Chapter 4

Satisfiability Checking

I can't get no satisfaction . . . I try and I try and I try and I try
The Rolling Stones

Contents

Checking satisfiability of a set of propositional formulas is a famous decision problem having a number of applications.

DEFINITION 4.1 (Satisfiability Problem) *Satisfiability Problem* is the following decision problem. An instance is a finite set of formulas. The answer is "yes" if this set of formulas is satisfiable. ❑

The satisfiability problem is NP-complete. In fact, it is one of the most well-known NP-complete problems. If you open a textbook on complexity theory, propositional satisfiability is likely to be the first NP-complete problem introduced in this textbook.

There is a connection between theorem proving in mathematics and the satisfiability problem (though for more general notions of formulas than those presented here). In *mathematics*, a theorem is usually presented in the following way. Given formulas $A_1, \ldots, A_n$ (the *axioms*, or the *theory*), prove a formula G (the *conjecture*). In *mathematical logic*, provability of G from $A_1, \ldots, A_n$ is equivalent to checking validity of the implication $A_1 \wedge \ldots \wedge A_n \rightarrow G$. In *automated reasoning*, this instance of the validity problem can be represented as an instance of the (un)satisfiability problem: check that the set of formulas $\{A_1, \ldots, A_n, \neg G\}$ is unsatisfiable.

Consider an example how a solution to a logical puzzle can be represented as an instance of the satisfiability problem for propositional logic.

EXAMPLE 4.2 (Russian Spy Puzzle) Consider the following logical puzzle related a famous Soviet film about the Second World War. There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans. Moreover, every Russian must be a spy. When Stirlitz meets Müller in a corridor, he makes the following joke: "you know, Müller, you are as German as I am Russian". It is known that Stirlitz always says the truth when he is joking. We have to establish that Eismann is not a Russian spy.

In order to formalize this joke in propositional logic, we introduce several boolean variables denoting atomic propositions occurring in the puzzle. These variables are

(1) GS, GM, GE denoting that respectively Stirlitz, Müller, or Eismann is German.

(2) RS, RM, RE denoting that respectively Stirlitz, Müller, or Eismann is Russian.

(3) SS, SM, SE denoting that respectively Stirlitz, Müller, or Eismann is a spy.

Let us try to express our knowledge about the puzzle using propositional formulas. A possible formalization is shown in Figure 4.1 on the next page.

The statement that exactly one of the three persons is Russian, while the other two are Germans can be formalized as formula (1) in the figure. Formula (2) expresses that every Russian is a spy. Finally, Stirlitz's joke is formalized as formula (3). We also have to formalize the hidden assumption that Russians are not Germans, see formulas (4)–(6).

To check that Eismann is not a Russian spy, we can add formula (7) to our list of assumptions and show that the resulting set of formulas (1)–(7) is unsatisfiable.

Note that using the notion of satisfiability we can formalize other questions about this puzzle. For example, to show that Eismann can be a spy, we can replace (7) by the formula SE and check that the resulting set of formulas has a model, i.e. it is satisfiable. Likewise, to show that Eismann can be not a spy, we can replace formula (7) by $\neg SE$ and check that the resulting set of formulas is satisfiable too. ❑

4.1 Truth Tables

There exists a simple, but not very efficient, method of satisfiability checking by an explicit enumeration of all interpretations for the boolean variables occurring in the formula. This method is a straightforward extension of the formula evaluation method illustrated in Example 3.11 on page 36. As in that example, we build a table whose rows are the subformulas of the formula checked for satisfiability. But instead of one column containing the values of all subformulas in an interpretation we create several columns, one for every interpretation for the boolean variables of the formula. Such tables are usually called *truth tables*. We will illustrate the truth table method as an extension of Example 3.11.

(1)	$(RS \wedge GM \wedge GE) \vee (GS \wedge RM \wedge GE) \vee (GS \wedge GM \wedge RE)$
(2)	$(RS \rightarrow SS) \wedge (RM \rightarrow SM) \wedge (RE \rightarrow SE)$
(3)	$RS \leftrightarrow GM$
(4)	$RS \leftrightarrow \neg GS$
(5)	$RM \leftrightarrow \neg GM$
(6)	$RE \leftrightarrow \neg GE$
(7)	$RE \wedge SE$

Figure 4.1: The Russian Spy Puzzle

	subformula	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
1	$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$	1	1	1	1	1	1	1	1
2	$p \rightarrow r$	1	1	1	1	0	1	0	1
3	$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$	1	1	1	1	0	0	0	1
4	$p \wedge q \rightarrow r$	1	1	1	1	1	1	0	1
5	$p \rightarrow q$	1	1	1	1	0	0	1	1
6	$p \wedge q$	0	0	0	0	0	0	1	1
7	p p p	0	0	0	0	1	1	1	1
8	q q	0	0	1	1	0	0	1	1
9	r r	0	1	0	1	0	1	0	1

Figure 4.2: A truth table

EXAMPLE 4.3 Consider the formula $A = (p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$. There are three boolean variables occurring in this formula, namely p, q, r, hence there are $2^3 = 8$ interpretations to be considered. Let us build a table similar to that of Example 3.11, but with 8 columns representing these interpretations. The table is given in Figure 4.2. It follows that the formula is true in all of these interpretations, and so it is valid. ❑

Using truth tables for satisfiability or validity checking is not very effective. Indeed, if the formula contains n variables, then the number of possible interpretations is 2^n. State-of-the-art satisfiability checkers are able to routinely solve some instances of the satisfiability problems with thousands of variables, which would be impossible using truth tables. All efficient methods of satisfiability checking use, in one or another form, the following observation.

As we noted in Section 3.5, to evaluate a formula in an interpretation I, it is not necessary to know the value of all variables in I. For example, if we know that $I(r) = 1$, then we also know that $I(p \rightarrow r) = 1$, and hence $I((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)) = 1$. This

	subformula	I_0	I_1	I_2	I_3
1	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	1	1	1	1
2	$p \to r$	1	0	0	1
3	$(p \to q) \wedge (p \wedge q \to r)$	1	0	0	
4	$p \wedge q \to r$	1	1	0	1
5	$p \to q$	1	0	1	
6	$p \wedge q$	0	0	1	
7	p p p	0	1	1	
8	q q		0	1	
9	r r	0	0	0	1

Figure 4.3: The compact truth table for the order r, p, q

	subformula	I_0	I_1	I_2	I_3	I_4
1	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	1	1	1	1	1
2	$p \to r$	1	0	1	0	1
3	$(p \to q) \wedge (p \wedge q \to r)$	1	0	0	0	1
4	$p \wedge q \to r$	1	1	1	0	1
5	$p \to q$	1	0	0	1	1
6	$p \wedge q$	0	0	0	1	1
7	p p p	0	1	1	1	1
8	q q		0	0	1	1
9	r r		0	1	0	1

Figure 4.4: The compact truth table for the order p, q, r

observation gives us an idea of more compact truth tables, in which some truth values are unknown. For example, assuming that we consider the possible truth values $0, 1$ for r first, then p and then q, we will build the truth table given in Figure 4.3. If we choose a different order p, q, r, we obtain the truth table given in Figure 4.4. Therefore, the size of the tables can crucially depend on the order of evaluating the variables. This will become more clear when we consider ordered binary decision diagrams (OBDDs) and relations between the truth tables and OBDDs.

In the next section we consider the splitting method that can be regarded as a syntactic analogue of the compact truth tables.

4.2 Splitting

In this section we describe a simple method, called *splitting*, of checking satisfiability of a propositional formula or a set of formulas. The method is based on considering truth values

for boolean variables occurring in the formula and *simplifying* the formula depending on these values. After simplification it may turn out that it is not always necessary to find truth values for *all* boolean variables the formula.

To explain the idea of splitting, consider again the formula $(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$. In every interpretation I such that $I \vDash r$, this formula is true since $(p \to q) \wedge (p \wedge q \to \top) \to (p \to \top) \equiv (p \to q) \wedge (p \wedge q \to \top) \to \top \equiv \top$. If $I \nvDash r$, we cannot say immediately what is the value of this formula but we know that in this interpretation the formula is equivalent to $(p \to q) \wedge (p \wedge q \to \bot) \to (p \to \bot)$, which can be simplified to the equivalent formula $(p \to q) \wedge \neg(p \wedge q) \to \neg p$.

The splitting method can be considered both as a generalization of the formula evaluation method based on simplification rules of Section 3.6, and also as a syntactic analogue of compact truth tables. In compact truth tables, as soon as we know a value of a subformula B, we try to propagate this value to subformulas in which B occurs, and hopefully to the evaluated formula itself. In the splitting method we implement this propagation using simplification rules that replace subformulas by equivalent ones.

The theoretical basis for the splitting method is the following simple lemma. For every propositional formula A and atom p, denote by $A_p^\bot$ and $A_p^\top$ the formulas obtained by replacing in A all occurrences of p by $\bot$ and $\top$, respectively.

LEMMA 4.4 *Let p be an atom, A be a formula, and I be an interpretation.*

(*1*) *If $I \vDash \neg p$, then A is equivalent to $A_p^\bot$ in I.*

(*2*) *If $I \vDash p$, then A is equivalent to $A_p^\top$ in I.* ❑

PROOF. We prove only the second part, the first one is similar. We have $I \vDash \top \leftrightarrow p$. By Lemma 3.8 on page 35 we have $I \vDash A \leftrightarrow A_p^\top$, so $I \vDash A$ if and only if $I \vDash A_p^\top$. ❑

This lemma implies the following theorem which serves as a foundation for the method of splitting.

THEOREM 4.5 *Let A be a formula and p be an atom. Then A is satisfiable if and only if at least one of the formulas $A_p^\top$ and $A_p^\bot$ is satisfiable.*

PROOF. Suppose that A is satisfiable. Then for some interpretation I we have $I \vDash A$. Consider the case when $I \vDash p$ (the case $I \vDash \neg p$ is similar). In this case by Lemma 4.4 we have $I \vDash A \leftrightarrow A_p^\top$. Therefore, $I \vDash A_p^\top$, so $A_p^\top$ is satisfiable.

Conversely, suppose that $A_p^\top$ is satisfiable (the case of $A_p^\bot$ is similar). Then for some interpretation I we have $I \vDash A_p^\top$. Define an interpretation I' as follows: for all boolean variables q we have

$$I'(q) \stackrel{\text{def}}{=} \begin{cases} I(q), & \text{if } p \neq q; \\ 1, & \text{if } p = q. \end{cases}$$

Simplification rules for $\top$:

$$\neg\top \;\Rightarrow\; \bot$$
$$\top \wedge A_1 \wedge \ldots \wedge A_n \;\Rightarrow\; A_1 \wedge \ldots \wedge A_n$$
$$\top \vee A_1 \vee \ldots \vee A_n \;\Rightarrow\; \top$$
$$A \rightarrow \top \;\Rightarrow\; \top \qquad \top \rightarrow A \;\Rightarrow\; A$$
$$A \leftrightarrow \top \;\Rightarrow\; A \qquad \top \leftrightarrow A \;\Rightarrow\; A$$

Simplification rules for $\bot$:

$$\neg\bot \;\Rightarrow\; \top$$
$$\bot \wedge A_1 \wedge \ldots \wedge A_n \;\Rightarrow\; \bot$$
$$\bot \vee A_1 \vee \ldots \vee A_n \;\Rightarrow\; A_1 \vee \ldots \vee A_n$$
$$A \rightarrow \bot \;\Rightarrow\; \neg A \qquad \bot \rightarrow A \;\Rightarrow\; \top$$
$$A \leftrightarrow \bot \;\Rightarrow\; \neg A \qquad \bot \leftrightarrow A \;\Rightarrow\; \neg A$$

Figure 4.5: Simplification rules for $\top$ and $\bot$

Note that I and I' agree on all boolean variables except maybe p, and p does not occur in $A_p^\top$. This and $I \vDash A_p^\top$ implies $I' \vDash A_p^\top$. We have $I \vDash p$, so by Lemma 4.4 we have $I' \vDash A \leftrightarrow A_p^\top$. Therefore, $I' \vDash A$ and so A is satisfiable. ❑

This theorem can be applied to satisfiability-checking as formulas. Given a formula G, we select a boolean variable p occurring in G and check whether at least one of the formulas: $G_p^\top$ and $G_p^\bot$ is satisfiable. We can interleave this procedure of "splitting" into the cases $G_p^\top$ and $G_p^\bot$ with replacing the formulas $G_p^\top$ and $G_p^\bot$ by equivalent but "simpler" formulas.

To simplify formulas containing occurrences of $\top$ and $\bot$, we will use the rewrite rule system on formulas given in Figure 4.5. These rules are applied modulo permutation of arguments of $\wedge$ and $\vee$, that is, we do not make a difference between a formula $B_1 \wedge \ldots \wedge B_n$ and any formula $B_{i_1} \wedge \ldots \wedge B_{i_n}$, where $i_1, \ldots, i_n$ is a permutation of $1, \ldots, n$.

To illustrate these simplification rules, consider formula (1) of Figure 4.1 on page 43: $(RS \wedge GM \wedge GE) \vee (GS \wedge RM \wedge GE) \vee (GS \wedge GM \wedge RE)$. Let us replace the atom GE in this formula by $\bot$. We obtain the formula $(RS \wedge GM \wedge \bot) \vee (GS \wedge RM \wedge \bot) \vee (GS \wedge GM \wedge RE)$. We can apply the simplification rules as follows:

$$\begin{gathered}(RS \wedge GM \wedge \bot) \vee (GS \wedge RM \wedge \bot) \vee (GS \wedge GM \wedge RE) \Rightarrow \\ \bot \vee (GS \wedge RM \wedge \bot) \vee (GS \wedge GM \wedge RE) \Rightarrow \\ (GS \wedge RM \wedge \bot) \vee (GS \wedge GM \wedge RE) \Rightarrow \\ \bot \vee (GS \wedge GM \wedge RE) \Rightarrow \\ GS \wedge GM \wedge RE.\end{gathered}$$

We could apply the simplification rules in a different order, but the result will be the same (see Exercise 4.6).

Observe the following properties of this rewrite rule system:

(1) This system is terminating since every rewrite step rewrites a formula into a formula of a smaller size, so every formula B can be rewritten into a normal form B'.

(2) A normal form B' of B is equivalent to B, moreover

(a) every variable occurring in B' also occurs in B;

(b) either $B' = \top$, or $B' = \bot$, or B' contains no occurrences of $\top$ and $\bot$.

Indeed, the equivalence of B and B' follows from the fact that in every rule of Figure 4.5, the left-hand side of the rule is equivalent to its right-hand side. So, by Equivalent Replacement Theorem 3.9 on page 36 B' is equivalent to B.

Before defining the splitting algorithm, let us introduce a notion of *signed formula* which we will extensively use later.

DEFINITION 4.6 (Signed Formula) A *signed formula* is an expression $A = b$, where A is a formula and b a boolean value. A signed formula $A = b$ is *true* in an interpretation I, denoted by $I \vDash A = b$, if $I(A) = b$. If $A = b$ is true in I, we also say that I is a *model* of $A = b$, or that I *satisfies* $A = b$. A signed formula is *satisfiable* if it has a model. ❑

Observe the following properties of this definition:

(1) For every formula A and interpretation I exactly one of the signed formulas $A = 1$ and $A = 0$ is true in I.

(2) A formula A is satisfiable if and only if so is the signed formula $A = 1$.

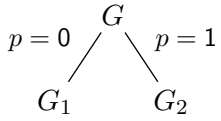
A signed atom $p = b$ can be considered as a truth assignment of the boolean value b to the variable p. The splitting method can be formalized as a sequence of truth assignments to variables of a formula.

ALGORITHM 4.7 (Splitting) The algorithm is shown in Figure 4.6 on the next page. The input is a formula G. The output is either "satisfiable" or "unsatisfiable". The algorithm is parametrized by a function $select\_signed\_atom$ which selects a signed atom $p = b$, where p is an atom occurring in G. If the selected signed atom is $p = 1$, then the algorithm first tries to find a model of G in which p is true. If such an interpretation is not found, it tries to a model of G in which p is false. If the selected atom is $p = 0$, then the algorithm first tries to find a model of G in which p is false, and then a model of G in which p is true. The method is called "splitting" because the current goal G is "split" in two subgoals: $G_p^\bot$ and $G_p^\top$. The function $simplify$ used in the algorithm simplifies the formula using the rewrite rules of Figure 4.5. ❑

This algorithm is best illustrated by a tree, called the *splitting tree*, which encodes the splitting steps. The nodes of the tree are labeled by formulas (the formula G at different steps of the algorithm). If a signed atom $p = 1$ has been selected at some step, we mark the children of the node G by the formulas $G_1 = simplify(G_p^\bot)$ and $G_2 = simplify(G_p^\top)$, and similar in the case when $p = 0$ has been selected. Arcs of the tree are labeled by the corresponding signed atoms.

```
procedure split(G)
parameters: function select_signed_atom
input: formula G
output: "satisfiable" or "unsatisfiable"
begin
  G := simplify(G)
  if G = ⊤ then return "satisfiable"
  if G = ⊥ then return "unsatisfiable"
  (p = b) := select_signed_atom(G)
  case b of
  1 ⇒
    if split(G_p^⊤) ="satisfiable"
      then return "satisfiable"
      else return split(G_p^⊥)
  0 ⇒
    if split(G_p^⊥) ="satisfiable"
      then return "satisfiable"
      else return split(G_p^⊤)
end
```

Figure 4.6: Splitting Algorithm for propositional satisfiability



EXAMPLE 4.8 Consider the negation of the formula of Example 4.3: $\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r))$. This formula is unsatisfiable. Two different splitting trees for this formula are given in Figure 4.7.

Observe the following:

(1) The size of a tree depends on the choices of signed atoms made by the algorithm.

(2) The order of choices on different branches may be different. For example, in the leftmost branch of the leftmost tree a split on the variable p was made but there was no split on r, while in the rightmost branch of the same tree a split on the variable r preceded a split on p. ❑

In the case when the formula is satisfiable, the size of the tree may also depend on the order of selecting boolean values for variables, that is, whether $p = 1$ or $p = 0$ is selected.

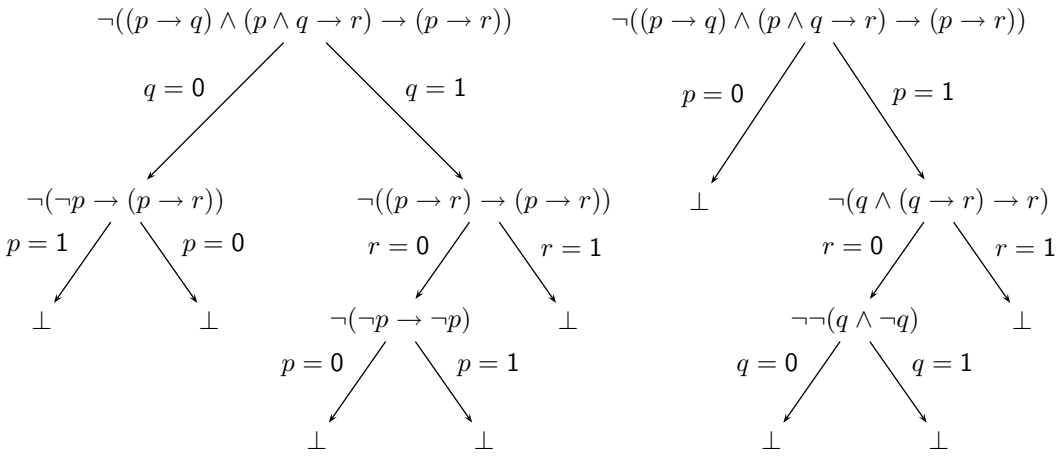


Figure 4.7: Two splitting trees for the same formula

THEOREM 4.9 *The Splitting Algorithm is sound, complete, and terminating. More exactly, for every formula G, (i) if G is satisfiable then the algorithms terminates and returns "satisfiable"; (ii) if G is unsatisfiable, then the algorithm terminates and returns "unsatisfiable".* ❑

PROOF. Termination follows from the following observation. Every splitting step decreases the number of different variables occurring in A: indeed, every variable occurring in either $A_p^\top$ or $A_p^\bot$ also occurs in A, but neither of these two formulas contains p. All other properties follow from termination and Theorem 4.5. ❑

Note the following property of splitting trees. If the tree contains an arc labeled by $p = b$ and connecting nodes A and B, then for every interpretation I such that $I(p) = b$ we have $I \vDash A \leftrightarrow B$. Consider any branch in the tree connecting the formula G with a node $\top$. Let the arcs on this branch are labeled by $p_1 = b_1, \ldots, p_n = b_n$. This property implies that every interpretation I such that $I(p_1) = b_1, \ldots, I(p_n) = b_n$ satisfies G. Therefore, the splitting algorithm can also be used to find models of satisfiable formulas.

It is straightforward to modify the splitting algorithm for finite sets of formulas. Such a generalization is given in Figure 4.8 on the following page. It works with finite sets of formulas instead of single formulas. The function $simplify$ simplifies every member of the set of formulas and removes $\top$ from the resulting set.

A splitting tree using sets of formulas for the Russian spy puzzle is shown in Figure 4.9 on page 51. In this tree we write $\bot$ for every set of formulas containing $\bot$. Since every leaf of the tree is $\bot$, the initial set of formulas is unsatisfiable.

procedure $split(\mathcal{G})$
parameters: function $select\_signed\_atom$
input: set of formulas $\mathcal{G}$
output: "satisfiable" or "unsatisfiable"
begin
  $\mathcal{G}$:= $simplify(\mathcal{G})$
  if $\bot \in \mathcal{G}$ **then return** "unsatisfiable"
  if $\mathcal{G} = \phi$ **then return** "satisfiable"
  $(p = b)$:= $select\_signed\_atom(\mathcal{G})$
  case b **of**
  $1 \Rightarrow$
    if $split(\mathcal{G}_p^\top) =$"satisfiable"
      then return "satisfiable"
      else return $split(\mathcal{G}_p^\bot)$
  $0 \Rightarrow$
    if $split(\mathcal{G}_p^\bot) =$"satisfiable"
      then return "satisfiable"
      else return $split(\mathcal{G}_p^\top)$
end

Figure 4.8: The Splitting Algorithm for sets of formulas

4.3 Polarity of Subformulas

In this section we introduce a notion of *polarity* of a subformula. This notion allows us to introduce an important optimization in propositional satisfiability algorithms.

Introduce an order $<$ on truth values by defining $0 < 1$ and consider the behavior of the logical connectives w.r.t. this order. The order $<$ induces the relation $\leq$ on truth values: we have $0 \leq 0$, $0 \leq 1$, $1 \leq 1$ but not $1 \leq 0$. Recall the definition of monotonicity on functions as applied to boolean functions.

DEFINITION 4.10 (Monotonic and anti-monotonic functions) Let $f(x_1, \ldots, x_n)$ be a function of n arguments and k an integer such that $1 \leq k \leq n$. We say that f is *monotonic* on its kth argument if for all values $v_1, \ldots, v_n, v'_k$, if $v_k \leq v'_k$, then $f(v_1, \ldots, v_k, \ldots, v_n) \leq f(v_1, \ldots, v'_k, \ldots, v_n)$. Likewise, We say that f is *anti-monotonic* on its kth argument if for all values $v_1, \ldots, v_n, v'_k$, if $v_k \leq v'_k$, then $f(v_1, \ldots, v'_k, \ldots, v_n) \leq f(v_1, \ldots, v_k, \ldots, v_n)$. ❑

Logical connectives can be regarded as boolean functions, so we can consider their behaviour with respect to monotonicity. The connectives $\wedge$ and $\vee$ are easily seen to be monotonic on all of their arguments. This means that $I(A_1 \wedge \ldots \wedge A_k \wedge \ldots \wedge A_n) \leq$

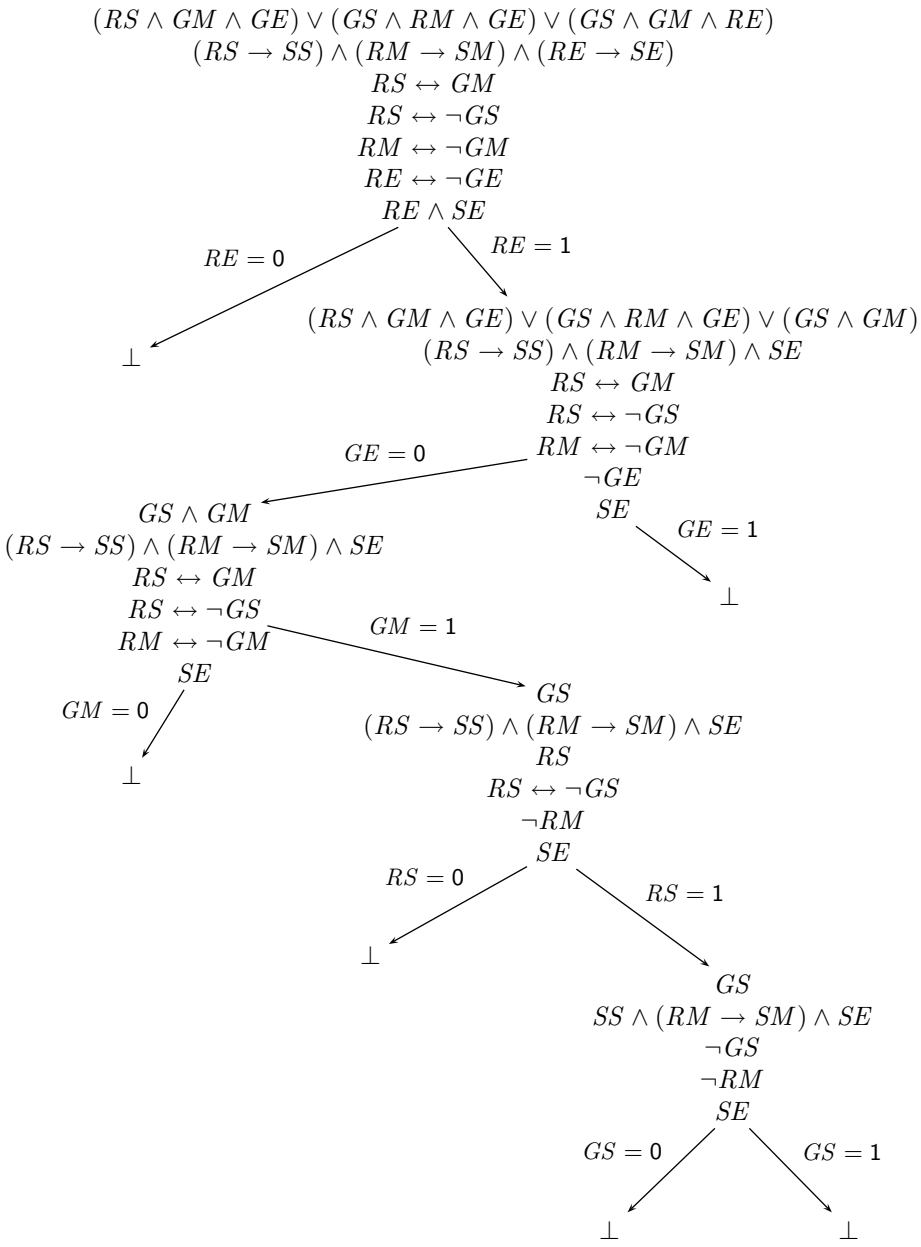


Figure 4.9: A splitting tree for the Russian Spy puzzle

$I(A_1 \wedge \ldots \wedge A'_k \wedge \ldots \wedge A_n)$, whenever $I(A_k) \leq I(A'_k)$, and similar for $\vee$ instead of $\wedge$. The negation $\neg$ is anti-monotonic, i.e. if $I(A) \leq I(B)$, then $I(\neg B) \leq I(\neg A)$. The implication $\rightarrow$ is monotonic on its second argument, but anti-monotonic on its first argument. The equivalence $\leftrightarrow$ is neither monotonic nor anti-monotonic on either of its arguments.

The notions of monotonicity and anti-monotonicity can be regarded as a description of behavior of the values of formulas depending on the values of their immediate subformulas. For example, if the value of B in $A \rightarrow B$ increases, then the value of $A \rightarrow B$ cannot decrease. We will introduce a notion of *polarity* which formalizes a relation between a formula and its subformulas with respect to monotonicity. To define this notion, we will recall the definitions of *position* and *subformula at a position*, see Section 2.7.3.

DEFINITION 4.11 (Position, Polarity) A *position* is any sequence of positive integers $a_1, \ldots, a_n$, where $n \geq 0$, written as $a_1.a_2. \ \ldots \ .a_n$. When $n = 0$, i.e. the sequence is empty, we call it the *empty position* and denote it by ϵ. A *polarity* is one of the values $-1, 0, 1$.

Let us define by induction the notions of (i) *position in a formula*, (ii) the *subformula of a formula A at a position π*, denoted by $A|_\pi$, and (iii) the *polarity of the subformula at a position π*, denoted $pol(A, \pi)$ as follows.

(1) For every formula A, ϵ is a position in A, $A|_\epsilon \stackrel{\text{def}}{=} A$ and $pol(A, \epsilon) \stackrel{\text{def}}{=} 1$;

(2) Let $A|_\pi = B$.

(a) If B has the form $B_1 \wedge \ldots \wedge B_n$ or $B_1 \vee \ldots \vee B_n$, then for all $i \in \{1, \ldots, n\}$ the position $\pi.i$ is a position in A, $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$, and $pol(A, \pi.i) \stackrel{\text{def}}{=} pol(A, \pi)$.

(b) If B has the form $\neg B_1$, then $\pi.1$ is a position in A, $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$ and $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$.

(c) If B has the form $B_1 \rightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and we have $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$, $A|_{\pi.2} \stackrel{\text{def}}{=} B_2$, $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$, $pol(A, \pi.2) \stackrel{\text{def}}{=} pol(A, \pi)$.

(d) If B has the form $B_1 \leftrightarrow B_2$, then for all $i \in \{1, 2\}$ the position $\pi.i$ is a position in A, $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$ and $pol(A, \pi.i) \stackrel{\text{def}}{=} 0$.

If $A|_\pi = B$, we also say that *B occurs in A at the position π*. If, in addition, $pol(A, \pi) = 1$ (respectively, $pol(A, \pi) = -1$), then we call the occurrence of B at the position π in A a *positive occurrence* (respectively, *negative occurrence*) of B in A. ❑

EXAMPLE 4.12 To illustrate these notions, consider the negation of the formula of Example 4.3 on page 42. This negation is $\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$ and it is unsatisfiable, see Figure 4.7. Denote this formula by A. The positions in A, subformulas at these positions, and the corresponding polarities are given in the table of Figure 4.10. ❑

position	subformula	polarity
ϵ	$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r))$	1
1	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \wedge (p \wedge q \to r)$	1
1.1.1	$p \to q$	1
1.1.1.1	p	-1
1.1.1.2	q	1
1.1.2	$p \wedge q \to r$	1
1.1.2.1	$p \wedge q$	-1
1.1.2.1.1	p	-1
1.1.2.1.2	q	-1
1.1.2.2	r	1
1.2	$p \to r$	-1
1.2.1	p	1
1.2.2	r	-1

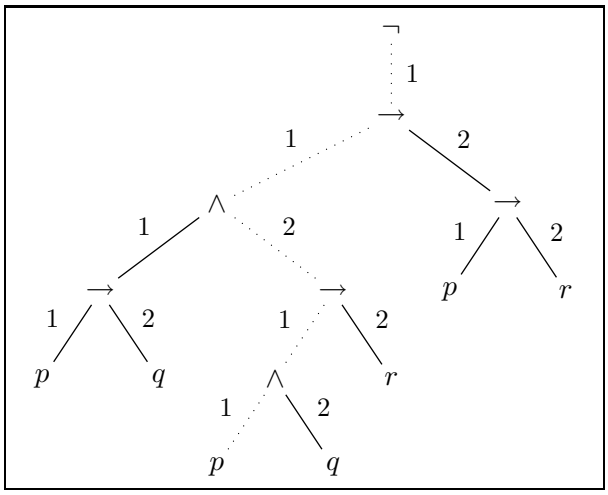


Figure 4.10: Positions and polarity

The notions of position and polarity are best illustrated when we draw the parse tree for the formula A, so that the nodes are labeled by connectives or atoms, see Figure 4.10. For each node in the parse tree, we label the arcs coming from this node by integers $1, 2, \ldots$. To find a position of a subformula B of A, we simply collect all integers on the path from the root to the node representing B. For example, the second occurrence of p is at the position $1.1.2.1.1$ (see the path denoted by the dotted arcs). When we depict the parse tree a formula, we can calculate the polarity of an occurrence of a subformula B in the following way. Consider the node n corresponding to this occurrence and the path from the root to this node. If any node on this path (excluding n itself) is labeled by $\leftrightarrow$, then the polarity is 0. Otherwise, count the number of arcs on the path such that either (i) the upper node of this arc is labeled with $\neg$; or (ii) the upper node of this arc is labeled with $\rightarrow$ and the arc itself is labeled by 1. If the total number of such paths is even, then the polarity is 1, otherwise it is -1.

In other words,

(1) if B occurs in the range of any equivalence $\leftrightarrow$, then the polarity of π is 0;

(2) if B does not occurs in the range of an equivalence $\leftrightarrow$, then the polarity of π is 1 (respectively, -1) if and only if B occurs in the range of a negation $\neg$ or left-hand side of an implication $\rightarrow$ an even (respectively, odd) number of times.

For example, the dotted path in Figure 4.10 contains one arc with $\neg$ on top, and two arcs having $\rightarrow$ on top and labeled with 1. The total number of such arcs is 3, so the second occurrence of p is negative.

The main property of polarities is that they provide us with a syntactic analogue of the notion of monotonicity. Note that the property $I(A) \leq I(B)$ can be expressed as $I \models A \rightarrow B$. We have the following result.

LEMMA 4.13 (Monotonic Replacement) *Let A, B, B' be formulas, I be an interpretation, and $I \models B \rightarrow B'$. If $pol(A, \pi) = 1$, then $I \models A[B]_\pi \rightarrow A[B']_\pi$. Likewise, if $pol(A, \pi) = -1$, then $I \models A[B']_\pi \rightarrow A[B]_\pi$.* ❑

This result implies the following theorem, which is similar to Equivalent Replacement Theorem 3.9.

THEOREM 4.14 (Monotonic Replacement) *Let A, B, B' be formulas such that $B \rightarrow B'$ is valid (respectively, $B' \rightarrow B$ is valid). Let a formula A' be obtained from A by replacing one or more positive (respectively, negative) occurrences of B by B'. Then $A \rightarrow A'$ is valid.* ❑

This theorem has a corollary related to satisfiability.

COROLLARY 4.15 *Let A, B, B' be formulas such that $B \rightarrow B'$ is valid (respectively, $B' \rightarrow B$ is valid). Let a formula A' be obtained from A by replacing one or more positive (respectively, negative) occurrences of B by B'. If A is satisfiable, then so is A'.* ❑

The notion of polarity has an important application to the propositional satisfiability problem. We say that an atom p is *pure* in a formula A, if either all occurrences of p in A are positive or all occurrences of p in A are negative.

LEMMA 4.16 (Pure Atom) *Let p be pure in A. Let $I \models A$ and I' be obtained from I as follows:*

$$I'(q) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } p = q \text{ and } p \text{ occurs in } A \text{ only positively}; \\ 0, & \text{if } p = q \text{ and } p \text{ occurs in } A \text{ only negatively}; \\ I(q), & \text{if } p \neq q. \end{cases}$$

Then $I' \models A$.

PROOF. We consider only the case when all occurrences of p in A are positive. The formula $p \to \top$ is, obviously, a tautology, so by Monotonic Replacement Theorem 4.14 the formula $A \to A_p^\top$ is a tautology, too. Since $I \models A$, we have $I \models A_p^\top$. Since p does not occur in $A_p^\top$ and I agrees with I' on all variables different from p, we also have $I' \models A_p^\top$. But by the construction of I' we have $I' \models p \leftrightarrow \top$, so by Equivalent Replacement Lemma 3.8 we have $I' \models A$. ❑

This lemma means that, when checking whether A is satisfiable, instead of considering two possible truth values for p, we can restrict ourselves by considering only one truth value, namely 1 if p has only positive occurrences and 0 if p has only negative occurrences. Of course this lemma is not applicable if p is not pure in A, i.e., it either has occurrences of polarity 0 or both positive and negative occurrences.

A syntactic analogue of the Pure Atom Lemma is as follows.

THEOREM 4.17 (Pure Atom) *Let an atom p has only positive (respectively, only negative) occurrences in A. Then A is satisfiable if and only if so is $A_p^\top$ (respectively, $A_p^\bot$).*

PROOF. ($\Rightarrow$) Since $p \to \top$ is tautology, by Monotonic Replacement Theorem 4.14 $A \to A_p^\top$ is a tautology too, so every model of A is also a model of $A_p^\top$.
($\Leftarrow$) Let I be a model of $A_p^\top$. Change the interpretation I into I' by redefining it on the variable p so that $I'(p) = 1$. Then I' is a model of $A_p^\top$, since I and I' agree on all variables of $A_p^\top$. But $I' \models p \leftrightarrow \top$, so by Equivalent Replacement Lemma 3.8 we have $I' \models A \leftrightarrow A_p^\top$, hence $I' \models A$ and so A is satisfiable. ❑

This theorem gives us an important optimization for the Splitting Algorithm. If an atom p is found to be pure in G, then we replace G by $G_p^\top$ (or $G_p^\bot$, depending on the polarity of p in G), thus avoiding any splits on p.

EXAMPLE 4.18 Take the formula of Example 4.12 and replace the rightmost occurrence of p in it by $\neg p$. We obtain the formula $\neg((p \to q) \wedge (p \wedge q \to r) \to (\neg p \to r))$. Let us check if this formula is satisfiable. All occurrences of p in this formula are negative. By

Pure Atom Theorem we can replace p by $\bot$. By simplifying the resulting formula using the rules of Figure 4.5, we obtain

$$\begin{array}{rl}
\neg((\bot \to q) \wedge (\bot \wedge q \to r) \to (\neg\bot \to r)) & \Rightarrow \\
\neg(\top \wedge (\bot \wedge q \to r) \to (\neg\bot \to r)) & \Rightarrow \\
\neg((\bot \wedge q \to r) \to (\neg\bot \to r)) & \Rightarrow \\
\neg((\bot \to r) \to (\neg\bot \to r)) & \Rightarrow \\
\neg(\top \to (\neg\bot \to r)) & \Rightarrow \\
\neg(\neg\bot \to r) & \Rightarrow \\
\neg(\top \to r) & \Rightarrow \\
\neg r. &
\end{array}$$

In the resulting formula the atom r is pure and has only negative occurrences, so we can safely substitute $\bot$ for r, obtaining the partial interpretation $\{p \mapsto 0, r \mapsto 0\}$ which satisfies the original formula. Note that we have found a satisfying interpretation with no splits at all, that is, completely deterministically. ❑

Exercises

EXERCISE 4.1 Explain how satisfiability, validity, and equivalence of propositional formulas can be expressed in terms of one another. ❑

EXERCISE 4.2 Build a truth table for each of the following formulas

$$\begin{array}{l}
p \to \neg r; \\
p \leftrightarrow (\neg r \to \neg p); \\
(p \leftrightarrow r) \leftrightarrow (p \leftrightarrow \neg r).
\end{array}$$

❑

EXERCISE 4.3 (★) Find a shortest formula equivalent to

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

Can you prove that this formula *is* the shortest one? ❑

EXERCISE 4.4 Apply the Splitting Algorithm to the formulas of Examples 4.12 and 4.18. In both cases split on the atom r first. ❑

EXERCISE 4.5 (►295◄) Check, using splitting, whether the formula $(p \leftrightarrow q) \wedge ((p \wedge \neg q) \vee (q \wedge \neg p))$ is satisfiable. Split on the atom p first. ❑

EXERCISE 4.6 (★) Prove that the set of rules of Figure 4.5 on page 46 for simplifying formulas is confluent (confluence is defined on page 11). ❑

EXERCISE 4.7 The pigeonhole problem is the following problem (or rather a family of problems for each value of the parameters k, m described below). There are k pigeons and m holes. Is it possible to assign pigeons to holes so that every pigeon occupies some hole and no hole contains more than one pigeon? Formalize this problem, for each pair k, m, as a propositional satisfiability problem. *Hint:* use the boolean variables $\{p_{i,j} | i = 1 \dots k, j = 1 \dots m\}$, where $p_{i,j}$ denotes that the pigeon i occupies the hole j. ❑

EXERCISE 4.8 Apply the Splitting Algorithm to establish the unsatisfiability of the pigeonhole problem for the case of three pigeons and two holes. ❑

EXERCISE 4.9 Make a table with all positions, the corresponding subformulas and polarities in each of the following formulas

$$p \to (\neg p \vee (q \leftrightarrow \neg q) \vee (q \to (r \wedge p))); \tag{4.1}$$
$$p \wedge \neg q \wedge ((q \leftrightarrow \neg q) \vee p \vee (q \to (r \vee p))). \tag{4.2}$$

(See Figure 4.10 on page 53 for an example.) ❑

EXERCISE 4.10 Consider the formula $(p \to q) \wedge (q \to \neg p)$. Is this formula

(1) monotonic, anti-monotonic, or neither on p?

(2) monotonic, anti-monotonic, or neither on q? ❑

EXERCISE 4.11 (►295◄) Consider the formula $(p \leftrightarrow q) \leftrightarrow p$. Is this formula

(1) monotonic, anti-monotonic, or neither on p?

(2) monotonic, anti-monotonic, or neither on q? ❑

EXERCISE 4.12 (★★) Prove, using splitting, the following property. Let A be a formula built from atoms using only $\neg$ and $\leftrightarrow$. Then A is unsatisfiable if and only if (1) each atom occurs in A an even number of times, and the negation $\neg$ occurs in A an odd number of times. ❑

EXERCISE 4.13 (★) Prove that for any unsatisfiable formula A of Exercise 4.12, every splitting tree has 2^n branches, where n is the number of distinct atoms in A. ❑

EXERCISE 4.14 Prove, using splitting, that

(1) the formulas $p \to q$ and $\neg q \to \neg p$ are equivalent;

(2) the formulas $p \leftrightarrow \neg q$ and $\neg p \leftrightarrow q$ are equivalent.

EXERCISE 4.15 (★) We know that checking validity is coNP-complete. Prove that the problem of determining whether a formula is monotonic on one of its variables is coNP-complete too. ❑

EXERCISE 4.16 Draw the parse tree for the formula $r \wedge \neg p \wedge q \to ((p \leftrightarrow \neg q) \to r)$ and mark the nodes corresponding to the negative occurrences of subformulas (e.g., encircle them). Write down all negative subformulas of this formula. ❑