Fausto Giunchiglia

The DPLL $\mathcal{LOP}$ decision procedure

Handouts

December 10, 2023

Disclaimer: These working notes are extracted from and somehow extend the content of the slides discussed in class (also available on line). As such, these notes may contain mistakes of various types (e.g., misspellings, typos, mistakes in formulas). Please keep this in mind and compare the content of these notes with that of the slides. In case you still have doubts, please send an email to `fausto.giunchiglia@unitn.it`. You are also welcome to send us comments aimed at improving the quality of this material. Your feedback will be used to improve the next version of these notes and

slides, which will be used in next year's lectures. Thank you in advance for your feedback.

Contents

Chapter 1
The DPLL $\mathcal{LOP}$ decision procedure

1.1 Introduction

DPLL (for Davis-Putnam-Logemann-Loveland) is the de-facto reference standard for the implementation of SAT-based reasoning. It implements satisfiability of a specific subclass of $\mathcal{LOP}$, the so called Conjunctive Normal Form (CNF).
The standard approach to using DPLL is as follows:

1. The input problem, if different from satisfiabiliy, namely if validity, unsatisfiability, logical consequence or logical equivalence, must be translated into a satisfiability problem.
2. The resulting input formula is translated into CNF.
3. DPLL is run on the CNF formula.

1.2 Conjunctive Normal Form (CNF)

1.2.1 Definitions

Definition 1.1 (Literal) A literal is either a propositional variable or the negation of a propositional variable.

Example 1.1 (Literal) Two examples or literls are: $p, \neg q$.

Definition 1.2 (Clause) A clause is a disjunction of literals,

Example 1.2 (Clause) An example of clause id $(p \vee \neg q \vee r)$.

Definition 1.3 (Conjunctive normal form (CNF)) A formula is in conjunctive normal form, if it is a conjunction of clauses,

Example 1.3 (CNF formulas) An example of CNF formula is the following

$$(p \vee \neg q \vee r) \wedge (q \vee r) \wedge (\neg p \vee \neg q) \wedge r$$

Example 1.4 (CNF formulas, special cases)

- $\{\}$
- p
- $\neg p$
- $p \wedge q \wedge r$
- $p \vee q \vee r$

Definition 1.4 A CNF formula has the following shape:

$$\left(L_{(1,1)} \vee \ldots \vee L_{(1,n_1)}\right) \wedge \ldots \wedge \left(L_{(m,1)} \vee \ldots \vee L_{(m,n_m)}\right)$$

equivalently written as:

$$\bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{n_j} L_{ij} \right)$$

where L_{ij} is the j-th literal of the i-th clause

Proposition 1.1 (Existence) *Every formula can be rewritten into Conjunctive Normal Form.*

Proposition 1.2 (Equivalence) $\models CNF(\phi) \equiv \phi$

1.2.2 Properties of CNF formulas

Proposition 1.3 (Order of literals does not matter) *If a clause is obtained by reordering the literals of a clause then the two clauses are equivalent.*

Example 1.5 (Order of literals does not matter)

$$(p \vee q \vee r \vee \neg r) \equiv (\neg r \vee q \vee p \vee r)$$

Observation 1.1 (Order of literals does not matter) The order of literals does not matter as a consequence of the commutativity of disjunction.

$$\phi \vee \psi \equiv \psi \vee \phi$$

Proposition 1.4 (Multiple literals can be merged) *If a clause contains more than one occurrence of the same literal then it is equivalent to the clause obtained by deleting all but one of these occurrences.*

Example 1.6 (Multiple literals can be merged)

$$(p \vee q \vee r \vee q \vee \neg r) \equiv (p \vee q \vee r \vee \neg r)$$

Observation 1.2 (Multiple literals can be merged) Literals can be merged because of the property of absorption of disjunction.

$$\phi \vee \phi \equiv \phi$$

Observation 1.3 (Clauses as sets of literals) We can represent a clause as a set of literals, by leaving disjunction implicit and by ignoring replication and order of literals.

Example 1.7 (Clauses as sets of literals)

$$(p \vee q \vee r \vee \neg r)$$

is represented by the set

$$\{p, q, r, \neg r\}$$

Observation 1.4 (Clauses as sets of literals) This type of representation if a proxy for the complex data structures (e.g., multi-dimensional vectors) used to implements formulas in DPLL.

Proposition 1.5 (Order of clauses does not matter) *: If a CNF formula ϕ' is obtained by reordering the clauses of a CNF formula ϕ' then ϕ and ϕ' are equivalent.*

Example 1.8 (Order of clauses does not matter)

$$(p \vee q) \wedge (r \vee \neg q) \wedge (\neg q) \equiv (r \vee \neg q) \wedge (\neg q) \wedge (p \vee q)$$

Observation 1.5 (Order of clauses does not matter) The order of clauses does not matter as a consequence of the commutativity of conjunction.

$$\phi \wedge \psi \equiv \psi \wedge \phi$$

Proposition 1.6 (Multiple clauses can be merged) *If a CNF formula contains more than one occurrence of the same clause then it is equivalent to the formula obtained by deleting all but one of the duplicated occurrences*

Example 1.9 (Multiple clauses can be merged)

$$(p \vee q) \wedge (r \vee \neg q) \wedge (p \vee q) \equiv (p \vee q) \wedge (r \vee \neg q)$$

Observation 1.6 (Multiple clauses can be merged) Multiple clauses can be merged because of the property of absorption of conjunction.

$$\phi \wedge \phi \equiv \phi$$

Observation 1.7 (CNF formulas as sets of clauses) A CNF formula can be represented as a set of sets of literals

Example 1.10 (CNF formulas as sets of clauses)

$$(p \vee q) \wedge (r \vee \neg q) \wedge (\neg \vee q)$$

is represented by

$$\{\{p, q\}, \{r, \neg q\}, \{\neg q\}\}$$

Observation 1.8 (CNF formulas as sets of clauses) This type of representation if a proxy for the complex data structures (e.g., multi-dimensional vectors) used to implements formulas in DPLL, extending the notation used for clauses.

1.2.3 Generating CNF formulas

Definition 1.5 (The CNF function) Given a PL formula ϕ the function CNF, which transforms ϕ in its CNF form, called $CNF(\phi)$ is recursively defined as follows:

$$\begin{array}{ll}
CNF(\text{p}) & = \text{p if p} \in \textbf{PROP} \\
CNF(\neg\text{p}) & = \neg\text{p if p} \in \textbf{PROP} \\
CNF(\phi \supset \psi) & = CNF(\neg\phi) \otimes CNF(\psi) \\
CNF(\phi \wedge \psi) & = CNF(\phi) \wedge CNF(\psi) \\
CNF(\phi \vee \psi) & = CNF(\phi) \otimes CNF(\psi) \\
CNF(\phi \equiv \psi) & = CNF(\phi \supset \psi) \wedge CNF(\psi \supset \phi) \\
CNF(\neg\neg\phi) & = CNF(\phi) \\
CNF(\neg(\phi \supset \psi)) & = CNF(\phi) \wedge CNF(\neg\psi) \\
CNF(\neg(\phi \wedge \psi)) & = CNF(\neg\phi) \otimes CNF(\neg\psi) \\
CNF(\neg(\phi \vee \psi)) & = CNF(\neg\phi) \wedge CNF(\neg\psi) \\
CNF(\neg(\phi \equiv \psi)) & = CNF(\phi \wedge \neg\psi) \otimes CNF(\psi \wedge \neg\phi)
\end{array}$$

where $(C_1 \wedge \ldots \wedge C_n) \otimes (D_1 \wedge \ldots \wedge D_m)$ is defined as:

$$(C_1 \vee D_1) \wedge \ldots \wedge (C_1 \vee D_m) \wedge \ldots \wedge (C_n \vee D_1) \wedge \ldots \wedge (C_n \vee D_m)$$

where C_i is a conjunction of literals (possibly a literal) and D_j is a disjunction of literals (possibly a single literal)

Example 1.11 (CNF transformation) Compute the CNF of

$$(a \wedge b) \vee \neg(c \supset d)$$

We proceed as follows:

$$
\begin{aligned}
CNF((a \wedge b) \vee \neg(c \supset d)) = \\
CNF(a \wedge b) \otimes CNF(\neg C(\supset d)) = \\
(CNF(a) \wedge CNF(b)) \otimes (CNF(c) \wedge CNF(\neg d)) = \\
(a \wedge b) \otimes (c \wedge \neg d) = \\
(a \vee c) \wedge (a \vee \neg d) \wedge (b \vee c) \wedge (b \vee \neg d) =
\end{aligned}
$$

Example 1.12 (CNF transformation, exponential explosion) Compute the CNF of

$$p1 \equiv (p2 \equiv (p3 \equiv (p4 \equiv (p5 \equiv p6))))$$

We proceed as from above. The formula resulting from the first step is:

$$CNF(p1 \supset (p2 \equiv (p3 \equiv (p4 \equiv (p5 \equiv p6))))) \wedge CNF((p2 \equiv (p3 \equiv (p4(p5 \equiv p6)))) \supset p1)$$

Notice that in the above expansion the formulas has doubles its size. Continuing in the expansion the formula will keep growing exponentially.

Observation 1.9 (Cost of generation of a CNF formulas) In the worst case the formula $CNF(\phi)$ is exponentially longer than ϕ (construct a formula where this is the case using only conjunction, disjunction and negation).

1.3 Satisfiability of a CNF formula

Proposition 1.7 (Satisfiability of a set of clauses) *Let $CNF(\phi) = \{C_0, \ldots, C_n\}$, where $C_0, \ldots C_n$ are clauses. Then we have the following:*

- *$\mathcal{I} \models \phi$, if and only if $\mathcal{I} \models C_i$ for all $i = 0, \ldots, n$;*
- *$\mathcal{I} \models C_i$, if and only if for some literal $l \in C_i, \mathcal{I} \models l$.*

Observation 1.10 (Partial assignment) To check if a model satisfies ϕ we do not need to know the truth of all the literals appearing in ϕ. For instance, if $I(p) =$ True and $I(q) =$ False, we can say that

$$\mathcal{I} \models \{\{p, q, \neg r\}, \{\neg q, s\}\}$$

is satisfiable.

Definition 1.6 (Partial evaluation) A partial evaluation is a partial function that associates to some propositional variables of the alphabet $\{P\}$ a truth value (either True or False) and can be undefined for the other elements of $\{P\}$.

Observation 1.11 (Partial evaluation) Under a partial evaluation $\mathcal{I}$, the literals and clauses can be true, false or undefined. We have the following:

- A clause is true under $\mathcal{I}$ if at least one of its literals is true;
- A clause is false (or conflicting) under $\mathcal{I}$ if all literals are false;

- In all the other cases, a clause C is undefined (or unresolved).

Within a (partial or full) evaluation, a clause is left undefined when the truth value of its literals is irrelevant to the computation of the current interpretation $\mathcal{I}$.

Definition 1.7 ((Formula simplification by positive literal)) For any CNF formula ϕ and atom p, $\phi|_p$ stands for the formula obtained from ϕ by

- replacing all occurrences of p by the truth value True ($\top$ from now on) and
- by simplifying the result by removing:
  - the clauses containing the disjunctive term $\top$;
  - the literals $\neg\top = \bot$ (where $\bot$ stands for False) in all remaining clauses.

Definition 1.8 ((Formula simplification by negative literal)) For any CNF formula ϕ and atom p, $\phi|_{\neg p}$ stands for the formula obtained from ϕ by

- replacing all occurrences of p by the truth value $\bot$ and
- by simplifying the result by removing:
  - the clauses containing the disjunctive term $\neg\ \bot = \top$;
  - the literals $\top$ in all remaining clauses.

Example 1.13 (Simplification of a formula by an evaluated literal (example))

$$\{\{p, q, \neg r\}, \{\neg p, \neg r\}\}|_{\neg p} = \{\{q, \neg r\}\}$$

- The second clause is verified because it contains $\neg p$ which we assume to be true from the $|_{\neg p}$ notation.
- The first clause which contains p isn't verified by assuming $\neg p$ as $\top$, so we leave there the clause and we try to verify it by using the remaining literals.

Proposition 1.8 (CNF satisfiability) *Let CNF$(\phi) = \{C_0, \ldots C_n\}$, where $C_0, \ldots C_n$ are the clauses in CNF(ϕ). Let us assume that we iterate the process of literal evaluation. Then the process will terminate with one of two possible situations:*

- $\{\}$*, that is, with an empty set of clauses, in which case ϕ is satisfiable;*
- $\{\ldots\{\}\ldots\}$*, that is, with a non empty set of clauses containing one empty clause, in which case ϕ this unsatisfiable.*

Observation 1.12 (CNF satisfiability) The process above will terminate, independently of the order of selection of the literals being evaluated. We have two possible outcomes.

- The first situation arises when, within CNF(ϕ), all the clauses have been progressively eliminated because of multiple occurrences of $\top$;
- The second situation arises when, within one clause in CNF(ϕ), all the literals have been progressively eliminated because of multiple occurrences of $\bot$.

Example 1.14 (CNF satisfiability) Check the satisfiability of the following formula

$$(\neg p \vee q) \wedge (\neg r \vee q)$$

1. $(\neg p \vee q) \wedge (\neg r \vee q)$
2. $\{\{\neg p, q\}, \{\neg r, q\}\}$
3. $\{\{\neg p, \top\}, \{\neg r, \top\}\}|_q$
4. $\{\}$

1.4 Normal forms

Definition 1.9 (Disjunctive Normal Form (DNF) A formula in Disjunctive Normal Form (DNF) is a disjunction of conjunctions of literals.

Example 1.15 (DNF formulas) An example of CNF formula is the following

$$(p \wedge \neg q \wedge r) \vee (q \wedge r) \vee (\neg p \wedge \neg q) \vee r$$

Example 1.16 (DNF formulas, special cases)

- $\{\}$
- p
- $\neg p$
- $p \wedge q \wedge r$
- $p \vee q \vee r$

Note how CNF formulas and DNF formulas has exactly the same special cases. Of course the process by which they get constructed is different, somehow opposite

Definition 1.10 (Disjunctive Normal Form) A DNF formula has the following shape:

$$\left(L_{(1,1)} \wedge \ldots \wedge L_{(1,n_1)}\right) \vee \ldots \vee \left(L_{(m,1)} \wedge \ldots \wedge L_{(m,n_m)}\right)$$

equivalently written as:

$$\bigvee_{i=1}^{m} \left(\bigwedge_{j=1}^{n_j} L_{ij} \right)$$

where L_{ij} is the j-th literal of the i-th clause

Proposition 1.9 (Existence) *Every formula can be rewritten into Disjunctive Normal Form.*

Proposition 1.10 (Equivalence) $\models DNF(\phi) \equiv \phi$

Observation 1.13 (Simmetry of DNF and CNF) All the steps above for CNF apply to DNF simply switching the roles of conjunction and disjunction: This applies also the process of generation of CNF/DNF formulas. Check it by yourself.

Proposition 1.11 (Length of DNF formulas) *Similarly to CNF formulas, the generation of a DNF formula from a $\mathcal{LOP}$ formula can generate a generate an exponentially long formula.*

Definition 1.11 (Negation Normal Form (NNF) A formula is in Negation Normal Form (NNF) if negation applies only to propositions

Proposition 1.12 (Existence) *Every formula can be rewritten into Negation Normal Form.*

Proposition 1.13 (Equivalence) $\models NNF(\phi) \equiv \phi$

Observation 1.14 (CNF, DNF, NNF) CNF formulas and DNF formulas are also NNF formulas

Definition 1.12 (Canonical form) A class of formulas has a canonical form if the process of reducing a $\mathcal{LOP}$ formula into a formula of that class generates a single formula, so-called in canonical form.

Proposition 1.14 (Canonical form of CNF, DNF, NNF) *CNF formulas and DNF formulas have a canonical form. NNF formulas do not have a canonical forms. For example,*

$$a \wedge (b \vee \neg c)$$

and

$$(a \wedge b) \vee (a \wedge \neg c)$$

are equivalent, and they are both in negation normal form.

Proposition 1.15 (Complexity of satisfiability of a CNF formula) *Computing the satisfiability of a CNF formula has a worst case which is exponential.*

Observation 1.15 (Complexity of satisfiability of a CNF formula) As shown above, the algorithm must find the truth assignment which makes all clauses True. The problem is the mutual influence across clauses where the same literal may occur positive in one clause and negative in another.

Proposition 1.16 (Complexity of validity of a CNF formula) *Computing the validity of a CNF formula can be done in polynomial time.*

Observation 1.16 (Complexity of validity of a CNF formula) All the clauses must be true, independently of the specific assignment. The only situation which makes it possible is when, in all the clauses, there is a literal, not necessarily the same which appears both positive and negative

Proposition 1.17 (Complexity of satisfiability of a DNF formula) *Computing the satisfiability of a DNF formula can be done in polynomial time.*

Observation 1.17 (Complexity of satisfiability of a DNF formula) For a DNF formula to be satisfiable, being it a disjunction, it is sufficient to find an assignment which makes true one of the many conjunctive clauses which appear in the disjunction. This is always possible unless, in all conjuncts, the conjunction of a formula and its negation appears, not necessarily the same.

Proposition 1.18 (Complexity of validity of a DNF formula) *Computing the validity of a DNF formula has a worst case which is exponential.*

Observation 1.18 (Complexity of validity of a DNF formula) A DNF formula is a disjunction of conjunctions. As such, it is sufficient, for each conjunct, to find an assignment which makes it true. This can be done in polynomial time. The exponential worst case generates from the fact that this test must be done for all possible interpretations.

1.5 The DPLL decision procedure

Definition 1.13 (Unit clause) If a CNF formula ϕ contains a clause $C = \{l\}$ that consists of a single literal l, it is a unit clause.

Observation 1.19 (Satisfiability of the Unit Clause) A formula ϕ containing a unit clause $\{l\}$

- is satisfiable only if l is evaluated as $\top$.
- can be simplified by appyling the rules above for l.

Definition 1.14 (Simplification of a formula by unit propagation) We have the following pseudo-code

---

while ϕ *contains a unit clause* $\{l\}$ **do**
  $\phi = \phi\,|_l$
  if $l = p$, then $\mathcal{I}(p) = \top$
  if $l = \neg p$, then $\mathcal{I}(p) = \bot$
end

Example 1.17 (Unit propagation) Consider the following CNF formula

$$\phi = \{\{p\}, \{\neg p, \neg q\}, \{\neg q, r\}\}$$

Check whether ϕ is satisfiable by unit propagation. If so, find an interpretation $\mathcal{I}$ so that $\mathcal{I} \models \phi$.

$$\begin{aligned}
&\{\{p\}, \{\neg p, \neg q\}, \{\neg q, r\}\}\\
&\{\{p\}, \{\neg p, \neg q\}, \{\neg q, r\}\}|_p\\
&\{\{\top\}, \{\bot, \neg q\}, \{\neg q, r\}\}\\
&\{\{\neg q\}, \{\neg q, r\}\}\\
&\{\{\neg q\}, \{\neg q, r\}\}|_{\neg q}\\
&\{\{\top\}, \{\top, r\}\}\\
&\{\}
\end{aligned}$$

ϕ is satisfiable, and $\mathcal{I} = \{p, \neg q\}$. The literal r is left undefined, because in order to satisfy the formula there is no need to evaluate it. This is an example of partial evaluation.

Example 1.18 (Unit propagation) Consider the following CNF formula

$$\phi = \{\{p\}, \{\neg p\}, \{\neg q, r\}\}$$

Check whether ϕ is satisfiable by unit propagation. If so, find an interpretation $\mathcal{I}$ so that $\mathcal{I} \models \phi$.

$$\{\{p\}, \{\neg p\}, \{\neg q, r\}\}$$
$$\{\{p\}, \{\neg p, \neg q\}, \{\neg q, r\}\}|_p$$
$$\{\{\top\}, \{\bot\}, \{\neg q, r\}\}$$
$$\{\{\}, \{\neg q, r\}\}$$
$$\{\ldots, \{\}, \ldots\}$$

The second clause is not satisfiable, and thus the entire formula is not satisfiable (no other options can be tried as we have failed in the assignment of a unit clause).

Observation 1.20 (Beyond Unit propagation) There are cases in which Unit Propagation does not generate one of the two termination conditions. In this case, we have to guess and assign truth values of literals. In general, but not necessarily, each unassigned literal will generate a branch of two cases, one for each truth value. Each branch may double the number of interpretation functions to be analyzed. This is where the exponential explosion arises.

Example 1.19 (Unit propagation non-termination) Consider the following formula

$$\{\{p, q\}, \{\neg q, r\}\}$$

There are no unit clauses. We need to check the truth values of the unassigned literals.

Definition 1.15 (The DPLL decision procedure - First version)

DPLL$(\phi, \mathcal{I})$
if $\phi = \{\ldots, \{\}, \ldots\}$ **then**
  return False
end
if $\phi = \{\}$ **then**
  return $\mathcal{I}$;
end
select literal $l \in C_i \in \phi$
DPLL $\left(\phi|_l, I \cup (\mathcal{I}(l) = \text{True})\right)$ **or** DPLL $\left(\phi|_{\neg l}, \mathcal{I} \cup (\mathcal{I}(l) = \text{False})\right)$

Observation 1.21 (The DPLL decision procedure - Start) At the beginning of the process, DPLL is called with $\mathcal{I} = \{\}$, namely with an empty set of assignments.

Observation 1.22 (The DPLL decision procedure - Branching) In the last line of DPLL the **or** means a branching of DPLL, where each branch will look for a different interpretation function.

Observation 1.23 (The DPLL decision procedure - Literal selection) The selection of the literal in the line before the last, in heuristic in the sense that there is no guarantee that it will generate the assignment which will converge to a solution faster. Notice that there are two steps in the decision:

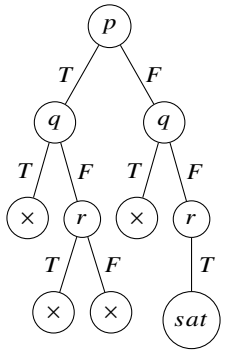
1. selection of the literal;
2. selection of which of the truth values for the selected literal should be tried first.

Observation 1.24 (The DPLL decision procedure - Backtracking) In case of a wrong decision, namely of an assignment which produces an interpretation function $\mathcal{I}$ for which ϕ is not True, then DPLL needs to bakctrack to the (usually closest, modulo advanced algorithm) decision point (see previous observation.

Example 1.20 (Backtracking) Consider the following formula

$$(p \vee \neg q) \wedge (p \vee r) \wedge (\neg p)$$

The search for an assignment can be represented by the following tree:



Notice that, thanks to Unit Propagation, the branch with $p = \top$ will not be tried, and the same for the branch $q = \bot$.

Observation 1.25 (Unit Clause heuristic) The Unit clause heuristic is guaranteed to save time as the evaluation which is dropped is guaranteed to fail.

Example 1.21 (Computational efficiency of the Unit Clause heuristic) Consider the following examples

1. $((p \supset q) \supset r) \wedge p \wedge \neg q$
2. $((p \wedge q) \vee \neg p) \supset r$
3. $(p \wedge r) \vee (\neg q \wedge p) \vee (\neg r \wedge \neg p)$

Execute DPLL with and without optimizations. Then, compute how much iterations you saved.

Observation 1.26 (Pure Literal heuristic) When a literal occurs with the same polarity in all clauses, then this heuristic suggests to try only one value: positive if the literal occurs positively, negative if the literal occurs negatively. Notice that the evaluation could succeed also with the assignment which is not considered. However:

- the complexity if the same or higher;
- the interpretation function is the same or less partial (that is with a smaller number of models)

Example 1.22 (Pure Literal heuristic) Consider the following examples

1. $(p \supset q \supset r) \wedge p \wedge q$
2. $(p \wedge q) \vee \supset r \wedge (p \supset r)$
3. $(p \wedge \neg r) \vee (q \wedge p) \vee (\neg r \wedge q)$

Execute DPLL with and without optimizations. Then, compute how much iterations you saved.

Observation 1.27 (Number of Occurrences heuristic) Count the occurrence of literals and select first the ones which occur most often. You can count the total number or the number for a given polarity. The hope is that higher number of occurrences will lead to a higher level of simplification of the resulting formula. However this heuristic is not guaranteed to succeed as less frequent literals could generate faster simplifications.

Example 1.23 (Number of Occurrences heuristic) Consider the following examples

1. $(p \supset q \supset r) \wedge p \wedge \neg q$
2. $(p \wedge q) \vee \neg p \supset r$
3. $(p \wedge r) \vee (\neg q \wedge p) \vee (\neg r \wedge \neg p)$

Execute DPLL with and without optimizations. Then, compute how much iterations you saved.

Definition 1.16 (The DPLL decision procedure - Final version)

Algorithm 1: DPLL$(\phi, \mathcal{I})$

Input: A set of clauses ϕ, an empty Interpretation Function $\mathcal{I}$
Output: An Interpretation Function $\mathcal{I}$
function DPLL(ϕ)
  if $\phi = \{\}$ is empty **then** return $\mathcal{I}$;
  if $\phi = \{\ldots, \{\}, \ldots\}$ **then** return false;
  while Unit Clause $\{l\} \in \phi$ **then**
    $\phi \leftarrow$ `unit-propagate` (l, ϕ);
  while pure literal $l \in \phi$ **then**
    $\phi \leftarrow$ `pure-literal-assign` (l, ϕ);
  $\mathcal{I} \leftarrow$ `select-literal`(ϕ);
  DPLL$(\phi \wedge \{I\})$ or **DPLL**$(\phi \wedge \{\neg(I)\})$;

Observation 1.28 (The DPLL decision procedure - Final version) In the above code, the heuristic counting the number of literals, like all the other heuristics, is embedded in the code implementing `select-literal`.

1.6 Exercises

1.6.1 Reduction to Negation Normal Form (NNF)

Exercise 1.1 Reduce to Negative Normal Form (NNF) the formula

$$\neg(\neg p \vee q) \vee (r \supset \neg s)$$

Exercise 1.2 Reduce to NNF the formula

$$(\neg p \supset q) \supset (q \supset \neg r)$$

1.6.2 Reduction to Conjunctive Normal Form (CNF)

Exercise 1.3 Reduce to Conjunctive Normal Form (CNF) the formula

$$\neg(\neg p \vee q) \vee (r \supset \neg s)$$

Exercise 1.4 Reduce to CNF the formula

$$(\neg p \supset q) \supset (q \supset \neg r)$$

Exercise 1.5 Given the WFF(well-formed-formulas) formula:

$$(A \iff B) \vee C$$

say which of the following WWF formulas are reformulations in CNF of the above formula:

1. $(\neg A \vee B \vee C) \wedge (\neg B \vee A \vee C)$
2. $(C \vee A \vee \neg B \vee C) \wedge (B \vee \neg A \vee C)$
3. $(\neg B \vee A \vee C) \wedge (A \vee B \vee \neg C)$
4. $(B \vee \neg A \vee \neg C) \wedge (\neg B \vee A \vee C)$

Exercise 1.6 Given the formula:

$$(\neg A \vee B \vee D) \wedge (A \vee \neg C) \wedge (D \vee C)$$

which of the following sequences of literal assignments could be generated by DPLL? The assignments are shown in order: then

C, D, A

means: first C, then D, then A. Choose one or more of the following:

1. D, ¬ C
2. C, A,B

3. C,A, D
4. C, A, ¬B
5. C, B, A

Exercise 1.7 Compute the CNF of the formula below

$$(q \wedge p) \vee \neg p$$

Exercise 1.8 Compute the CNF of the formula below

$$(p \supset q) \equiv (\neg q \supset \neg p)$$

Exercise 1.9 Compute the CNF of the formula below

$$(p \wedge r) \supset q$$

1.6.3 Check Satisfiability via CNN

Exercise 1.10 Check the satisfiability of the following formula

$$(p \vee q) \wedge (p \vee \neg p) \wedge (\neg q \vee q) \wedge (\neg q \vee \neg p) \wedge (\neg q \vee \neg p) \wedge (\neg q \vee \neg p) \wedge (\neg q \vee q) \wedge (p \vee \neg p) \wedge (p \vee q)$$

Exercise 1.11 Check the satisfiability of the following formula

$$(q \vee \neg p) \wedge (q \vee \neg p)$$

Exercise 1.12 Check the satisfiability of the following formula

$$(q \vee \neg p) \wedge (\neg q \vee p) \wedge (p \vee q)$$

Exercise 1.13 Consider the following CNF formula

$$\phi = \{\{p\}, \{\neg p\}, \{\neg q, r\}\}$$

is satisfiable and if so, find an interpretation / so that $I \models \phi$.

Solutions

Exercises of Chapter 1

1.6.4 Reduction to Negative Normal Form (NNF)

Solution **1.1**

1. $\neg(\neg p \vee q) \vee (\neg r \vee \neg s)$
2. $(\neg\neg p \wedge \neg q) \vee (\neg r \vee \neg s)$
3. $(p \wedge \neg q) \vee (\neg r \vee \neg s)$

Solution **1.2**

1. $\neg(\neg p \rightarrow q) \vee (q \rightarrow \neg r)$
2. $\neg(p \vee q) \vee (\neg q \vee \neg r)$
3. $(\neg p \wedge \neg q) \vee (\neg q \vee \neg r)$

1.6.5 Reduction to Conunctive Normal Form (NNF)

Solution **1.3**

1. $\neg(\neg p \vee q) \vee (\neg r \vee \neg s)$
2. $(\neg\neg p \wedge \neg q) \vee (\neg r \vee \neg s)$
3. $(p \wedge \neg q) \vee (\neg r \vee \neg s)$ *NNF*
4. $(p \vee \neg r \vee \neg s) \wedge (\neg q \vee \neg r \vee \neg s)$

Solution **1.4**

1. $\neg(\neg p \rightarrow q) \vee (q \rightarrow \neg r)$
2. $\neg(p \vee q) \vee (\neg q \vee \neg r)$
3. $(\neg p \wedge \neg q) \vee (\neg q \vee \neg r)$ *NNF*
4. $(\neg p \vee \neg q \vee \neg r) \wedge (\neg q \vee \neg r)$

Solution **1.5**

- $(\neg A \vee B \vee C) \wedge (\neg B \vee A \vee C)$
- $(C \vee A \vee \neg B \vee C) \wedge (B \vee \neg A \vee C)$

Solution **1.6**

- D,¬ C
- C, A, B
- C, A, D

Solution **1.7** [Example $(q \wedge p) \vee \neg p$]

$$\begin{aligned} & CNF((q \wedge p) \vee \neg p) \\ & CNF((q \wedge p)\ ) \otimes CNF(\ \neg p) \\ & (CNF(q\ ) \wedge CNF(\ p)) \otimes \neg p \\ & (q \wedge p) \otimes \neg p \\ & (q \vee \neg p) \wedge (q \vee \neg p) \end{aligned}$$

Solution **1.8** [Example $(p \supset q) \equiv (\neg q \supset \neg p)$]

$CNF((p \supset q) \equiv (\neg q \supset \neg p))$
$CNF((p \supset q) \supset (\neg q \supset \neg p)\ ) \wedge CNF(\ (\neg q \supset \neg p) \supset (p \supset q))$
$CNF((\neg(\mathrm{p} \supset \mathrm{q}))\ ) \otimes CNF(\ (\neg \mathrm{q} \supset \neg \mathrm{p})\ ) \wedge CNF(\ (\neg(\neg \mathrm{q} \supset \neg \mathrm{p}))\ ) \otimes CNF(\ (\mathrm{p} \supset \mathrm{q}))$
$(CNF(\mathrm{p}\ ) \wedge CNF(\ \neg q)\ ) \otimes (CNF(\mathrm{q}\ ) \otimes CNF(\ \neg \mathrm{p})) \wedge (CNF((\neg q)\ ) \wedge CNF(\ \mathrm{p})) \otimes$
$(CNF(\neg \mathrm{p}\ ) \otimes CNF(\ \mathrm{q}))$
$(p \wedge \neg q) \otimes (q \otimes \neg p) \wedge (\neg q \wedge p) \otimes (\neg p \otimes q)$
$(p \wedge \neg q) \otimes (q \wedge \neg p) \wedge (\neg q \wedge p) \otimes (\neg p \wedge q)$
$(p \vee q) \wedge (p \vee \neg p) \wedge (\neg q \vee q) \wedge (\neg q \vee \neg p) \wedge (\neg q \vee \neg p) \wedge (\neg q \vee \neg p) \wedge (\neg q \vee$
$q) \wedge (p \vee \neg p) \wedge (p \vee q)$

Solution **1.9** [Example $(p \wedge r) \supset q$]

$$\begin{aligned} & CNF((p \wedge r) \supset q) \\ & CNF((\neg(p \wedge r))\ ) \otimes CNF(\ q) \\ & (CNF((\neg p)\ ) \otimes CNF(\ \neg r)) \otimes q \\ & (\neg p \otimes \neg r) \otimes q \\ & (\neg p \vee \neg r) \otimes q \\ & (\neg p \vee q) \wedge (\neg r \vee q) \end{aligned}$$

1.6.6 Check satisfiability cia CNN

Solution **1.10** Here is the solution by using DPLL:

1. $(p \vee q) \wedge (p \vee \neg p) \wedge (\neg q \vee q) \wedge (\neg q \vee \neg p) \wedge (\neg q \vee \neg p) \wedge (\neg q \vee \neg p) \wedge (\neg q \vee$ $q) \wedge (p \vee \neg p) \wedge (p \vee q)$
2. $\{\{p, q\}, \{p, \neg p\}, \{\neg q, q\}, \{\neg q, \neg p\}, \{\neg q, \neg p\}, \{\neg q, \neg p\}, \{\neg q, q\}, \{p, \neg p\}, \{p, q\}\}$

3. $\{\{\top, q\}, \{\top, \bot\}, \{\neg q, q\}, \{\neg q, \bot\}, \{\neg q, \bot\}, \{\neg q, \bot\}, \{\neg q, q\}, \{\top, \bot\}, \{\top, q\}\}|_p$
4. $\{\{\neg q, q\}, \{\neg q\}, \{\neg q\}, \{\neg q\}, \{\neg q, q\}\}$
5. $\{\{\top, \bot\}, \{\top\}, \{\top\}, \{\top\}, \{\top, \bot\}\}|_{\neg q}$
6. $\{\}$

Solution **1.11** Here is the solution by using DPLL:

1. $(q \vee \neg p) \wedge (q \vee \neg p)$
2. $\{\{q, \neg p\}, \{q, \neg p\}\}$
3. $\{\{q, \top\}, \{q, \top\}\}|_{\neg p}$
4. $\{\}$

Solution **1.12** Here is the solution by using DPLL:

1. $(q \vee \neg p) \wedge (\neg q \vee p) \wedge (p \vee q)$
2. $\{\{q, \neg p\}, \{\neg q, p\}, \{p, q\}\}$
3. $\{\{q, \top\}, \{\neg q, \bot\}, \{\bot, q\}\}|_{\neg p}$
4. $\{\{\neg q\}, \{q\}\}$
5. $\{\{\bot\}, \{\top\}\}|_q$
6. $\{\{\}\}$

Solution **1.13**

$\{\{p\}, \{\neg p\}, \{\neg q, r\}\}$
$\{\{p\}, \{\neg p\}, \{\neg q, r\}\}|_p$
$\{\{\top\}, \{\bot\}, \{\neg q, r\}\}$
$\{\{\}, \{\neg q, r\}\}$
$\{\ldots, \{\}, \ldots\}$

The second clause cannot be verified, and thus the entire formula is not satisfiable.