Resume : Formal Methods

January 29, 2018

Chapter 1

Hoare Logic

1.1 Hoare Triple

Logical formulas can be used to express information about program states. We called $\{P\}S\{Q\}$ a "Hoare Triple" :

- A precondition P what can be assumed to be true before executing a sequence of statements S.
- A postcondition Q states what will be true after the execution of S.

We write $\{P\}S\{Q\}$ to indicate : if P is true, then executing S will make Q true.
We also use the notation x' to denote the variable x after the execution of S. For example

$$\{true\}\ x = x + 1;\ \{x' = x + 1\}$$

is a valid Hoare Triple in which x' is the value of x after the execution of S. We can write any predicate in between any two lines of code (an assertion), we assume that such a predicate is the postcondition of the previous line and the precondition of the following line.

1.2 Correctness of Hoare Triple

We translate our program S into a formula ϕ_S and then, we check

$$P \wedge \phi_S \rightarrow Q$$

or, equivalently

$$\phi_S \rightarrow (P \rightarrow Q)$$

So, for example, we turn the following Hoare Triple

$$\{x \neq 0\}\ x = 1/x;\ x = 1/x;\ \{x' = x\}$$

into the following formula (also using the primed notation) :

$$\underbrace{x \neq 0}_{P} \wedge \underbrace{x'' = 1/x \wedge x' = 1/x''}_{S} \rightarrow \underbrace{x' = x}_{Q}$$

Which is true by elementary arithmetic.

1.2.1 If Clauses

We turn

$$\{P\}\ if(condition)\ \{prog1\}\ else\ \{prog2\};\ \{Q\}$$

into

$$\{P \wedge condition\}\ prog1\ \{Q\}$$

and

$$\{P \wedge \neg condition\}\ prog2\ \{Q\}$$

Both of those Hoare triple must be true for the if clause to be correct.

1.2.2 Loops Clauses

In order to check the total correctness of a program with loops, we have to check the partial correctness and the termination of the program. Such a program is in the following form :

$$\{P\}\ initialisation;\ while\ (condition)\ \{loop\ body\};\ \{Q\}$$

Partial correctness

A loop invariant is a logical formula that is true

- before the loop,
- before each execution of the loop body,
- after each execution of the loop body,
- after the loop.

Then, from

$$\{P\}\ initialisation;\ while\ (condition)\ \{loop\ body\};\ \{Q\}$$

we get

$$\{P\}\ initialisation;\ \{inv\}$$
$$\{inv \wedge condition\}\ loop\ body;\ \{inv\}$$
$$\{inv \wedge \neg condition\}\ skip;\ \{Q\}$$

Termination

A loop variant is an integer-valued expression that

- is decreased at least by 1 in each execution of the loop body,
- cannot go below 0.

Then, from

$$\{P\}\ initialisation;\ while\ (condition)\ \{loop\ body\};\ \{Q\}$$

we get

$$\{int\ var \wedge var > 0\}\ loop\ body;\ \{var > var' \geq 0\}$$

Example

$$\begin{array}{ll} \{n > 0 \wedge x = 1\} & sum = 1; \\ & while\ (x < n)\ \{ \\ & x = x + 1; \\ & sum = sum + x; \\ & \}; \{sum = n(n+1)/2\} \end{array}$$

For the invariant, we try $sum = x(x+1)/2$ and we get the following Hoare Triples :

$$\{n > 0 \wedge x = 1\}\ sum = 1;\ \{sum = x(x+1)/2\}$$
$$\{sum = x(x+1)/2 \wedge x < n\}\ x = x + 1;\ sum = sum + x\ \{sum = x(x+1)/2\}$$
$$\{sum = x(x+1)/2 \wedge \neg(x < n)\}\ skip;\ \{sum = n(n+1)/2\}$$

(1) : We obtain the following formula to prove :

$$n > 0 \wedge x = 1 \wedge sum = 1 \rightarrow sum = \frac{x(x+1)}{2}$$

$1 = \frac{1(1+1)}{2} = \frac{2}{2} = 1$

(2) : We obtain the following formula to prove :

$$sum = x(x+1)/2 \wedge x < n \wedge x' = x + 1 \wedge sum' = sum + x' \rightarrow sum' = x'(x'+1)/2$$

$$sum = \frac{x(x+1)}{2}$$
$$x < n$$
$$x' = x + 1$$
$$sum' = sum + x' = sum + x + 1 = \frac{x(x+1)}{2} + x + 1$$
$$sum' = \frac{x'(x'+1)}{2}$$
$$\frac{x(x+1)}{2} + x + 1 = \frac{(x+1)(x+1+1)}{2}$$
$$\frac{x(x+1)}{2} + x + 1 = \frac{x^2+x}{2} + x + 1 = 0.5x^2 + 1.5x^2 + 1$$
$$\frac{(x+1)(x+1+1)}{2} = \frac{x^2+2+2x+x}{2} = \frac{x^2+3x+2}{2} = 0.5x^2 + 1.5x + 1$$

(3) : We obtain the following formula to prove :

$$sum = \frac{x(x+1)}{2} \wedge x \geq n \rightarrow sum = \frac{n(n+1)}{2}$$

Which is true because if $x = n$ then both side of the equation are equivalent.

termination : we try $n - x$ for the variant and we got the following formula :

$$
\begin{aligned}
&int\ var \wedge \\
&var > 0 \wedge \\
&x = 1 \wedge \\
&n > 0 \wedge \\
&var = n - x \wedge \\
&x' = x + 1 \wedge \\
&sum' = sum + x' \wedge \\
&var' = n - x' \ \rightarrow \\
&var > var' \geq 0
\end{aligned}
$$

$$var' = n - x' = n - x + 1$$

$$n - x > n - x - 1 \geq 0$$

Termination is proved.

1.3 Weakness and Strength of Predicates

P is weaker than $Q \leftrightarrow Q \rightarrow P$ ($\leftrightarrow$ stand for if and only if). true is the weakest predicate and false is the strongest one.
If P is weaker than P' ($P' \rightarrow P$), then proving $\{P\}\ S\ \{Q\}$ guarantees the truth of $\{P'\}\ S\ \{Q\}$.
If Q is stronger than Q' ($Q \rightarrow Q'$), then proving $\{P\}\ S\ \{Q\}$ guarantees the truth of $\{P\}\ S\ \{Q'\}$.

Example : assume that we have proved

$$
\begin{aligned}
&\{x > 0\} \\
&x = 1/x; \\
&\{x' > 0\} \\
&\{x \neq 0\} \\
&\text{if } (x < 0) \text{then } x = -x \text{ else } x = x; \\
&\{x' > 0\}
\end{aligned}
$$

How to prove

$$
\begin{aligned}
&\{x > 0\} \\
&x = 1/x; \\
&\text{if } (x < 0) \text{then } x = -x \text{ else } x = x; \\
&\{x' > 0\}
\end{aligned}
$$

We can assume that both of the assertion present in the first one are just replaced by the assertion $\{true\}$. Then, we would have $\{x > 0\}\ x = 1/x\ \{true\}$ where $Q' = \{true\}$. From the first one, we can extract the Hoare Triple $\{x > 0\}\ x = 1/x\ \{x' > 0 \wedge x \neq 0\}$, where $Q = \{x' > 0 \wedge x \neq 0\}$. Q is stronger that Q' and $\{P\}\ S\ \{Q\}$ is proved, then $\{P\}\ S\ \{Q'\}$ is proved too.

Chapter 2

Propositional Logic

A formula in propositional logic can be constructed as follows :

- $\top$ (true), $\bot$ (false) and propositional variables ($q, p, m, m, \dots$) are formulas of propositional logic.
- if F and G are formulas of propositional logic, then so are :
  - (F)
  - $\neg F$
  - $F \wedge G$
  - $F \vee G$
  - $F \to G$
  - $F \leftrightarrow G$

Definition. *If P is a variable, then P and $\neg P$ are called **literals**.*

Definition. *An **interpretation** I is a truth-value assignment to propositional variables $P, Q, \dots$*

$$I : \{P \mapsto \top, Q \mapsto \bot, \dots\}$$

Definition. *The truth value of a variable P under an interpretation I is denoted by $I[P]$. For example, we would have*

$$I[P] = \top, I[Q] = \bot, \dots$$

Definition. *For an interpretation I, we write*

$$I \models F$$

if and only if propositional formula F is true under the interpretation I.

Definition. *We have the following for F and G being a propositional variables :*

- $I \models \top$
- $I \not\models \bot$
- $I \models F$ *iff* $I[F] = \top$
- $I \not\models F$ *iff* $I[F] = \bot$

- $I \models (F)$ *iff* $I \models F$
- $I \models \neg F$ *iff* $I \not\models F$
- $I \models F \wedge G$ *iff* $I \models F$ *and* $I \models G$
- $I \models F \vee G$ *iff* $I \models F$ *or* $I \models G$ *or both*
- $I \models F \to G$ *iff* $I \not\models F$ *or* $I \models G$ *or both*
- $I \models F \leftrightarrow G$ *iff* $I \models F \to G$ *and* $I \models G \to F$

2.1 Proof rules

$$\frac{I \models \neg F}{I \not\models F} \qquad \frac{I \models \neg F}{I \models F} \qquad \frac{I \models F \wedge G}{\begin{array}{l} I \models F \\ I \models G \end{array}} \qquad \frac{I \not\models F \wedge G}{I \not\models F \quad | \quad I \not\models G} \qquad \frac{I \models F \vee G}{I \models F \quad | \quad I \models G} \qquad \frac{I \not\models F \vee G}{\begin{array}{l} I \not\models F \\ I \not\models G \end{array}}$$

$$\frac{I \models F \to G}{I \not\models F \quad | \quad I \models G} \qquad \frac{I \not\models F \to G}{\begin{array}{l} I \models F \\ I \not\models G \end{array}} \qquad \frac{I \models F \leftrightarrow G}{I \models F \wedge G \quad | \quad I \not\models F \vee G}$$

$$\frac{I \not\models F \leftrightarrow G}{I \models \neg F \wedge G \quad | \quad I \models F \wedge \neg G} \qquad \frac{\begin{array}{c} I \models F \\ I \not\models F \end{array}}{I \models \top}$$

Definition. *A propositional formula F is* ***satisfiable*** *if and only if there exists an interpretation I such that $I \models F$.*

Definition. *A propositional formula F is* ***valid*** *if and only if for all interpretations I, we have $I \models F$.*

Definition. *F is* ***valid*** *if and only if $\neg F$ is not satisfiable.*

Definition. *F is* ***satisfiable*** *if and only if $\neg F$ is not valid.*

Example : Prove the validity of $(P \wedge (P \to Q)) \to Q$:
We prove the satisfiability of $\neg((P \wedge (P \to Q)) \to Q)$ to prove the validity of $(P \wedge (P \to Q)) \to Q$.

1. $I \models \neg((P \wedge (P \to Q)) \to Q)$
2. $I \not\models (P \wedge (P \to Q)) \to Q$
3. $I \models P \wedge (P \to Q)$ and $I \not\models Q$
4. $I \models P$ and $I \models P \to Q$ and $I \not\models Q$
5. $I \models P$ and $I \not\models Q$ and $I \not\models P$ | $I \models P$ and $I \not\models Q$ and $I \models Q$
6. $I \models \bot$ | $I \models \bot$

$\neg((P \wedge (P \to Q)) \to Q)$ is not satisfiable, then $(P \wedge (P \to Q)) \to Q$ is valid.

2.2 Equivalence and Implication of Formulas

Definition. *Two formulas F and G are **equivalent** (written : $F \Leftrightarrow G$) if and only if $F \leftrightarrow G$ is a valid formula.*

Definition. *Formula F **implies** formula G (written $F \Rightarrow G$) if and only if $F \rightarrow G$ is a valid formula.*

Example : Does $P \wedge (P \rightarrow Q)$ imply Q ?
We just prove that $(P \wedge (P \rightarrow Q)) \rightarrow Q$ is a valid formula, then $P \wedge (P \rightarrow Q) \Rightarrow Q$.

2.3 Substitutions

Definition. *A substitution σ is a mapping from formulas to formulas :*

$$\{F_1 \mapsto G_1, F_2 \mapsto G_2, \ldots, F_n \mapsto G_n\}$$

*We denote σ's **domain** by $domain(\sigma) = \{F_1, F_2, \ldots, F_n\}$ and σ's **range** by $range(\sigma) = \{G_1, G_2, \ldots, G_n\}$. If the domain of σ consists solely of single variables, then σ is valled a **variable substitution**.*

If we apply a substitution $\sigma = \{F_1 \mapsto G_1, F_2 \mapsto G_2, \ldots, F_n \mapsto G_n\}$ to formula F, written as F_σ, then each occurence of sub-formula F_i in F replaced by G_i.
If F contains sub-formulas F_i and F_j is larger than F_i, then we substitute F_j first.
If σ is a variable substitution and F is valid, then F_σ is valid.
If $F_i \Leftrightarrow G_i$ for all i, then $F \Leftrightarrow F_\sigma$.

Example : Prove the validity of

$$((P \leftrightarrow \neg R) \wedge ((P \leftrightarrow \neg R) \rightarrow (P \leftrightarrow \neg(R \wedge \neg Q)))) \rightarrow (P \leftrightarrow \neg(R \wedge \neg Q))$$

We use the following substitution

$$\sigma = \{P \leftrightarrow \neg R \mapsto P, P \leftrightarrow \neg(R \wedge \neg Q) \mapsto Q\}$$

Then, we have

$$F_\sigma = (P \wedge (P \rightarrow Q)) \rightarrow Q$$

which is a tautology.

Definition. *A formula F is in **negation formal-form**, if and only if negations $\neg$ appear only directly in front of variables, i.e. they appear only in literals, and the formulas contain only disjunctions and conjunctions.*
We can use the following equivalences :

$$\begin{aligned}
\neg\neg F &\Leftrightarrow F \\
\neg\top &\Leftrightarrow \bot \\
\neg\bot &\Leftrightarrow \top \\
\neg(F \wedge G) &\Leftrightarrow (\neg F \vee \neg G) \\
\neg(F \vee G) &\Leftrightarrow (\neg F \wedge \neg F) \\
F \rightarrow G &\Leftrightarrow \neg F \vee G \\
F \leftrightarrow G &\Leftrightarrow (F \rightarrow G) \wedge (G \rightarrow F)
\end{aligned}$$

Definition. *A formula F is in **disjunctive normal-form** if and only if it is in negation normal-form and it is the disjunction ($\vee$) of conjunctions ($\wedge$) :*

$$F = \bigvee_i \bigwedge_j L_{ij}$$

We can use the following additional equivalences :

$$(F \vee G) \wedge H \Leftrightarrow (F \wedge H) \vee (G \wedge H)$$
$$F \wedge (G \vee H) \Leftrightarrow (F \wedge G) \vee (F \wedge H)$$

Definition. *A formula F is in **conjunctive normal-form** if and only if it is in negation normal-form and it is the conjunction ($\wedge$) of disjunctions ($\vee$) :*

$$F = \bigwedge_i (\bigvee_j L_{ij})$$

We can use the following additional equivalences :

$$(F \wedge G) \vee H \Leftrightarrow (F \vee H) \wedge (G \vee H)$$
$$F \vee (G \wedge H) \Leftrightarrow (F \vee G) \wedge (F \vee H)$$

Example : Turn $(P \wedge (P \to Q)) \to Q$ into CNF :

$$\begin{aligned}
(P \wedge (P \to Q)) \to Q &= \neg(P \wedge (P \to Q)) \vee Q \\
&= (\neg P \vee \neg(P \to Q)) \vee Q \\
&= (\neg P \vee \neg(\neg P \vee Q)) \vee Q \\
&= (\neg P \vee (\neg\neg P \vee \neg Q)) \vee Q \\
&= (\neg P \vee (P \vee \neg Q)) \vee Q \\
&= \neg Q \vee Q \\
&= \top
\end{aligned}$$

2.4 Deciding satisfiability

If formula F contains n propositional variables, then we can try out all value assignments to the variables to see whether or not F is satisfiable. The truth table of such a construction will have size $O(2^n)$. However, we can check this more space efficiently :

function SAT(F)
  if $F = \top$ **then**
    return *true*
  else if $F = \bot$ **then**
    return *false*
  else
    $P = choose\_vars(F)$
    return $SAT(F\{P \mapsto \top\}) \vee SAT(F\{P \mapsto \bot\})$
  end if

end function

However, the SAT method requires that the formula is in conjunctive normal-form, but turning a formula of the form

$$\bigvee_{i=1}^{n}(P_i \wedge Q_i)$$

is exponential.

Instead of turning the formula F into an equivalent formula in CNF, we turn it into an equisatisfiable formula F' in CNF. An equisatisfiable formula F' to F is satisfiable if and only if F is satisfiable (this is not equivalence).

Definition. *Propositional variables $rep(F)$ that represent formula F :*

$$\begin{aligned}
&rep(\top) = \top\\
&rep(\bot) = \bot\\
&rep(P) = P \text{ for variables } P\\
&rep(F) = P_F \text{ for all other formulas } F
\end{aligned}$$

Definition. *Formulas $enc(F)$ that make sure that $rep(F)$ represents formulas F :*

$$\begin{aligned}
enc(\top) &= \top\\
enc(\bot) &= \top\\
enc(P) &= \top\\
enc(\neg F) &= (\neg P_{\neg F} \vee \neg rep(F)) \wedge (P_{\neg F} \vee rep(F))\\
enc(F \wedge G) &= (\neg P_{F\wedge G} \vee rep(F)) \wedge (\neg P_{F\wedge G} \vee rep(G)) \wedge (P_{F\wedge G} \vee \neg rep(F) \vee \neg rep(G))\\
enc(F \vee G) &= (P_{F\vee G} \vee \neg rep(F)) \wedge (P_{F\vee G} \vee \neg rep(G)) \wedge (\neg P_{F\vee G} \vee rep(F) \vee rep(G))\\
enc(F \to G) &= (P_{F\to G} \vee rep(F)) \wedge (P_{F\to G} \vee \neg rep(G)) \wedge (\neg P_{F\to G} \vee \neg rep(F) \vee rep(G))\\
enc(F \leftrightarrow G) &= (\neg P_{F\leftrightarrow G} \vee \neg rep(F) \vee rep(G))\\
&\quad \wedge (\neg P_{F\leftrightarrow G} \vee rep(F) \vee \neg rep(G))\\
&\quad \wedge (P_{F\leftrightarrow G} \vee rep(F) \vee rep(G))\\
&\quad \wedge (P_{F\leftrightarrow G} \vee \neg rep(F) \vee \neg rep(G))
\end{aligned}$$

Finally, we replace formula F with the equisatisfiable formula

$$rep(F) \wedge \bigwedge_{G \in sub\_formulas(F)} enc(G)$$

where $sub\_formulas(F)$ is the set of all sub-formulas of F including F itself.

Definition. *Let F be a formula in CNF, i.e. it is the conjunction of clauses (disjunction of literals). Let C_1 and C_2 be two clauses in F both containing, but disagreeing on variable P. Assume (without loss of generality) that C_1 contains P in its positive and C_2 contains P in its negative form. We write $C_1[P]$ and $C_2[\neg P]$.*
*Then we get the following rule, denote **resolution step** :*

$$\frac{C_1 \qquad C_2}{C_1\{P \mapsto \bot\} \vee C_2\{\neg P \mapsto \bot\}}$$

$C_1\{P \mapsto \bot\} \vee C_2\{\neg P \mapsto \bot\}$ is called the resolvent of C_1 and C_2.

To check the satisfiability of a formula F

- As long as we can, we apply the resolution step.
- In case we ever derive $\perp$ in a resolution step, F is not satisfiable.
- Otherwise, if no more resolution steps can be executed, F is satisfiable.

Example : Show that $\phi = (P \wedge (P \rightarrow Q)) \rightarrow Q$ is satisfiable by using the resolution steps. First, we turn ϕ into an equisatisfiable formula by using its represent :

$$\begin{aligned} rep((P \wedge (P \rightarrow Q)) \rightarrow Q) &= P_1 \\ rep(P \wedge (P \rightarrow Q)) &= P_2 \\ rep(P \rightarrow Q) &= P_3 \\ rep(Q) &= Q \\ rep(P) &= P \end{aligned}$$

and its encoding :

$$\begin{aligned} enc(\phi) &= rep(\phi) \wedge enc(P_1) \wedge enc(P_2) \wedge enc(P_3) \\ rep(\phi) &= P_1 \\ enc(P_1) &= (P_1 \vee P_2) \wedge (P_1 \vee \neg Q) \wedge (\neg P_1 \vee \neg P_2 \vee Q) \\ enc(P_2) &= (\neg P_2 \vee P) \wedge (\neg P_2 \vee P_3) \wedge (P_2 \vee \neg P \vee \neg P_3) \\ enc(P_3) &= (P3 \vee P) \wedge (P_3 \vee \neg Q) \wedge (\neg P_3 \vee \neg P \vee Q) \\ enc(\phi) &= P_1 \\ &\quad \wedge (P_1 \vee P_2) \wedge (P_1 \vee \neg Q) \wedge (\neg P_1 \vee \neg P_2 \vee Q) \\ &\quad \wedge (\neg P_2 \vee P) \wedge (\neg P_2 \vee P_3) \wedge (P_2 \vee \neg P \vee \neg P_3) \\ &\quad \wedge (P3 \vee P) \wedge (P_3 \vee \neg Q) \wedge (\neg P_3 \vee \neg P \vee Q) \end{aligned}$$

Then we apply the resolution steps on :

$$P_1 \wedge (P_1 \vee P_2) \wedge (P_1 \vee \neg Q) \wedge (\neg P_1 \vee \neg P_2 \vee Q) \wedge (\neg P_2 \vee P) \wedge (\neg P_2 \vee P_3) \wedge (P_2 \vee \neg P \vee \neg P_3) \wedge (P_3 \vee P) \wedge (P_3 \vee \neg Q) \wedge (\neg P_3 \vee \neg P \vee Q)$$

Using P_1

$$\dfrac{\dfrac{P_1 \qquad P_1 \vee P_2 \qquad P_1 \vee \neg Q \qquad \neg P_1 \vee \neg P_2 \vee Q}{(\neg P_2 \vee Q) \wedge (P_2 \vee \neg P_2 \vee Q) \wedge (\neg P_2 \vee Q \vee \neg Q)}}{(\neg P_2 \vee Q)}$$

Using P_2

$$\dfrac{\dfrac{\neg P_2 \vee Q \qquad \neg P_2 \vee P \qquad \neg P_2 \vee P_3 \qquad P_2 \vee \neg P \vee \neg P_3}{(\neg P \vee \neg P_3 \vee Q) \wedge (\neg P \vee \neg P_3 \vee P) \wedge (\neg P \vee \neg P_3 \vee P_3)}}{(\neg P \vee \neg P_3 \vee Q)}$$

Using P_3

$$\dfrac{\dfrac{\neg P \vee \neg P_3 \vee Q \qquad P_3 \vee P \qquad P_3 \vee \neg Q}{(\neg P \vee P \vee Q) \wedge (\neg P \vee \neg Q \vee Q)}}{\top}$$

Definition. *The Boolean Constraint Propagation (BCP) on a formula F uses* ***unit resolution*** *(resolution against literals). Let P be a propositional variable, then we get the rules :*

$$\dfrac{P \qquad C[\neg P]}{C\{\neg P \mapsto \bot\}} \qquad \dfrac{\neg P \qquad C[P]}{C\{P \mapsto \bot\}}$$

Then we obtain the DPLL algorithm used in practice :

function DPLL(F)
  $F' = BCP(F)$
  if $F' = \top$ **then**
    return $true$
  else if $F' = \bot$ **then**
    return $false$
  else
    $P = choose\_vars(F')$
    return $SAT(F'\{P \mapsto \top\}) \vee SAT(F'\{P \mapsto \bot\})$
  end if
end function

Example : Show that $(P \wedge (P \to Q)) \to Q$ is satisfiable by using DPLL.

$$\mathbf{DPLL}(P_1 \wedge (P_1 \vee P_2) \wedge (P_1 \vee \neg Q) \wedge (\neg P_1 \vee \neg P_2 \vee Q) \wedge (\neg P_2 \vee P) \wedge (\neg P_2 \vee P_3) \wedge (P_2 \vee \neg P \vee$$

$$\neg P_3) \wedge (P_3 \vee P) \wedge (P_3 \vee \neg Q) \wedge (\neg P_3 \vee \neg P \vee Q)) = \mathbf{BCP}(\dots) =$$

$$(\neg P_2 \vee Q) \wedge (P_1 \vee P_2) \wedge (P_1 \vee \neg Q) \wedge (\neg P_2 \vee P) \wedge (\neg P_2 \vee P_3) \wedge (P_2 \vee \neg P \vee \neg P_3) \wedge (P_3 \vee P) \wedge$$

$$(P_3 \vee \neg Q) \wedge (\neg P_3 \vee \neg P \vee Q)$$

$$\mathbf{DPLL}(\dots [P_2 \mapsto \top]) = \mathbf{BCP}(Q \wedge P \wedge P_3 \wedge (P_1 \vee \neg Q) \wedge (P_3 \vee P) \wedge (P_3 \vee \neg Q) \wedge (\neg P_3 \vee \neg P \vee Q)) =$$

$$(w.r.t.\ \mathrm{Q}) P \wedge P_3 \wedge P_1 \wedge (P_3 \vee P) \wedge (\neg P_3 \vee \neg P \vee =$$

$$(\text{w.r.t.}\ \ P_3)\ \mathrm{P} \wedge P_1 \wedge (P_3 \wedge P) \wedge (\neg P \vee Q) =$$

$$(\text{w.r.t.}\ \ P)\ \mathrm{P}_1 \wedge (P_3 \vee P) \wedge Q =$$

$$\mathbf{DPLL}(\dots [\mathrm{P}_1 \mapsto \top]) =$$

$$\mathbf{DPLL}(\dots [\mathrm{Q} \mapsto \top]) =$$

$$\mathbf{DPLL}(\dots [\mathrm{P}_3 \mapsto \top]) =$$

$$\mathbf{DPLL}(\dots [\mathrm{P} \mapsto \top])$$

$(P \wedge (P \to Q)) \to Q$ is satisfiable with the interpretation

$$I = \{P \mapsto \top, Q \mapsto \top\}$$

Chapter 3

Computability

3.1 Undecidability of First-order Logic

Chapter 4

Complexity

4.1 Cook's Theorem

4.2 NP-completeness

4.3 PSPACE-Completeness

Chapter 5

Polynomial Time Reductions