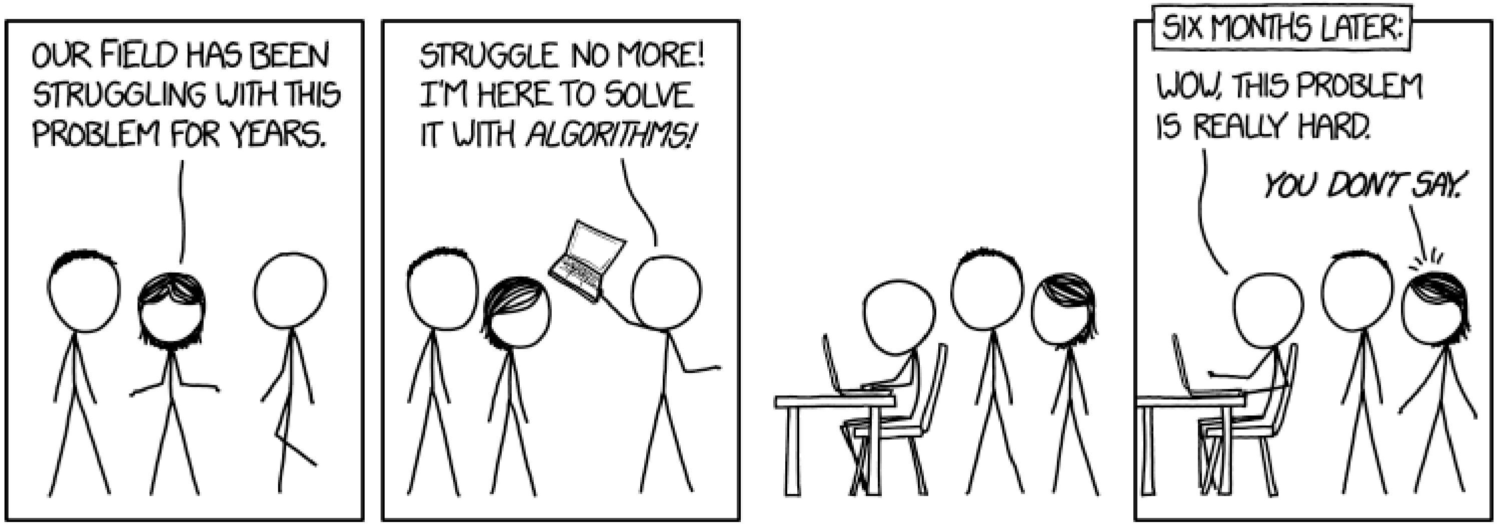
2. Classical Planning: Can we achieve our goal by applying a sequence of actions?
   $\rightarrow$**PSPACE**-complete

$\rightarrow$These problems cannot be solved in polynomial time (unless **P**=**NP**)!

But that is only worst case assymptotic complexity...

- Do the interesting real-world instances pertain to the worst case?
- Are interesting real-world instances small enough so that we can solve them?

You say I can't solve that? Hold my beer!



"We TOLD you it was hard." "Yeah, but now that I'VE tried, we KNOW it's hard." XKCD.com/1831

Two Questions

What algorithms can we use to solve these hard problems?
$\rightarrow$Explored in the Lectures and Exercises

How to solve these hard problems in practice using solvers?
$\rightarrow$Explored in the mini-projects

- Preparation: Install tool in your computer
- (shorter) Lecture
- Project: Use a solver to solve some problems
- Optional: submit your solution to receive feedback
- Exam relevant!: in 2021 no one submitted their solution and at least half of the students failed to answer the exam question!

The SAT Problem

SAT

Is a propositional logic formula ϕ satisfiable?

- Propositional Logic: Satisfiability can refer to many different logics. In this course, we focus on one of the simplest!
- Satisfiable: A formula is satisfiable if it is possible to find an interpretation (assignment) that makes the formula true.

$\rightarrow$Does there exist an assignment that makes the formula true?
Examples:

- $(x \vee y) \wedge (\neg x \vee \neg y)$, Yes! $x = 0, y = 1$
- $(x \vee y) \wedge (\neg x) \wedge (\neg y)$, No!

Our Agenda for This Topic

→ Our treatment of the topic "SAT Solving" consists of Chapters 7 and 8.

- **This Chapter:** Basic definitions and concepts; resolution; DPLL.

  → Sets up the framework. Resolution is the quintessential reasoning procedure underlying most successful solvers.

- **Chapter 8:** Clause learning; practical problem structure.

  → State-of-the-art algorithms for satisfiability in propositional logic, and an important observation about how they behave.

- **Mini-project:** SAT modulo theories (SMT)

  → Extension beyond propositional formulas!

Our Agenda for This Chapter

- **Basics:** What's the SAT problem about?
  $\rightarrow$ Introduces what our problem is about.
- **Applications:** What is all this useful for?
  $\rightarrow$ Brief description of some of the applications of SAT.
- **Resolution:** How does resolution work? What are its properties?
  $\rightarrow$ Formally introduces the most basic reasoning method.
- **The Davis-Putnam (Logemann-Loveland) Procedure:** How to systematically test satisfiability?
  $\rightarrow$ The quintessential SAT solving procedure, DPLL.

Syntax of Propositional Logic

$\rightarrow$ Atoms Σ in propositional logic = Boolean variables.

Definition (Syntax). *Let Σ be a set of atomic propositions. Then:*

1. $\bot$ *and* $\top$ *are Σ-formulas.* ("False", "True")
2. *Each $P \in \Sigma$ is a Σ-formula.* ("Atom")
3. *If φ is a Σ-formula, then so is $\neg\varphi$.* ("Negation")

If φ and ψ are Σ-formulas, then so are:

4. $\varphi \wedge \psi$ ("Conjunction")
5. $\varphi \vee \psi$ ("Disjunction")
6. $\varphi \rightarrow \psi$ ("Implication")
7. $\varphi \leftrightarrow \psi$ ("Equivalence")

Notation: Atoms and negated atoms are called literals. Operator precedence: $\neg > \dots$ (we'll be using brackets except for negation).

Semantics of Propositional Logic

Definition (Interpretation). *Let Σ be a set of atomic propositions. An interpretation of Σ, also called a truth assignment, is a function $I : \Sigma \mapsto \{1, 0\}$. We set:*

$$
\begin{array}{lll}
I \models \top & & \\
I \not\models \bot & & \\
I \models P & \textit{iff} & P^I = 1 \\
I \models \neg\varphi & \textit{iff} & I \not\models \varphi \\
I \models \varphi \wedge \psi & \textit{iff} & I \models \varphi \text{ and } I \models \psi \\
I \models \varphi \vee \psi & \textit{iff} & I \models \varphi \text{ or } I \models \psi \\
I \models \varphi \rightarrow \psi & \textit{iff} & \textit{if } I \models \varphi, \textit{ then } I \models \psi \\
I \models \varphi \leftrightarrow \psi & \textit{iff} & I \models \varphi \textit{ if and only if } I \models \psi
\end{array}
$$

If $I \models \varphi$, we say that I satisfies φ, or that I is a model of φ. The set of all models of φ is denoted by $M(\varphi)$.

Semantics of Propositional Logic: Examples

Example

Formula: $\varphi = [(P \vee Q) \leftrightarrow (R \vee S)] \wedge [\neg(P \wedge Q) \wedge (R \wedge \neg S)]$

$\rightarrow$ For I with $I(P) = 1, I(Q) = 1, I(R) = 0, I(S) = 0$, do we have $I \models \varphi$? No: $(P \vee Q)$ is true but $(R \vee S)$ is false, so the left-hand side of the conjunction is false and the overall formula is false.

Example

Formula: $\varphi = \text{InSatisfiabilityClass} \rightarrow \text{HavingAGreatTime}$

$\rightarrow$ For I with $I(\text{InSatisfiabilityClass}) = 0$, $I(\text{HavingAGreatTime}) = 1$, do we have $I \models \varphi$? Yes: $\varphi = \psi_1 \rightarrow \psi_2$ is true iff either ψ_1 is false, or ψ_2 is true (i.e., $\psi_1 \rightarrow \psi_2$ has the same models as $\neg\psi_1 \vee \psi_2$).

Terminology

Satisfiability

A formula φ is:

- satisfiable if there exists I that satisfies φ.
- unsatisfiable if φ is not satisfiable.
- falsifiable if there exists I that doesn't satisfy φ.
- valid if $I \models \varphi$ holds for all I. We also call φ a tautology.

Equivalence

Formulas φ and ψ are equivalent, $\varphi \equiv \psi$, if $M(\varphi) = M(\psi)$.

Entailment

Formula φ entails ψ ($\varphi \models \psi$), if $M(\psi) \subseteq M(\varphi)$.

General Problem Solving using SAT

(some new problem)

model problem in logic ↦ use off-the-shelf SAT solver

(its solution)

- "Any problem that can be formulated as SAT."
- *Very* successful using propositional logic and modern solvers for SAT! (Propositional satisfiability testing, **Chapter 8**.)

Applications

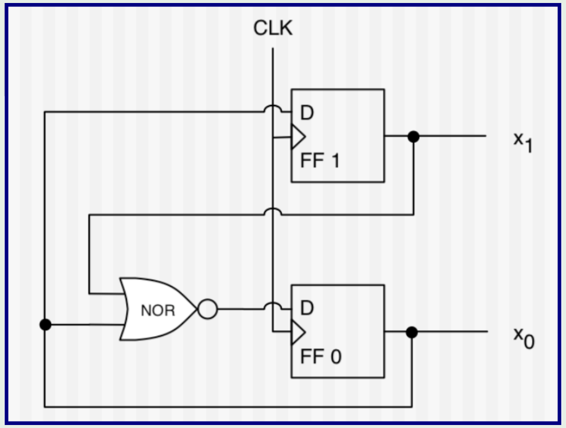
Lots of interesting problems can be formulated as SAT!

- Lots of NP problems
- Scheduling
- Hardware Verification
- Logical Deduction
- Planning **(Chapter 12)**

And we can even extend this by considering SAT modulo theories **(mini project)**

Example Application: Hardware Verification

Example



- Counter, repeatedly from $c = 0$ to $c = 2$.
- 2 bits x_1 and x_0; $c = 2 * x_1 + x_0$.
- ("FF" Flip-Flop, "D" Data IN, "CLK" Clock)
- **To Verify:** If $c < 3$ in current clock cycle, then $c < 3$ in next clock cycle.

Step 1: Encode into propositional logic.

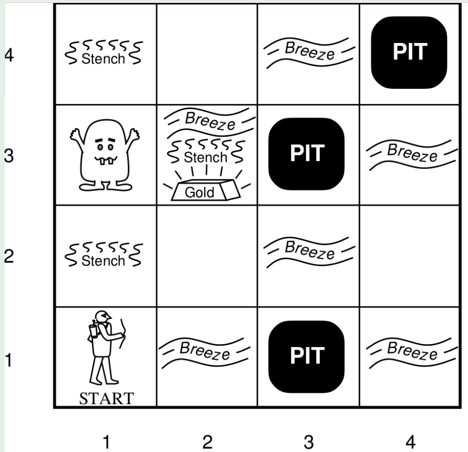
- Propositions: x_1, x_0; and x_1', x_0' (value in next cycle).
- Transition relation: $x_1' \leftrightarrow x_0$; $x_0' \leftrightarrow \neg(x_1 \vee x_0)$.
- Initial state: $\neg(x_1 \wedge x_0)$. Error property: $x_1' \wedge x_0'$.

Step 2: Transform to CNF, encode as set Δ of clauses.
$\rightarrow \{\{\neg x_1', x_0\}, \{x_1', \neg x_0\}, \{x_0', x_1, x_0\}, \{\neg x_o', \neg x_1\}, \{\neg x_0', \neg x_0\}, \{\neg x_1, \neg x_0\}, \{x_1'\}, \{x_0'\}\}$

Step 3: Call a SAT solver (up next).

Example Application: Logical Deduction (Wumpus)

Example



- The player cannot see the entire board, only if there is Stench or Breeze on the current cell.
- **To Verify:** After visiting [2,1] and [1,2], are we sure cell [2,2] is free?.

Step 1: Encode into propositional logic.

- Propositions: $Stench[i,j], Breeze[i,j], Wumpus[i,j], Pit[i,j]$;
- Knowledge Base: $KB = \bigwedge_{i,j} Wumpus[i,j] \implies Stench[i,j+1] \wedge Stench[1,2] \wedge \dots$
- Question: Q $= \neg(Wumpus[2,2] \wedge Pit[2,2])$.

Step 2: Transform $KB \wedge \neg(Q)$ to CNF, encode as set Δ of clauses.

Step 3: Call a SAT solver (up next). If unsatisfiable, then we can conclude Q.

The Truth Table Method

Want: Determine whether φ is satisfiable, valid, etc.

Method: Build the truth table, enumerating all interpretations of Σ.

Example

Is $\varphi = ((P \vee H) \wedge \neg H) \rightarrow P$ valid?

P	H	$P \vee H$	$(P \vee H) \wedge \neg H$	$(P \vee H) \wedge \neg H \rightarrow P$
0	0	0	0	1
0	1	1	0	1
1	0	1	1	1
1	1	1	0	1

$\rightarrow$ Yes. φ is true for all possible combinations of truth values.

$\rightarrow$ Is this a good method for answering these questions? No! For N propositions, the truth table has 2^N rows. [Satisfiability (validity) testing is **NP**-hard (**co-NP**-hard), but that pertains to *worst-case* behavior.]

Normal Forms

The two quintessential normal forms: (there are others as well)

- A formula is in conjunctive normal form (CNF) if it consists of a conjunction of disjunctions of literals:

$$\bigwedge_{i=1}^{n}\left(\bigvee_{j=1}^{m_i} l_{i,j}\right)$$

- A formula is in disjunctive normal form (DNF) if it consists of a disjunction of conjunctions of literals:

$$\bigvee_{i=1}^{n}\left(\bigwedge_{j=1}^{m_i} l_{i,j}\right)$$

$\rightarrow$ Given a propositional formula φ, we can in polynomial time construct a CNF/DNF formula ψ that is satisfiable if and only if φ is. (Proof omitted)

Transformation to Normal Form

CNF Transformation (DNF Transformation: Analogously)

Exploit the equivalences:

1. $(\varphi \leftrightarrow \psi) \equiv [(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)]$ (Eliminate "$\leftrightarrow$")
2. $(\varphi \rightarrow \psi) \equiv (\neg\varphi \vee \psi)$ (Eliminate "$\rightarrow$")
3. $\neg(\varphi \wedge \psi) \equiv (\neg\varphi \vee \neg\psi)$ and $\neg(\varphi \vee \psi) \equiv (\neg\varphi \wedge \neg\psi)$ (Move "$\neg$" inwards)
4. $[(\varphi_1 \wedge \varphi_2) \vee (\psi_1 \wedge \psi_2)] \equiv [(\varphi_1 \vee \psi_1) \wedge (\varphi_2 \vee \psi_1) \wedge (\varphi_1 \vee \psi_2) \wedge (\varphi_2 \vee \psi_2)]$ (Distribute "$\vee$" over "$\wedge$")

Example: $((P \vee H) \wedge \neg H) \rightarrow P$

(eliminate $\rightarrow$):	$\neg((P \vee H) \wedge \neg H) \vee P$	
(move $\neg$ inwards):	$(\neg(P \vee H) \vee H) \vee P$	$((\neg P \wedge \neg H) \vee H) \vee P$
(distr. "$\vee$" over "$\wedge$"):	$(((\neg P \vee H) \wedge (\neg H \vee H))) \vee P$	
(distr. "$\vee$" over "$\wedge$"):	$(((\neg P \vee H \vee P) \wedge (\neg H \vee H \vee P)))$	

$\rightarrow$ Note: The formula may grow exponentially! ("Distribute" step)

$\rightarrow$ However, satisfiability-preserving CNF transformation is polynomial!

Questionnaire

Question!

A CNF formula is . . .

(A): Valid iff at least one disjunction is valid.

(B): Valid iff every disjunction is valid.

(C): Satisfiable if at least one disjunction is satisfiable.

(D): Satisfiable if every disjunction is satisfiable.

→ (A): No, other parts of the global conjunction may be false under any one given interpretation.

→ (B): Yes: The CNF is a conjunction of valid formulas, so is valid itself. (Compare the CNF transformation of the example formula on slide 26).

→ (C): No since we need *all* disjuncts to be satisfied together.

→ (D): No since we need all disjuncts to be satisfied together *by the same interpretation*.

Clausal Form

$\rightarrow$ For the remainder of this chapter, we assume that the input is a set Δ of clauses: (The same will be assumed in **Chapter 8**)

Terminology and Notation

- A literal l is an atom or the negation thereof (e.g., $P, \neg Q$); the negation of a literal is denoted $\bar{l}$ (e.g., $\overline{\neg Q} = Q$).
- A clause C is a disjunction of literals. We identify C with the set of its literals (e.g., $P \vee \neg Q$ becomes $\{P, \neg Q\}$).
- We identify a CNF formula ψ with the set Δ of its clauses (e.g., $(P \vee \neg Q) \wedge R$ becomes $\{\{P, \neg Q\}, \{R\}\}$).
- The empty clause is denoted $\Box$.

$\rightarrow$ An interpretation I satisfies a clause C iff there exists $l \in C$ such that $I \models l$. I satisfies Δ iff, for all $C \in \Delta$, we have $I \models C$.

Satisfiability in the Clausal Form: Rim Cases

It's normally simple . . .

- E.g., I with $I(P) = 0, I(Q) = 0, I(R) = 0$ does not satisfy $\Delta = \{\{P, \neg Q\}, \{R\}\}$.

. . . but can be confusing in the "rim cases":

- Does there exist I so that $I \models \Box$? No, there exists no literal $l \in \Box$ that we can satisfy.
- With $\Delta = \{\Box\}$, does there exist I so that $I \models \Delta$? No, because we can't satisfy $\Box$.
- With $\Delta = \{\}$, does there exist I so that $I \models \Delta$? Yes, because I satisfies all clauses $C \in \Delta$ (trivial as there is no clause in Δ).

Deduction

Basic Concepts in Deduction

- **Inference rule**: Rule prescribing how we can infer new formulas.

  $\rightarrow$ For example, if the KB is $\{\dots, (\varphi \rightarrow \psi), \dots, \varphi, \dots\}$ then ψ can be deduced using the inference rule $\frac{\varphi, \varphi \rightarrow \psi}{\psi}$.
- **Calculus**: Set $\mathcal{R}$ of inference rules.
- **Derivation**: φ can be derived from KB using $\mathcal{R}$, KB $\vdash_{\mathcal{R}} \varphi$, if starting from KB there is a sequence of applications of rules from $\mathcal{R}$, ending in φ.
- **Soundness**: $\mathcal{R}$ is sound if all derivable formulas do follow logically: if KB $\vdash_{\mathcal{R}} \varphi$, then KB $\models \varphi$.
- **Completeness**: $\mathcal{R}$ is complete if all formulas that follow logically are derivable: if KB $\models \varphi$, then KB $\vdash_{\mathcal{R}} \varphi$.

$\rightarrow$ If $\mathcal{R}$ is sound and complete, then to check whether KB $\models \varphi$, we can check whether KB $\vdash_{\mathcal{R}} \varphi$.

The Resolution Rule

Definition (Resolution Rule). *Resolution uses the following inference rule (with exclusive union $\dot\cup$ meaning that the two sets are disjoint):*

$$\frac{C_1\dot\cup\{l\}, C_2\dot\cup\{\bar{l}\}}{C_1 \cup C_2}$$

If Δ contains parent clauses of the form $C_1\dot\cup\{l\}$ and $C_2\dot\cup\{\bar{l}\}$, the rule allows to add the resolvent clause $C_1 \cup C_2$. l and $\bar{l}$ are called the resolution literals.

Example: $\{P, \neg R\}$ resolves with $\{R, Q\}$ to $\{P, Q\}$.

Lemma. *The resolvent follows from the parent clauses.*

Proof. If $I \models C_1\dot\cup\{l\}$ and $I \models C_2\dot\cup\{\bar{l}\}$, then I must make at least one literal in $C_1 \cup C_2$ true.

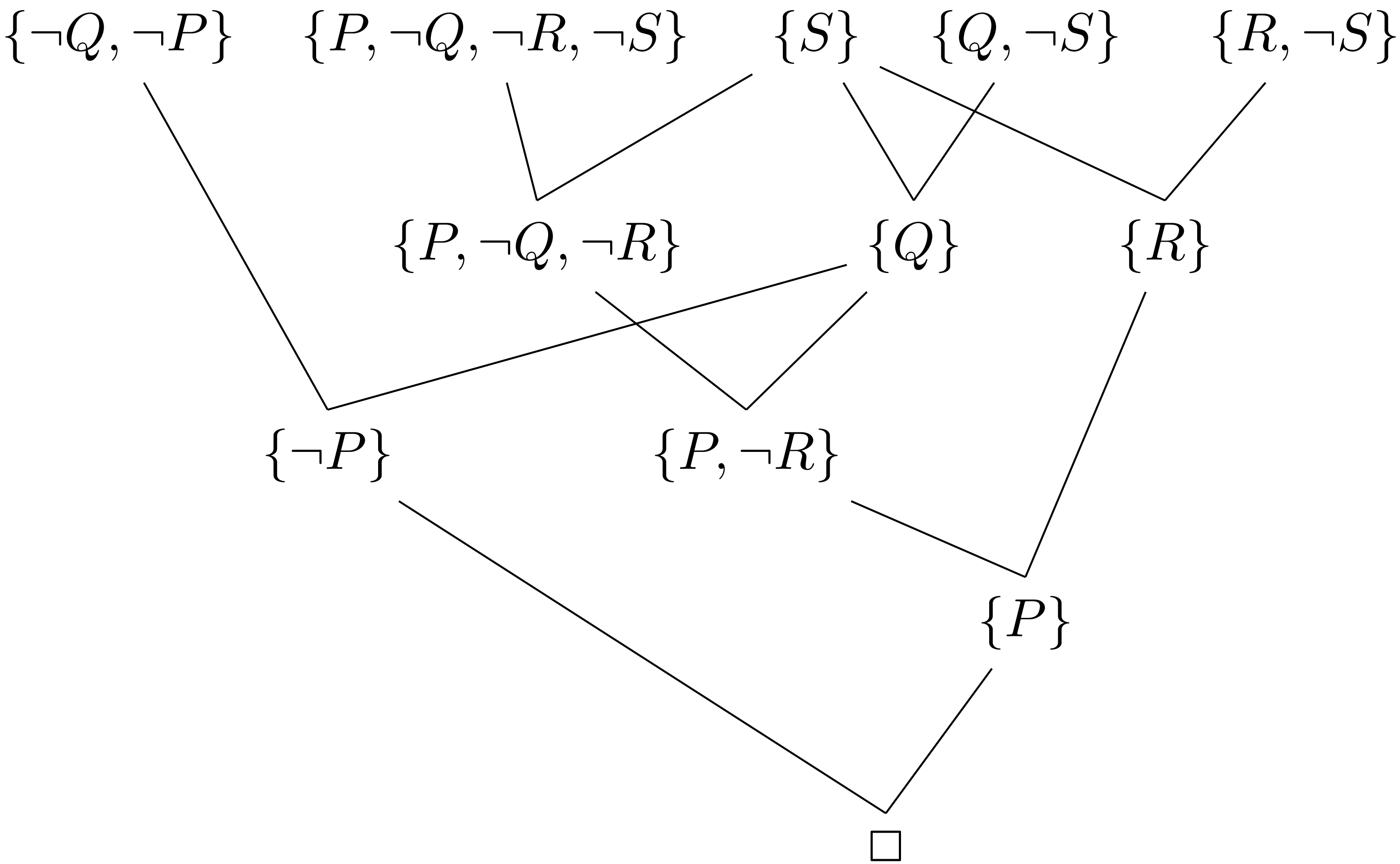
Theorem (Soundness). *If $\Delta \vdash D$, then $\Delta \models D$.* (Direct from Lemma.)

$\rightarrow$ What about the other direction? Is the resolvent *equivalent* to its parents?
No, because to satisfy the resolvent it is enough to satisfy one of C_1, C_2. E.g.: Setting $I(P) = 0$ and $I(Q) = 1$, we satisfy $\{P, Q\}$ but do not satisfy $\{P, \neg R\}$ when setting the resolution literal to $I(R) = 1$.

Using Resolution: A Simple Example

$\Delta = \{\{\neg Q, \neg P\}, \{P, \neg Q, \neg R, \neg S\}, \{Q, \neg S\}, \{S\}, \{R, \neg S\}\}$

Derive $\Box$ by applying the resolution rule.



Using Resolution: A Frequent Mistake

Question: Given clauses $C_1\dot{\cup}\{P,Q\}$ and $C_2\dot{\cup}\{\neg P,\neg Q\}$, can we resolve them to $C_1 \cup C_2$?

Answer: NO!

Observation 1: Consider $\Delta = \{\{P,Q\},\{\neg P,\neg Q\}\}$, and assume we were able to resolve as above. Then we could derive the empty clause. However, Δ is satisfiable (e.g. $P := T, Q := F$), so this deduction would be unsound.

Observation 2: The proof of the lemma on slide 32 is not valid for the hypothetical resolution of $C_1\dot{\cup}\{P,Q\}$ and $C_2\dot{\cup}\{\neg P,\neg Q\}$ to $C_1 \cup C_2$. This is due to Observation 1: An interpretation can set, e.g., $P := T, Q := F$, satisfying *both* $\{P,Q\}$ *and* $\{\neg P,\neg Q\}$ *together*, avoiding the need to satisfy either of C_1 or C_2.

Questionnaire

Question!

What are resolvents of $\{P, \neg Q, R\}$ and $\{\neg P, Q, R\}$?

(A): $\{Q, \neg Q, P, R\}$.

(B): $\{P, \neg P, R, S\}$.

(C): $\{R\}$.

(D): $\{Q, \neg Q, R\}$.

$\rightarrow$ (A): No. If we resolve on P then it disappears completely.

$\rightarrow$ (B): No. By resolving on Q we get this clause except S, and although the larger clause always is sound as well of course, we are not allowed to deduce it by the rule.

$\rightarrow$ (C): No. If we resolve on P then we get both Q and $\neg Q$ into the clause, similar if we resolve on Q.
$\rightarrow$ We can resolve on only ONE literal at a time, cf. slide 34.

$\rightarrow$ (D): Yes, this is what we get by resolving on P.

The DPLL Procedure

Call on input Δ and the empty partial interpretation I:

function DPLL(Δ, I) **returns** a partial interpretation I, or "unsatisfiable"
/* Unit Propagation (UP) Rule: */
  Δ' := a copy of Δ; $I' := I$
  while Δ' contains a unit clause $C = \{l\}$ **do**
    extend I' with the respective truth value for the proposition underlying l
    simplify Δ' /* remove false literals and true clauses */
/* Termination Test: */
  if $\Box \in \Delta'$ **then return** "unsatisfiable"
  if $\Delta' = \{\}$ **then return** I'
/* Splitting Rule: */
  select some proposition P for which I' is not defined
  $I'' := I'$ extended with one truth value for P; Δ'' := a copy of Δ'; simplify Δ''
  if $I''' :=$ DPLL$(\Delta'', I'') \neq$ "unsatisfiable" **then return** I'''
  $I'' := I'$ extended with the other truth value for P; $\Delta'' := \Delta'$; simplify Δ''
  return DPLL(Δ'', I'')

$\rightarrow$ In practice, of course one uses flags etc. instead of "copy".

DPLL: Example (Vanilla1)

$$\Delta = \{\{P, Q, \neg R\}, \{\neg P, \neg Q\}, \{R\}, \{P, \neg Q\}\}$$

1. UP Rule: $R \mapsto 1$
   $\{\{P, Q\}, \{\neg P, \neg Q\}, \{P, \neg Q\}\}$
2. Splitting Rule:

2a. $P \mapsto 0$
$\{\{Q\}, \{\neg Q\}\}$

3a. UP Rule: $Q \mapsto 1$
$\{\Box\}$

2b. $P \mapsto 1$
$\{\{\neg Q\}\}$

3b. UP Rule: $Q \mapsto 0$
$\{\}$

DPLL: Example (Vanilla2)

$$\Delta = \{\{\neg Q, \neg P\}, \{P, \neg Q, \neg R, \neg S\}, \{Q, \neg S\}, \{R, \neg S\}, \{S\}\}$$

1. UP Rule: $S \mapsto 1$
   $\{\{\neg Q, \neg P\}, \{P, \neg Q, \neg R\}, \{Q\}, \{R\}\}$
2. UP Rule: $Q \mapsto 1$
   $\{\{\neg P\}, \{P, \neg R\}, \{R\}\}$
3. UP Rule: $R \mapsto 1$
   $\{\{\neg P\}, \{P\}\}$
4. UP Rule: $P \mapsto 1$
   $\{\Box\}$

Properties of DPLL

Unsatisfiable case:

- What can we say if "unsatisfiable" is returned?

  $\rightarrow$ In this case, we know that Δ is unsatisfiable: Unit propagation is *sound*, in the sense that it does not reduce the set of solutions. ($=$ Soundness of calculus, cf. next two slides.)

Satisfiable case:

- What can we say when a partial interpretation I is returned?

  $\rightarrow$ Any extension of I to a complete interpretation satisfies Δ. (By construction, I suffices to satisfy all clauses.)

DPLL is an example of a successful algorithmic pattern: Search $+$ Inference

- DPLL $\approx$ Search $=$ Backtracking, with Inference() $=$ unit propagation.
- Unit propagation is sound: It does not reduce the set of solutions. (Also: $=$ Soundness of calculus, cf. next slide.)

UP = Unit Resolution

The Unit Propagation (UP) Rule ...

while Δ' contains a unit clause $\{l\}$ **do**
  extend I' with the respective truth value for the proposition underlying l
  simplify Δ' /* remove false literals */

... corresponds to a calculus:

Definition (Unit Resolution). *Unit Resolution is the calculus consisting of the following inference rule:*

$$\frac{C\dot{\cup}\{\bar{l}\}, \{l\}}{C}$$

That is, if Δ contains parent clauses of the form $C\dot{\cup}\{\bar{l}\}$ and $\{l\}$, the rule allows to add the resolvent clause C.

$\rightarrow$ Unit propagation = Resolution restricted to the case where one of the parent clauses is unit.

UP/Unit Resolution: Soundness/Completeness

Soundness:

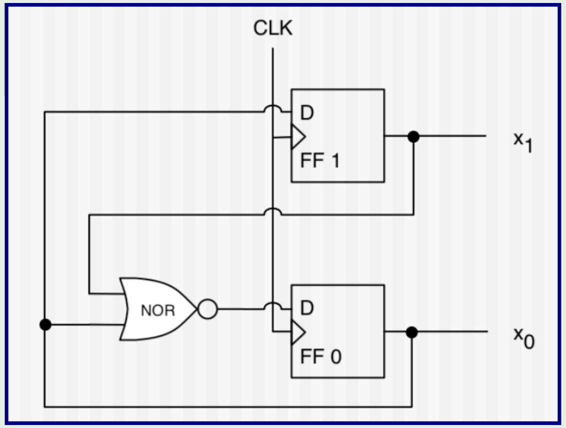
- Need to show: *If Δ' can be derived from Δ by UP, then $\Delta \models \Delta'$.*
- Yes, because any derivation made by unit resolution can also be made by (full) resolution, which we already know has this property.
- (Intuitively: if Δ' contains the unit clause $\{l\}$, then l must be made true so $C\dot{\cup}\{\bar{l}\}$ implies C.)

Completeness:

- Need to show: *If $\Delta \models \Delta'$, then Δ' can be derived from Δ by UP.*
- No. UP makes only limited inferences, as long as there are unit clauses. It does not guarantee to infer everything that can be inferred.
- Example: $\{\{P, Q\}, \{P, \neg Q\}, \{\neg P, Q\}, \{\neg P, \neg Q\}\}$ is unsatisfiable but UP cannot derive the empty clause $\Box$.

Questionnaire

Example



- Counter, repeatedly from $c = 0$ to $c = 2$.
- **To Verify:** If $c < 3$ in current clock cycle, then $c < 3$ in next clock cycle.
- $\Delta = \{\{\neg x_1', x_0\}, \{x_1', \neg x_0\}, \{x_0', x_1, x_0\}, \{\neg x_o', \neg x_1\}, \{\neg x_0', \neg x_0\}, \{\neg x_1, \neg x_0\}, \{x_1'\}, \{x_0'\}\}$

Question!

How many recursive calls to DPLL are made on Δ?

(A): 0 (B): 1

(C): 4 (D): 11

$\rightarrow$ The correct answer is (B): UP derives the empty clause (via $\{x_1'\}$, $\{\neg x_1', x_0\}$, $\{\neg x_0', \neg x_0\}$, $\{x_0'\}$) in the first recursive call, so exactly 1 search node is generated.

Summary

- SAT: Is a propositional logic formula ϕ satisfiable?
  - Hard problem in general (NP-hard)
  - Many applications
- Propositional logic formulas are built from atomic propositions, with the connectives "and, or, not".
- Every propositional formula can be brought into conjunctive normal form (CNF), which can be identified with a set of clauses.
- Resolution is a deduction procedure based on trying to derive the empty clause. It is refutation-complete, and can be used to prove $\mathrm{KB} \models \varphi$ by showing that $\mathrm{KB} \cup \{\neg\varphi\}$ is unsatisfiable.
- SAT solvers decide satisfiability of CNF formulas. This can be used for deduction, and is highly successful as a general problem solving technique (e.g., in Verification).
- DPLL = backtracking with inference performed by unit propagation (UP), which iteratively instantiates unit clauses and simplifies the formula.

Further Reading

The main material for the course are the post-handouts. If you are interested on more detailed overview of the topic, you can check these books:

- The Art of Computer Programming by Donald E. Knuth, Vol 4. Section 7.2.2.2
- Handbook of Satisfiability, Hans van Maaren, Armin Biere, Toby Walsh.