

CS156: The Calculus of Computation

Zohar Manna

Autumn 2008

Lecturer:

Zohar Manna (manna@cs.stanford.edu)

Office Hours: MW 12:30-1:00 at Gates 481

TAs:

Boyu Wang (wangboyu@stanford.edu)

Office Hours: MTu 3:00-5:00 at Durand 1st floor lounge

Greg Goldgof (ggoldgof@stanford.edu)

Office Hours: F 3:00-5:00 at Durand 1st floor lounge

Calculus of Computation?

*It is reasonable to hope that the relationship between **computation** and **mathematical logic** will be as fruitful in the next century as that between **analysis** and **physics** in the last. The development of this relationship demands a concern for both applications and mathematical elegance.*

John McCarthy

A Basis for a Mathematical Theory of Computation, 1963

Grading

- ▶ Homeworks (60%)
 - ▶ weekly (totally 8)
 - ▶ no late assignments
 - ▶ no collaboration
- ▶ Final Exam (40%)
 - ▶ open book and notes
 - ▶ date: Monday, December 8th, 8:30-11:30 a.m.

Coverage

- ▶ Skip * sections
- ▶ Skip Chapter 6 and 12 of the book
- ▶ Skip complexity remarks

Website

- ▶ `cs156.stanford.edu`

Assignment #1 (due Mon, Sept 29)

- ▶ 1.1 e, f [10 points each]
- ▶ 1.2 s, x [10 points each]
- ▶ 1.3 (note typo: the last $\vee$ should be a $\wedge$) [30 points]
- ▶ 1.5 d [30 points]

THE CALCULUS OF COMPUTATION: Decision Procedures with Applications to Verification

by
Aaron Bradley
Zohar Manna

Springer 2007

There are two copies in CS-Math Library and you could also use `socrates.stanford.edu` to read the book according to its policy.

Aaron R. Bradley
Zohar Manna

Decision Procedures with Applications to Verification

Topics: Overview

1. First-Order logic
2. Specification and verification
3. Satisfiability decision procedures

Part I: Foundations

1. Propositional Logic
2. First-Order Logic
3. First-Order Theories
4. Induction
5. Program Correctness: Mechanics
Inductive assertion method, Ranking function method

Part II: Decision Procedures

- 7. Quantified Linear Arithmetic
Quantifier elimination for integers and rationals
- 8. Quantifier-Free Linear Arithmetic
Linear programming for rationals
- 9. Quantifier-Free Equality and Data Structures
- 10. Combining Decision Procedures
Nelson-Oppen combination method
- 11. Arrays
More than quantifier-free fragment

CS156: The Calculus of Computation

Zohar Manna

Autumn 2008

Motivation

Motivation I

Decision Procedures are algorithms to decide formulae.

These formulae can arise

- ▶ in software verification.
- ▶ in hardware verification

Consider the following program:

```
for
  @  $\ell \leq i \leq u \wedge (rv \leftrightarrow \exists j. \ell \leq j < i \wedge a[j] = e)$ 
  (int  $i := \ell$ ;  $i \leq u$ ;  $i := i + 1$ ) {
    if ( $a[i] = e$ )  $rv := \text{true}$ ;
  }
```

How can we decide whether the formula is a loop invariant?

Motivation II

Prove:

assume $\ell \leq i \leq u \wedge (rv \leftrightarrow \exists j. \ell \leq j < i \wedge a[j] = e)$

assume $i \leq u$

assume $a[i] = e$

$rv := \text{true};$

$i := i + 1$

@ $\ell \leq i \leq u \wedge (rv \leftrightarrow \exists j. \ell \leq j < i \wedge a[j] = e)$

Motivation III

assume $\ell \leq i \leq u \wedge (rv \leftrightarrow \exists j. \ell \leq j < i \wedge a[j] = e)$

assume $i \leq u$

assume $a[i] \neq e$

$i := i + 1$

@ $\ell \leq i \leq u \wedge (rv \leftrightarrow \exists j. \ell \leq j < i \wedge a[j] = e)$

A Hoare triple $\{P\} S \{Q\}$ holds, iff

$$P \rightarrow wp(S, Q)$$

(wp denotes “weakest precondition”)

Motivation IV

For assignments wp is computed by substitution:

$\text{assume } \ell \leq i \leq u \wedge (rv \leftrightarrow \exists j. \ell \leq j < i \wedge a[j] = e)$

$\text{assume } i \leq u$

$\text{assume } a[i] = e$

$rv := \text{true};$

$i := i + 1$

$@ \ell \leq i \leq u \wedge (rv \leftrightarrow \exists j. \ell \leq j < i \wedge a[j] = e)$

Substituting $\top$ for rv and $i + 1$ for i , the postcondition (denoted by the $@$ symbol) holds if and only if:

$$\begin{aligned} & \ell \leq i \leq u \wedge (rv \leftrightarrow \exists j. \ell \leq j < i \wedge a[j] = e) \wedge i \leq u \wedge a[i] = e \\ \rightarrow & \ell \leq i + 1 \leq u \wedge (\top \leftrightarrow \exists j. \ell \leq j < i + 1 \wedge a[j] = e) \end{aligned}$$

Motivation V

We need an algorithm that decides whether this formula holds. If the formula does not hold, the algorithm should give a counterexample; e.g.,

$$\ell = 0, i = 1, u = 1, rv = \textit{false}, a[0] = 0, a[1] = 1, e = 1.$$

We will discuss such algorithms in later lectures.

CS156: The Calculus of Computation

Zohar Manna

Autumn 2008

Chapter 1: Propositional Logic (PL)

Propositional Logic (PL)

PL Syntax

<u>Atom</u>	<u>truth symbols</u> $\top$ (“true”) and $\perp$ (“false”) <u>propositional variables</u> $P, Q, R, P_1, Q_1, R_1, \dots$	
<u>Literal</u>	atom α or its negation $\neg\alpha$	
<u>Formula</u>	literal or application of a <u>logical connective</u> to formulae F, F_1, F_2	
	$\neg F$	“not” (negation)
	$F_1 \wedge F_2$	“and” (conjunction)
	$F_1 \vee F_2$	“or” (disjunction)
	$F_1 \rightarrow F_2$	“implies” (implication)
	$F_1 \leftrightarrow F_2$	“if and only if” (iff)

Example:

formula $F : (P \wedge Q) \rightarrow (\top \vee \neg Q)$

atoms: $P, Q, \top$

literals: $P, Q, \top, \neg Q$

subformulae: $P, Q, \top, \neg Q, P \wedge Q, \top \vee \neg Q, F$

abbreviation

$$F : P \wedge Q \rightarrow \top \vee \neg Q$$

PL Semantics (meaning of PL)

Formula F + Interpretation I = Truth value
(true, false)

Interpretation

$$I : \{P \mapsto \text{true}, Q \mapsto \text{false}, \dots\}$$

Evaluation of F under I :

F	$\neg F$	where 0 corresponds to value false 1 true
0	1	
1	0	

F_1	F_2	$F_1 \wedge F_2$	$F_1 \vee F_2$	$F_1 \rightarrow F_2$	$F_1 \leftrightarrow F_2$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

Example:

$$F : P \wedge Q \rightarrow P \vee \neg Q$$

$$I : \{P \mapsto \text{true}, Q \mapsto \text{false}\} \quad \text{i.e., } I[P] = \text{true}, I[Q] = \text{false}$$

P	Q	$\neg Q$	$P \wedge Q$	$P \vee \neg Q$	F
1	0	1	0	1	1

1 = true

0 = false

F evaluates to true under I ; i.e., $I[F] = \text{true}$.

Inductive Definition of PL's Semantics

$I \models F$ if F evaluates to true under I
 $I \not\models F$ false

Base Case:

$I \models \top$ $I \not\models \perp$

$I \models P$ iff $I[P] = \text{true}$; i.e., P is true under I

$I \not\models P$ iff $I[P] = \text{false}$

Inductive Case:

$I \models \neg F$ iff $I \not\models F$

$I \models F_1 \wedge F_2$ iff $I \models F_1$ and $I \models F_2$

$I \models F_1 \vee F_2$ iff $I \models F_1$ or $I \models F_2$ (or both)

$I \models F_1 \rightarrow F_2$ iff $I \models F_1$ implies $I \models F_2$

$I \models F_1 \leftrightarrow F_2$ iff, $I \models F_1$ and $I \models F_2$,
or $I \not\models F_1$ and $I \not\models F_2$

Note:

$I \models F_1 \rightarrow F_2$ iff $I \not\models F_1$ or $I \models F_2$.

$I \not\models F_1 \rightarrow F_2$ iff $I \models F_1$ and $I \not\models F_2$.

$I \not\models F_1 \vee F_2$ iff $I \not\models F_1$ and $I \not\models F_2$.

Example of Inductive Reasoning:

$$F : P \wedge Q \rightarrow P \vee \neg Q$$

$$I : \{P \mapsto \text{true}, Q \mapsto \text{false}\}$$

1. $I \models P$ since $I[P] = \text{true}$
2. $I \not\models Q$ since $I[Q] = \text{false}$
3. $I \models \neg Q$ by 2 and $\neg$
4. $I \not\models P \wedge Q$ by 2 and $\wedge$
5. $I \models P \vee \neg Q$ by 1 and $\vee$
6. $I \models F$ by 4 and $\rightarrow$ Why?

Thus, F is true under I .

Note: steps 1, 3, and 5 are nonessential.

Satisfiability and Validity

F satisfiable iff there exists an interpretation I such that $I \models F$.

F valid iff for all interpretations I , $I \models F$.

F is valid iff $\neg F$ is unsatisfiable
--

Goal: devise an algorithm to decide validity or unsatisfiability of formula F .

Method 1: Truth Tables

Example $F : P \wedge Q \rightarrow P \vee \neg Q$

P	Q	$P \wedge Q$	$\neg Q$	$P \vee \neg Q$	F
0	0	0	1	1	1
0	1	0	0	0	1
1	0	0	1	1	1
1	1	1	0	1	1

Thus F is valid.

Example $F : P \vee Q \rightarrow P \wedge Q$

P	Q	$P \vee Q$	$P \wedge Q$	F
0	0	0	0	1
0	1	1	0	0
1	0	1	0	0
1	1	1	1	1

← satisfying /

← falsifying /

Thus F is satisfiable, but invalid.

Method 2: Semantic Argument

- ▶ Assume F is not valid and I a falsifying interpretation:
 $I \not\models F$
- ▶ Apply proof rules.
- ▶ If no contradiction reached and no more rules applicable,
 F is invalid.
- ▶ If in every branch of proof a contradiction reached,
 F is valid.

Proof Rules for Semantic Arguments I

$$\frac{I \models \neg F}{I \not\models F}$$

$$\frac{I \not\models \neg F}{I \models F}$$

$$\frac{I \models F \wedge G}{\begin{array}{l} I \models F \\ I \models G \end{array}} \leftarrow \text{and}$$

$$\frac{I \not\models F \wedge G}{\begin{array}{l} I \not\models F \quad | \quad I \not\models G \end{array}} \leftarrow \text{or}$$

$$\frac{I \models F \vee G}{I \models F \quad | \quad I \models G}$$

$$\frac{I \not\models F \vee G}{\begin{array}{l} I \not\models F \\ I \not\models G \end{array}}$$

Proof Rules for Semantic Arguments II

$$\frac{I \models F \rightarrow G}{I \not\models F \mid I \models G}$$

$$\frac{I \not\models F \rightarrow G}{I \models F}$$
$$I \not\models G$$

$$\frac{I \models F \leftrightarrow G}{I \models F \wedge G \mid I \not\models F \vee G}$$

$$\frac{I \not\models F \leftrightarrow G}{I \models F \wedge \neg G \mid I \models \neg F \wedge G}$$

$$I \models F$$

$$I \not\models F$$

$$\frac{}{I \models \perp}$$

Example: Prove

$$F : P \wedge Q \rightarrow P \vee \neg Q \quad \text{is valid.}$$

Let's assume that F is not valid and that I is a falsifying interpretation.

- | | | | |
|----|-----------------|--|---------------------------|
| 1. | $I \not\models$ | $P \wedge Q \rightarrow P \vee \neg Q$ | assumption |
| 2. | $I \models$ | $P \wedge Q$ | 1 and $\rightarrow$ |
| 3. | $I \not\models$ | $P \vee \neg Q$ | 1 and $\rightarrow$ |
| 4. | $I \models$ | P | 2 and $\wedge$ |
| 5. | $I \not\models$ | P | 3 and $\vee$ |
| 6. | $I \models$ | $\perp$ | 4 and 5 are contradictory |

Thus F is valid.

Example: Prove

$F : (P \rightarrow Q) \wedge (Q \rightarrow R) \rightarrow (P \rightarrow R)$ is valid.

Let's assume that F is not valid.

- | | | |
|----|--|---------------------|
| 1. | $I \not\models F$ | assumption |
| 2. | $I \models (P \rightarrow Q) \wedge (Q \rightarrow R)$ | 1 and $\rightarrow$ |
| 3. | $I \not\models P \rightarrow R$ | 1 and $\rightarrow$ |
| 4. | $I \models P$ | 3 and $\rightarrow$ |
| 5. | $I \not\models R$ | 3 and $\rightarrow$ |
| 6. | $I \models P \rightarrow Q$ | 2 and $\wedge$ |
| 7. | $I \models Q \rightarrow R$ | 2 and $\wedge$ |

- | | | |
|-------|-----------------------------|----------------------------------|
| 6. | $I \models P \rightarrow Q$ | 2 and $\wedge$ |
| 7. | $I \models Q \rightarrow R$ | 2 and $\wedge$ |
| 8a. | $I \not\models P$ | 6 and $\rightarrow$ (case a) |
| 9a. | $I \models \perp$ | 4 and 8 |
| 8b. | $I \models Q$ | 6 and $\rightarrow$ (case b) |
| 9ba. | $I \not\models Q$ | 7 and $\rightarrow$ (subcase ba) |
| 10ba. | $I \models \perp$ | 8b and 9ba |
| 9bb. | $I \models R$ | 7 and $\rightarrow$ (subcase bb) |
| 10bb. | $I \models \perp$ | 5 and 9bb |
| 9b. | $I \models \perp$ | 10ba and 10bb |
| 8. | $I \models \perp$ | 9a and 9b |

Our assumption is contradictory in all cases, so F is valid.

Example 3: Is

$$F : P \vee Q \rightarrow P \wedge Q$$

valid? Assume F is not valid:

- | | | |
|------|---|------------------------|
| 1. | $I \not\models P \vee Q \rightarrow P \wedge Q$ | assumption |
| 2. | $I \models P \vee Q$ | 1 and $\rightarrow$ |
| 3. | $I \not\models P \wedge Q$ | 1 and $\rightarrow$ |
| 4a. | $I \models P$ | 2, $\vee$ (case a) |
| 5aa. | $I \not\models P$ | 3, $\vee$ (subcase aa) |
| 6aa. | $I \models \perp$ | 4a, 5aa |
| 5ab. | $I \not\models Q$ | 3, $\vee$ (subcase ab) |
| 6ab. | ? | |
| 5a. | ? | |

- 4b. $I \models Q$ 2, $\vee$ (case b)
- 5ba. $I \not\models P$ 3, $\vee$ (subcase ba)
- 6ba. ?
- 5bb. $I \not\models Q$ 3, $\vee$ (subcase bb)
- 6bb. $I \models \perp$ 4b, 5bb
- 5b. ?
- 5. ?

We cannot derive a contradiction in both cases (4a and 4b), so we cannot prove that F is valid. To demonstrate that F is not valid, however, we must find a falsifying interpretation (here are two):

$$I_1 : \{P \mapsto \text{true}, Q \mapsto \text{false}\} \quad I_2 : \{Q \mapsto \text{true}, P \mapsto \text{false}\}$$

Note: we have to derive a contradiction in all cases for F to be valid!

Equivalence

F_1 and F_2 are equivalent ($F_1 \Leftrightarrow F_2$)

iff for all interpretations I , $I \models F_1 \leftrightarrow F_2$

To prove $F_1 \Leftrightarrow F_2$, show $F_1 \leftrightarrow F_2$ is valid, that is, both $F_1 \rightarrow F_2$ and $F_2 \rightarrow F_1$ are valid.

F_1 entails F_2 ($F_1 \Rightarrow F_2$)

iff for all interpretations I , $I \models F_1 \rightarrow F_2$

Note: $F_1 \Leftrightarrow F_2$ and $F_1 \Rightarrow F_2$ are not formulae!!

Example: Show

$$P \rightarrow Q \Leftrightarrow \neg P \vee Q$$

i.e.

$$F : (P \rightarrow Q) \leftrightarrow (\neg P \vee Q) \text{ is valid.}$$

Assume F is not valid, then we have two cases:

Case a: $I \not\models \neg P \vee Q$,

$$I \models P \rightarrow Q$$

Case b: $I \models \neg P \vee Q$,

$$I \not\models P \rightarrow Q$$

Derive contradictions in both cases.

Normal Forms

1. Negation Normal Form (NNF)

$\neg, \wedge, \vee$ are the only boolean connectives allowed.

Negations may occur only in literals of the form $\neg P$.

To transform F into equivalent F' in NNF, apply the following template equivalences recursively (and left-to-right):

$$\begin{aligned}\neg\neg F_1 &\Leftrightarrow F_1 & \neg\top &\Leftrightarrow \perp & \neg\perp &\Leftrightarrow \top \\ \neg(F_1 \wedge F_2) &\Leftrightarrow \neg F_1 \vee \neg F_2 \\ \neg(F_1 \vee F_2) &\Leftrightarrow \neg F_1 \wedge \neg F_2 & \left. \vphantom{\begin{aligned} \neg(F_1 \wedge F_2) &\Leftrightarrow \neg F_1 \vee \neg F_2 \\ \neg(F_1 \vee F_2) &\Leftrightarrow \neg F_1 \wedge \neg F_2 \end{aligned}} \right\} & \text{De Morgan's Law} \\ F_1 \rightarrow F_2 &\Leftrightarrow \neg F_1 \vee F_2 \\ F_1 \leftrightarrow F_2 &\Leftrightarrow (F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1)\end{aligned}$$

“Complete” syntactic restriction: every F has an equivalent F' in NNF.

Example: Convert

$$F : \neg(P \rightarrow \neg(P \wedge Q))$$

to NNF.

$$F' : \neg(\neg P \vee \neg(P \wedge Q)) \quad \rightarrow$$

$$F'' : \neg\neg P \wedge \neg\neg(P \wedge Q) \quad \text{De Morgan's Law}$$

$$F''' : P \wedge P \wedge Q \quad \neg\neg$$

F''' is equivalent to F ($F''' \Leftrightarrow F$) and is in NNF.

2. Disjunctive Normal Form (DNF)

Disjunction of conjunctions of literals

$$\bigvee_i \bigwedge_j \ell_{i,j} \quad \text{for literals } \ell_{i,j}$$

To convert F into equivalent F' in DNF,
transform F into NNF and then
use the following template equivalences (left-to-right):

$$\left. \begin{array}{l} (F_1 \vee F_2) \wedge F_3 \quad \Leftrightarrow \quad (F_1 \wedge F_3) \vee (F_2 \wedge F_3) \\ F_1 \wedge (F_2 \vee F_3) \quad \Leftrightarrow \quad (F_1 \wedge F_2) \vee (F_1 \wedge F_3) \end{array} \right\} dist$$

Note: formulae can grow exponentially as the distributivity laws are applied.

Example: Convert

$$F : (Q_1 \vee \neg\neg Q_2) \wedge (\neg R_1 \rightarrow R_2)$$

into equivalent DNF

$$F' : (Q_1 \vee Q_2) \wedge (R_1 \vee R_2) \quad \text{in NNF}$$

$$F'' : (Q_1 \wedge (R_1 \vee R_2)) \vee (Q_2 \wedge (R_1 \vee R_2)) \quad \text{dist}$$

$$F''' : (Q_1 \wedge R_1) \vee (Q_1 \wedge R_2) \vee (Q_2 \wedge R_1) \vee (Q_2 \wedge R_2) \quad \text{dist}$$

F''' is equivalent to F ($F''' \Leftrightarrow F$) and is in DNF.

3. Conjunctive Normal Form (CNF)

Conjunction of disjunctions of literals

$$\bigwedge_i \bigvee_j \ell_{i,j} \quad \text{for literals } \ell_{i,j}$$

To convert F into equivalent F' in CNF,
transform F into NNF and then
use the following template equivalences (left-to-right):

$$\begin{aligned}(F_1 \wedge F_2) \vee F_3 &\Leftrightarrow (F_1 \vee F_3) \wedge (F_2 \vee F_3) \\ F_1 \vee (F_2 \wedge F_3) &\Leftrightarrow (F_1 \vee F_2) \wedge (F_1 \vee F_3)\end{aligned}$$

A disjunction of literals is called a clause.

Example: Convert

$$F : P \leftrightarrow (Q \rightarrow R)$$

to an equivalent formula F' in CNF.

First get rid of $\leftrightarrow$:

$$F_1 : (P \rightarrow (Q \rightarrow R)) \wedge ((Q \rightarrow R) \rightarrow P)$$

Now replace $\rightarrow$ with $\vee$:

$$F_2 : (\neg P \vee (\neg Q \vee R)) \wedge (\neg(\neg Q \vee R) \vee P)$$

Drop unnecessary parentheses and apply De Morgan's Law:

$$F_3 : (\neg P \vee \neg Q \vee R) \wedge ((\neg\neg Q \wedge \neg R) \vee P)$$

Simplify double negation (now in NNF):

$$F_4 : (\neg P \vee \neg Q \vee R) \wedge ((Q \wedge \neg R) \vee P)$$

Distribute disjunction over conjunction (now in CNF):

$$F' : (\neg P \vee \neg Q \vee R) \wedge (Q \vee P) \wedge (\neg R \vee P)$$

Equisatisfiability

Definition

F and F' are *equisatisfiable*, iff

F is satisfiable if and only if F' is satisfiable

Every formula is equisatisfiable to either $\top$ or $\perp$.

Goal: Decide satisfiability of PL formula F

Step 1: Convert F to equisatisfiable formula F' in CNF

Step 2: Decide satisfiability of formula F' in CNF

Step 1: Convert F to equisatisfiable formula F' in CNF I

There is an *efficient conversion* of F to F' where

- ▶ F' is in CNF and
- ▶ F and F' are equisatisfiable

Note: efficient means polynomial in the size of F .

Basic Idea:

- ▶ Introduce a new variable P_G for every subformula G of F , unless G is already an atom.

Step 1: Convert F to equisatisfiable formula F' in CNF II

- For each subformula

$$G : G_1 \circ G_2,$$

produce a small formula

$$P_G \leftrightarrow P_{G_1} \circ P_{G_2}.$$

Here $\circ$ denotes an arbitrary connective ($\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$); if the connective is $\neg$, G_1 should be ignored.

Step 1: Convert F to equisatisfiable formula F' in CNF III

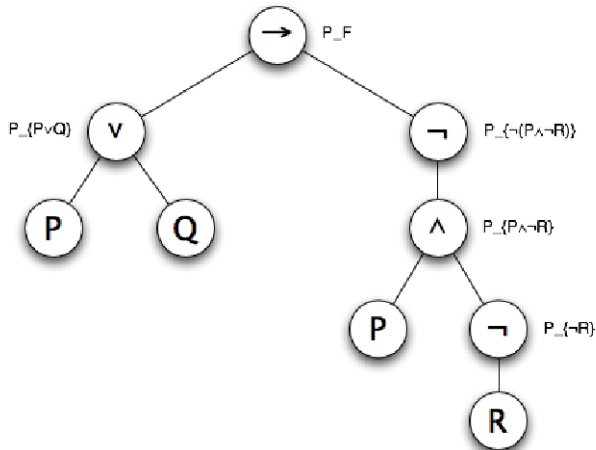


Figure: Parse tree for $F : P \vee Q \rightarrow \neg(P \wedge \neg R)$

Step 1: Convert F to equisatisfiable formula F' in CNF IV

- Convert each of these (small) formulae separately to an equivalent CNF formula

$$\text{CNF}(P_G \leftrightarrow P_{G_1} \circ P_{G_2}) .$$

Let S_F be the set of all non-atom subformulae G of F (including F itself). The formula

$$P_F \wedge \bigwedge_{G \in S_F} \text{CNF}(P_G \leftrightarrow P_{G_1} \circ P_{G_2})$$

is equisatisfiable to F . (Why?)

The number of subformulae is linear in the size of F .
The time to convert one small formula is constant!

Example: CNF I

Convert

$$F : P \vee Q \rightarrow P \wedge \neg R$$

to an equisatisfiable formula in CNF.

Introduce new variables: P_F , $P_{P \vee Q}$, $P_{P \wedge \neg R}$, $P_{\neg R}$.

Create new formulae and convert them to equivalent formulae in CNF separately:

► $F_1 = \text{CNF}(P_F \leftrightarrow (P_{P \vee Q} \rightarrow P_{P \wedge \neg R})):$

$$(\neg P_F \vee \neg P_{P \vee Q} \vee P_{P \wedge \neg R}) \wedge (P_F \vee P_{P \vee Q}) \wedge (P_F \vee \neg P_{P \wedge \neg R})$$

► $F_2 = \text{CNF}(P_{P \vee Q} \leftrightarrow P \vee Q):$

$$(\neg P_{P \vee Q} \vee P \vee Q) \wedge (P_{P \vee Q} \vee \neg P) \wedge (P_{P \vee Q} \vee \neg Q)$$

Example: CNF II

► $F_3 = \text{CNF}(P_{P \wedge \neg R} \leftrightarrow P \wedge P_{\neg R})$:

$$(\neg P_{P \wedge \neg R} \vee P) \wedge (\neg P_{P \wedge \neg R} \vee P_{\neg R}) \wedge (P_{P \wedge \neg R} \vee \neg P \vee \neg P_{\neg R})$$

► $F_4 = \text{CNF}(P_{\neg R} \leftrightarrow \neg R)$:

$$(\neg P_{\neg R} \vee \neg R) \wedge (P_{\neg R} \vee R)$$

$P_F \wedge F_1 \wedge F_2 \wedge F_3 \wedge F_4$ is in CNF and equisatisfiable to F .

Step 2: Decide the satisfiability of PL formula F' in CNF

Boolean Constraint Propagation (BCP)

If a clause contains one literal ℓ ,

Set ℓ to $\top$:

Remove all clauses containing ℓ :

Remove $\neg\ell$ in all clauses:

based on the unit resolution

$$\frac{\ell \quad \neg\ell \vee C \leftarrow \text{clause}}{C}$$

$$\begin{aligned} & \dots \wedge \overset{\top}{\ell} \wedge \dots \\ & \dots \wedge (\dots \vee \ell \vee \dots) \wedge \dots \\ & \dots \wedge (\dots \vee \cancel{\neg\ell} \vee \dots) \wedge \dots \end{aligned}$$

Pure Literal Propagation (PLP)

If P occurs only positive (without negation), set it to $\top$.

If P occurs only negative set it to $\perp$.

Then do the simplifications as in Boolean Constraint Propagation

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

Decides the satisfiability of PL formulae in CNF

Decision Procedure DPLL: Given F in CNF

```
let rec DPLL  $F$  =  
  let  $F' = \text{BCP } F$  in  
  let  $F'' = \text{PLP } F'$  in  
  if  $F'' = \top$  then true  
  else if  $F'' = \perp$  then false  
  else  
    let  $P = \text{CHOOSE vars}(F'')$  in  
    ( $\text{DPLL } F''\{P \mapsto \top\}$ )  $\vee$  ( $\text{DPLL } F''\{P \mapsto \perp\}$ )
```

Simplification

Simplify according to the template equivalences (left-to-right)
[exercise 1.2]

$$\neg \perp \Leftrightarrow \top$$

$$\neg \top \Leftrightarrow \perp$$

$$\neg \neg F \Leftrightarrow F$$

$$F \wedge \top \Leftrightarrow F$$

$$F \wedge \perp \Leftrightarrow \perp$$

...

$$F \vee \top \Leftrightarrow \top$$

$$F \vee \perp \Leftrightarrow F$$

...

Example I

Consider

$$F : (\neg P \vee Q \vee R) \wedge (\neg Q \vee R) \wedge (\neg Q \vee \neg R) \wedge (P \vee \neg Q \vee \neg R).$$

Branching on Q

On the first branch, we have

$$F\{Q \mapsto \top\} : (R) \wedge (\neg R) \wedge (P \vee \neg R).$$

By unit resolution,

$$\frac{R \quad (\neg R)}{\perp},$$

so $F\{Q \mapsto \top\} = \perp \Rightarrow \text{false}.$

Example II

Recall

$$F : (\neg P \vee Q \vee R) \wedge (\neg Q \vee R) \wedge (\neg Q \vee \neg R) \wedge (P \vee \neg Q \vee \neg R).$$

On the other branch, we have

$$F\{Q \mapsto \perp\} : (\neg P \vee R).$$

Furthermore, by PLP,

$$F\{Q \mapsto \perp, R \mapsto \top, P \mapsto \perp\} = \top \Rightarrow \text{true}$$

Thus F is satisfiable with satisfying interpretation

$$I : \{P \mapsto \text{false}, Q \mapsto \text{false}, R \mapsto \text{true}\}.$$

Example

$$F : (\neg P \vee Q \vee R) \wedge (\neg Q \vee R) \wedge (\neg Q \vee \neg R) \wedge (P \vee \neg Q \vee \neg R)$$

