Mini Mine Sweeper

Carlo Tomasi
Computer Science
Duke University

1 Introduction

Mine sweeper is a computer game in which the player starts from the top left square (start) of an array. Mines are hidden under some of the squares[1], and a player who walks on one of them loses the game. The goal of the game is to sweep, that is, click on, all of the squares with no mines. Squares on the frontier of the area the player has explored so far display the number of mines in unexplored, neighboring squares. Two squares are adjacent when they share a side or a vertex, so a square has up to eight other squares that are adjacent to it. Figure 1 shows the status of the array at some point during a game, through the display provided in the Microsoft Windows version of mine sweeper.

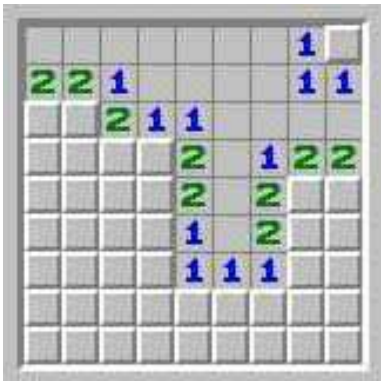

Figure 1: An intermediate stage in a game of minesweeper (Microsoft Windows version). The large area of non-embossed squares at the top has been explored so far. The squares with numbers are the frontier of the explored area, and are part of it. The number 2 in the square in the second row, first column means that there are two mines in the two unexplored squares that are adjacent to it. In this case, there is no uncertainty: walking onto any of those two squares results into horrible death. The number two in row 3, column 3 carries less information, because that square has three unexplored neighbors, and we do not know which two of these have mines.

Minesweeper is a simple game, and fun to play. We will make it trivial by the following two restrictions:

- The array has three rows and three columns.
- There is exactly one hidden mine.

Even in this embarrassingly simple version, *mini mine sweeper*, careful study of the game lets us explore several aspects of the use of logic in automated reasoning.

[1] But never on the start square.

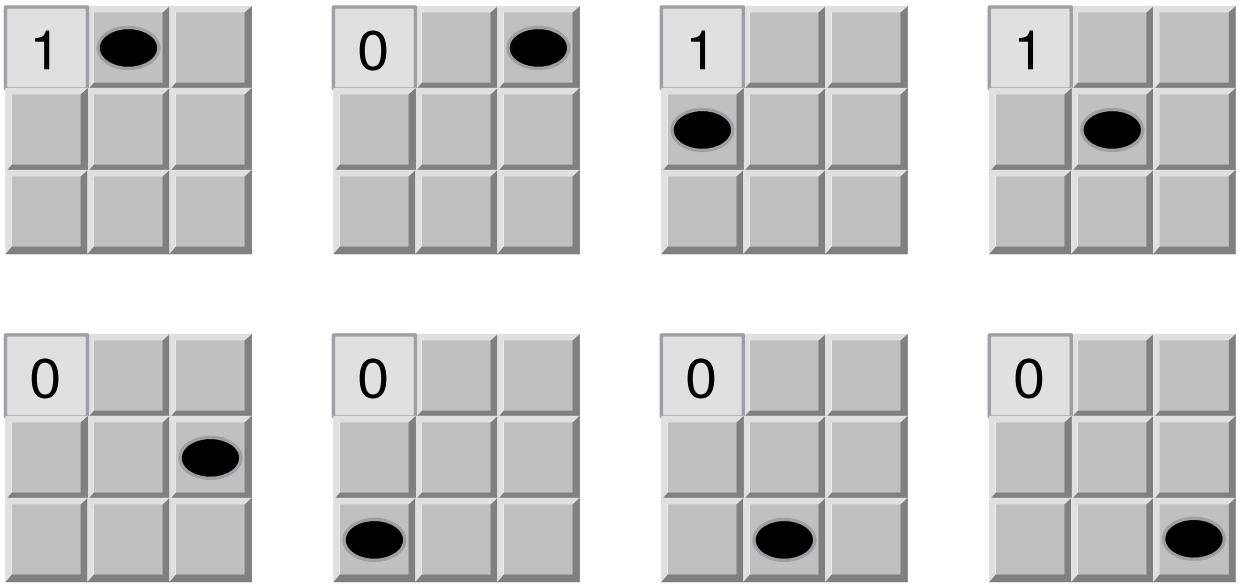

Figure 2: The eight possible models for the sentence “the player is on the first square” in the mini mine sweeper world. The black ellipse shows where the mine is in each model. Note that the player does not know where the mine is: a model describes a possible state of affairs of the world, not a state of knowledge (or belief) of the player.

2 Try Small, Think Big

Making mine sweeper so small lets us easily list all possible games exhaustively. However, all the ideas we explore can be generalized to the large version of the game, and indeed to a very large class of the kind of problems one often encounters in computer science.

So as we take this journey through mini mine sweeper, keep thinking how concepts scale up to bigger problems. The basic ideas often carry through unaltered, but creativity is needed to make them work in a reasonable amount of time.

3 World, States, and Models

Mini mine sweeper is our *world*, in that it contains all the objects, rules, and concepts we need to consider to talk about the game: a 3 by 3 array of squares, one mine, a start square, the frontier, the numbers that tell how many mines (0 or 1 in our case) are hidden in neighboring squares, the notion of adjacency, and so forth.

During a game, the world transitions through *states*. In the first state, the player is on the start square, and there is a mine lurking somewhere, although the player does not know where. Each move by the player brings about a *state transition.*

Facts about the world can be expressed by sentences in English. The world configurations in which the sentence is true are called *models* of that sentence. For instance, our world has eight models for the sentence “the player is on the first square.” These models are shown in Figure 2.

The notion of world and that of a model for a sentence come from the field of *semantics*. If you were to explain what it *means* for the player to be on the first square, you would have to come up with another sentence, such as ”the player has not moved yet” or ”the square in row 1, column 1 has been explored and the others have not” or something to this effect. This situation seems to exhibit some circularity: to explain sentences we use other sentences, and it is not clear where we can stop.

We took a safer alternative, and decided to “explain” the sentence by listing all possible states in which the sentence is the case in the given world. Our listing took the form of a drawing, that is, something that is not by itself a sentence,

and this prevents circularity: We assume that we somehow understand what a drawing means, *i. e.*, that a drawing need not be explained.

If you think that this is unsatisfactory, that to explain something we merely appeal to something else (drawings) that we leave unexplained, you are not alone: we have merely shifted the problem from sentences to drawings. We could have used something other than a drawing (a song, a finger pointing to an event outside our window, ...), but this is how *logic*, another field of study, operates: it tries to manipulate sentences about some world in a formal way, one that does not require knowing what sentences actually mean, and leaves the problem of interpretation to semantics, a separate discipline.

The really amazing fact is that the formal manipulation of logic, if carried out according to carefully designed rules, can actually lead to interesting conclusions. We will study this very puzzling fact (including understanding of when a "conclusion" is "interesting") through the study of inference in two types of logic: *propositional logic* and *first order predicate logic*.

Inference in propositional logic is simpler and smaller than in predicate logic, and is a good introduction to the main ideas. As we try to describe more and more complex worlds, inference in propositional logic becomes soon inadequate, and we will switch to the more powerful (but harder to deal with) inference techniques of predicate logic.

4 Inference in Propositional Logic

As a preview of what this all means, let us go through a typical manipulation in propositional logic, called *logical inference*. This is just another game, with symbols, strings, and rules for creating new strings out of old ones. We will only see a part of this. However, the complete game of logical inference is not much bigger in terms of the basic rules. What changes is only the number of moves, which is typically much bigger for "bigger" games.

In the inference game of propositional logic, we are given an initial bag of strings, called *axioms*. Let's say that our axioms are the following four strings, listed with commas separating them:

$$c,\ \neg d,\ b \to d,\ c \to a \vee b\ .$$

The period at the end is not part of the strings, but rather of the previous English sentence, starting with "Let's say..."

A set of rules let us add new strings to our bag. The rules are called *inference rules*, and the new strings are called *theorems*. Here are the rules we use in this preview. Again, we separate rules with commas:

$$\frac{S \to U}{\neg U \to \neg S},\ \frac{\begin{array}{c} S \\ S \to U \end{array}}{U},\ \frac{\begin{array}{c} S \vee U \\ \neg U \end{array}}{S}\ .$$

Let us call these rules rule 1, rule 2, rule 3. Rule 1 means "if you have a string in your bag whose format is $S \to U$ where S and U are any substrings, then you are allowed to add the string $\neg U \to \neg S$ to your bag, where S and U are the same substrings that you found in the string you took from the bag." For instance, if I have the string $a \wedge b \to c$ in my bag, then I am allowed to add the string $\neg c \to \neg(a \wedge b)$ to my bag if I wish to. Note that the role of S here is played by the string $a \wedge b$, which is called an *instantiation* of S. This instantiation is *composite*, that is, it is made of several symbols. Another rule states that if S in $\neg S$ is composite then we really need to write $\neg(S)$ instead of just $\neg S$.

Rule 2 is more demanding: in order to be allowed to add a string of the form U to the bag I need to already have *two* string in it: one is S (which can represent any string in the bag), and the other is of the form $S \to U$ (the S here and the original S must be the same). Rule 3 is similar.

The statement(s) above the line in an inference rule are called *precondition(s)*, and the statement(s) below the line are called *postcondition(s)*.

The game works by coming up with a new string, say

$$a$$

and looking for a series of applications of the rules that make the new string appear in the bag. This series is called a *proof*: if you come up with a proof for a, it becomes a theorem, and you win the game. Here is a proof for a. The first

row in the following table lists the initial contents of the bag (the axioms). In the subsequent rows, the first column shows the instantiation of the rule, the second show the rule number used, and the third column shows the contents of the bag after application of the rule. Theorems are underlined.

		$c,\ \neg d,\ b \to d,\ c \to a \vee b$
$\dfrac{c \quad c \to a \vee b}{a \vee b}$	Rule 2	$c,\ \neg d,\ b \to d,\ c \to a \vee b,\ \underline{a \vee b}$
$\dfrac{b \to d}{\neg d \to \neg b}$	Rule 1	$c,\ \neg d,\ b \to d,\ c \to a \vee b,\ \underline{a \vee b},\ \underline{\neg d \to \neg b}$
$\dfrac{\neg d \quad \neg d \to \neg b}{\neg b}$	Rule 2	$c,\ \neg d,\ b \to d,\ c \to a \vee b,\ \underline{a \vee b},\ \underline{\neg d \to \neg b},\ \underline{\neg b}$
$\dfrac{a \vee b \quad \neg b}{a}$	Rule 3	$c,\ \neg d,\ b \to d,\ c \to a \vee b,\ \underline{a \vee b},\ \underline{\neg d \to \neg b},\ \underline{\neg b},\ \underline{a}$

as desired. The contents of the bag never shrink: because of this, this form of game is called *monotonic* reasoning. Check carefully that the rules are always applied to strings that are already in the bag before the rule is applied.

This is what logic "sees" about inference and proofs: just a game of string manipulations, where the strings mean nothing. Now, what does the proof of "a" constructed through this game actually *mean*? To the game player, also called a *theorem prover*, nothing at all. This is just a game with strings and fixed rules.

We could however impose a semantics, that is, a meaning, on what is going on, by interpreting the symbols used as follows in the context of the mini mine sweeper game:

a	:	There is a mine at row 2, column 1
b	:	There is a mine either at row 2, column 2, or at row 2, column 3 (but not both)
c	:	There is a mine in one of the squares adjacent to the square at row 1, column 2
d	:	There is a mine in one of the squares adjacent to the square at row 1, column 3
$\neg S$	:	not S
$S \vee U$	:	S or U
$S \to U$	:	if S then U

If we translate the axioms and our final theorem (that is, "a") with these interpretations, we see that we proved the following:

> If there is a mine in one of the squares adjacent to the square at row 1, column 2 and there is not a mine in one of the squares adjacent to the square at row 1, column 3 and if whenever there is a mine at row 2, column 2, or at row 2, column 3 then we must conclude that there is a mine in one of the squares adjacent to the square at row 1, column 3 and if whenever there is a mine in one of the squares adjacent to the square at row 1, column 2 then we must conclude that there is a mine at row 2, column 1 or there is a mine at row 2, column 2, or there is a mine at row 2, column 3, then there is a mine at row 2, column 1.

Now this is a mouthful! We can summarize most of this with Figure 3 and the caption below it. What figure and caption do not show are two elements of "knowledge" about minesweeper, that is, the two facts $b \to d$ and $c \to a \vee b$. The first fact states that a mine that is in either one of the last two squares in row two is adjacent to the square at row

1, column 3 (an obvious fact of geometry). The second fact states that if there is a mine in a square adjacent to the square at row 1, column 2, then the mine must be in one of the squares of the second row. This fact is not generally true, but it is true in the situation in Figure 3, in which the entire first row has been explored (and therefore contains no mines). Please verify that these two facts, together with Figure 3 and its caption, are equivalent to the axioms from which we started the inference game.

Figure 3: In the situation in the figure, there must be a mine at row 2, column 1.

In the actual game, the square at row 1, column 1 would display a '1', because there is one mine in a square adjacent to this square. However, this fact is not used in the proof, so it was omitted from the figure.

The information summarized in Figure 3 is actually helpful for the player to decide what move to take next. We can say that our theorem prover analyzed the information available to the mini mine sweeper player at some intermediate stage in the game, and inferred the position of the hidden mine from it. This now allows the player to safely sweep the two squares in row 2 that can be deduced not to have a mine.

You get the idea, now: in order to prove the safety of squares in row 2 column 2, and in row 2, column 3, we need to add a statement [2] to our axioms that says that if there is a mine in the square at row 2, column 1, then there is no mine in any other square (a different way of saying that there is at most one mine anywhere in the array). This statement can be added before we start to play the game: we are not saying that all squares other than row 2, column 1 are safe, a fact that we could not know before we start to play. All we are saying is that *if* the square at row 2, column 1 has a mine, *then* all other squares are safe. We are only expressing an *implication*, that is, the fact that from one particular fact another particular fact must follow.

With only one mine in the array, our formal reasoning has made us win the game: we now know that the mine is at row 2, column 1 (fact "a"), and we can safely sweep the whole array, staying clear of the one mine.

In our proof we used axioms, theorems, and rules of inference. Axioms are given nuggets of truth, facts that are known to always hold (if there is a mine here, then there isn't one there) or facts that are known to hold for this particular game (there is one mine next to this particular square). Theorems are new statements derived from the axioms though rules of inference.

5 Soundness

Of course, for this game to work (that is, for it to have a useful interpretation through a given semantics) the rules of inference have to be chosen very carefully. Specifically, they must be such that if they are applied to true statements then the statements they produce are also true. Rules of inference that satisfy this requirement are said to be *sound* (*truth preserving* is a more descriptive synonym). As you can imagine, soundness is a very tall order: A sound inference rule must somehow capture a basic fact about reasoning.

[2] The terms "statement" and "proposition" are used as synonyms in this note.

Let us see how this works for one of the inference rules above, Rule 2:

$$\frac{\begin{array}{c} S \\ S \to U \end{array}}{U} \ .$$

This is one of the most well-known inference rules of logic, and dates back at least to Aristotle. It is called *modus ponens* ("the mode that affirms" in Latin).

The requirement that two statements S and $S \to U$ (recall that capital letters can stand for arbitrarily complex propositions) be in the bag of available statement when *modus ponens* is applied means that at that point the two statements are assumed to be both true. We also know that the implication $S \to U$ is false if and only if S is true and U is false. In other words, the truth table of $S \to U$ is as follows (T means "True" and F means "False"):

S	U	$S \to U$
T	T	T
T	F	F
F	T	T
F	F	T

.

The only row of this table in which both S and $S \to U$ (the first and *third* column, mind you) are true is the first one. From this table we see that in all circumstances in which S and $S \to U$ are both true U is true as well. So if we add the statement U to our bag we are adding something that is *consistent* with what's in the bag before we add U. That is, we add a new statement that does not contradict the old ones: we are adding a true statement. In this sense, *modus ponens* is truth-preserving, or sound.

Modus ponens has been explained in the past by appealing to intuition, and this is perhaps why this is such a widely used rule of inference. It is intuitively obvious that if S holds and if S implies U, then U must hold as well: if it rains, and if rain implies the existence of clouds in the sky, then there must be clouds in the sky. However, this may be a misleading line of thought, unless we specify unambiguously what "implies" means. In logic, the meaning of implication is completely captured by its truth table.

Interestingly, the soundness of *modus ponens* (and of any other inference rule, for that matter) can also be captured by a truth table: What *modus ponens* says is that if S and $S \to U$ are true, then U must be true. In logical symbols,

$$(S \wedge (S \to U)) \to U \ .$$

This compound statement is not true because of the truth values of its constituents. Rather, it is true because of its *structure*, that is, because of the way it is built: no matter what the truth value of S and U, or even of $S \to U$, or of $S \wedge (S \to U)$, the statement above is always true, that is, it is a *tautology*. This is a bit subtle: when we *use* the inference rule, we will need the fact that S is true (*i. e.*, that it is in our bag). However, the compound statement above is true even when S is false. Let us check this with truth tables:

S	U	$S \to U$	$S \wedge (S \to U)$	$(S \wedge (S \to U)) \to U$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

.

The last column of this table has only Ts in it: the statement in its header is true no matter what the truth value of either S or U (first two columns of the table) or of the values in any of the other columns is.

6 Logical Equivalence

Inference rules work only one way: if the current bag of statement contains what is above the line in an inference rule (the precondition(s)), then what is below it (the postcondition(s)) can be added to the bag. We cannot go the other way. For instance, if $\{S,\ S \to U\}$ is a subset of the statements in the current bag, then after applying *modus ponens* this subset is expanded to $\{S,\ S \to U,\ U\}$. As we saw at the end of the previous Section, this is also captured by the

tautology $(S \wedge (S \rightarrow U)) \rightarrow U$, which contains the implication symbol $\rightarrow$. We are not allowed to apply the inference rule the other way: if the bag contains U, we cannot add S and $S \rightarrow U$ to it.

If the arrow in the tautology went both ways ('$\leftrightarrow$' instead of '$\rightarrow$'), then of course we would be allowed to go either way. For instance, the biimplication

$$\neg(S \wedge U) \leftrightarrow \neg S \vee \neg U$$

is a tautology, as the following truth table verifies:

S	U	$\neg S$	$\neg U$	$S \wedge U$	$\neg(S \wedge U)$	$\neg S \vee \neg U$	$\neg(S \wedge U) \leftrightarrow \neg S \vee \neg U$
T	T	F	F	T	F	F	T
T	F	F	T	F	T	T	T
F	T	T	F	F	T	T	T
F	F	T	T	F	T	T	T

We would then be able to write one inference rule for each direction of the arrow:

$$\frac{\neg(S \wedge U)}{\neg S \vee \neg U} \quad \text{and} \quad \frac{\neg S \vee \neg U}{\neg(S \wedge U)} \ .$$

However, there is more. This pair of rules, taken together, is stronger than just a pair of inference rules, because the pair says that the two expressions (above and below the line in either inference) are *logically equivalent* to each other: their truth values are the same no matter what the truth values of their components S and U are, and since logic is only concerned with truth values there is no logical difference whatsoever between the two statements. This is important enough to deserve its own notation:

$$S \equiv U$$

means that statements S and U are logically equivalent, that is, that the biimplication $S \leftrightarrow U$ is a tautology. We can now use S and U interchangeably wherever they appear, even if one of the two is embedded within another expression.

Consider for instance the statement

$$p \rightarrow \neg(q \wedge r) \ .$$

An inference rule cannot be used to transform a piece of this statement: First, inference rules can only *add* statements, not change them. Second, they can only be applied if complete statements of the exact format specified in their precondition exist in the current bag. On the other hand, the logical equivalence

$$\neg(S \wedge U) \equiv \neg S \vee \neg U$$

can be used to transform just a part of the statement above to obtain the following statement:

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

(the parentheses here are redundant, because implication takes precedence over disjunction). Thus, logical equivalences can be thought of as "rewriting rules." The rule used above is one of *De Morgan's laws*. The other one is

$$\neg(S \vee U) \equiv \neg S \wedge \neg U \ .$$

As an exercise, build a truth table to check that this is indeed a logical equivalence, that is, check that $\neg(S \vee U) \leftrightarrow \neg S \wedge \neg U$ is a tautology.

7 Logical and Human Reasoning

Logic is often said to attempt to mimic human reasoning. Before delving into some of the basics of inference in propositional calculus, it is worth reflecting on this claim: In what way can we say that the string replacement game we played earlier "mimics" human reasoning? Perhaps the results of reasoning are the same: we and the theorem prover both somehow "know" that there is no mine at row 1, column 1, and so forth. We both also "know" many facts of the game: there is only one mine, one mine cannot be in two squares at once, *et cetera*. After some activity that could pass for "mental work" we both "deduce" that the mine is in the square at row 2 column 1.

However, it is plausible that the "mental work" in question is different for humans and formal theorem provers, and perhaps it may also be different for different humans (it is certainly different for different theorem provers). For instance, I reason in terms of pictures, and visualize different possible states of the mini mine sweeper array, looking for one that is consistent with the neighbor counts of Figure 3. I do not need to go systematically through all eight states in Figure 2: since there are squares with nonzero neighbor counts in the first array row in Figure 3, and since I have already explored the first row completely, I somehow know that the mine must be in row 2. Did I use propositional logic to infer this? Perhaps, or maybe not.

The main point of inference in propositional logic (and indeed all of logic) is to reason without any reference to what the strings it manipulates actually mean: no use of semantics is made within logic. A naive computer program that were to attempt proofs such as the one seen earlier would cycle through all sentences in the bag, apply all applicable inference rules to them, and then repeat, in the hope of sometimes stumbling across the desired theorem ("a"). Smarter code may come up with clever shortcuts, but no theorem prover ever asks what the symbols mean. It is exactly this independence of semantics that confers generality to the process of logical inference. Inference is valid because of its rules (its "structure"), not because of the statements it is applied to.

As a model for human thought, this process may feel a bit too ungrounded in reality, or devoid of whatever we call "intuition." And yet, given how fast computers are, can a computer program not at least *appear* to be intelligent by more or less blindly but systematically and very quickly applying inference rules? And if the program reaches the same conclusions as human thinkers do (and perhaps more of them, or faster, or with fewer mistakes), can we really still say that the computer program merely *appears* to be intelligent? I know that you are intelligent because of the results of your thinking and not so much because of its process, of which I am completely, and you are partially, unaware.

8 Automating Mini Minesweeper

It is instructive to try and list a set of axioms that would let us play mini minesweeper in all cases, rather than just in the situation of Figure 3. The main outcome of this is mixed: we can do it, but the endeavor is laborious, and unlikely to scale to more interesting problems, and even to bigger versions (more squares and more mines) of mini minesweeper.

First, we need to say that there is exactly one mine somewhere, and not on the start square. To say that there is *at least* one mine is easy. If $m_{11}, m_{12}, m_{13}, m_{21}, m_{22}, m_{23}, m_{31}, m_{32}, m_{33}$ are nine propositions that mean "there is a mine in the square at row 1, column 1" and so forth, then the disjunction

$$\text{Existence Axiom:} \quad m_{12} \vee m_{13} \vee m_{21} \vee m_{22} \vee m_{23} \vee m_{31} \vee m_{32} \vee m_{33}$$

states that there is a mine somewhere on the array (and does not say anything about the start square). To express the fact that there is exactly one mine and none in the start square requires nine more axioms:

$$\begin{aligned}
&\neg m_{11} \\
m_{12} &\rightarrow \neg(m_{13} \vee m_{21} \vee m_{22} \vee m_{23} \vee m_{31} \vee m_{32} \vee m_{33}) \\
m_{13} &\rightarrow \neg(m_{12} \vee m_{21} \vee m_{22} \vee m_{23} \vee m_{31} \vee m_{32} \vee m_{33}) \\
m_{21} &\rightarrow \neg(m_{12} \vee m_{13} \vee m_{22} \vee m_{23} \vee m_{31} \vee m_{32} \vee m_{33}) \\
m_{22} &\rightarrow \neg(m_{12} \vee m_{13} \vee m_{21} \vee m_{23} \vee m_{31} \vee m_{32} \vee m_{33}) \\
m_{23} &\rightarrow \neg(m_{12} \vee m_{13} \vee m_{21} \vee m_{22} \vee m_{31} \vee m_{32} \vee m_{33}) \\
m_{31} &\rightarrow \neg(m_{12} \vee m_{13} \vee m_{21} \vee m_{22} \vee m_{23} \vee m_{32} \vee m_{33}) \\
m_{32} &\rightarrow \neg(m_{12} \vee m_{13} \vee m_{21} \vee m_{22} \vee m_{23} \vee m_{31} \vee m_{33}) \\
m_{33} &\rightarrow \neg(m_{12} \vee m_{13} \vee m_{21} \vee m_{22} \vee m_{23} \vee m_{31} \vee m_{32})
\end{aligned}$$

We then need to describe adjacency: there is a mine in the square at row 1, column 2 if and only if the following two facts are true. First, there is a mine adjacent to each of the following squares: row 1, column 1; row 1, column 3; row 2, column 1; row 2, column 2; row 2, column 3. Second, there are zero mines adjacent to each of the following squares: row 3, column 1; row 3, column 2; row 3, column 3.

There are eight such statements. If a_{11} means "there is a mine adjacent to the square at row 1, column 1," together with eight more analogous propositions, then the following eight axioms express adjacency:

$$\begin{aligned}
m_{12} &\leftrightarrow (a_{11} \wedge a_{13} \wedge a_{21} \wedge a_{22} \wedge a_{23} \wedge \neg a_{31} \wedge \neg a_{32} \wedge \neg a_{33}) \\
m_{13} &\leftrightarrow (\neg a_{11} \wedge a_{12} \wedge \neg a_{21} \wedge a_{22} \wedge a_{23} \wedge \neg a_{31} \wedge \neg a_{32} \wedge \neg a_{33}) \\
m_{21} &\leftrightarrow (a_{11} \wedge a_{12} \wedge \neg a_{13} \wedge a_{22} \wedge \neg a_{23} \wedge a_{31} \wedge a_{32} \wedge \neg a_{33}) \\
m_{22} &\leftrightarrow (a_{11} \wedge a_{12} \wedge a_{13} \wedge a_{21} \wedge a_{22} \wedge a_{23} \wedge a_{31} \wedge a_{32} \wedge a_{33}) \\
m_{23} &\leftrightarrow (\neg a_{11} \wedge a_{12} \wedge a_{13} \wedge \neg a_{21} \wedge a_{22} \wedge \neg a_{31} \wedge a_{32} \wedge a_{33}) \\
m_{31} &\leftrightarrow (\neg a_{11} \wedge \neg a_{12} \wedge \neg a_{13} \wedge a_{21} \wedge a_{22} \wedge \neg a_{23} \wedge a_{32} \wedge \neg a_{33}) \\
m_{32} &\leftrightarrow (\neg a_{11} \wedge \neg a_{12} \wedge \neg a_{13} \wedge a_{21} \wedge a_{22} \wedge a_{23} \wedge a_{31} \wedge a_{33}) \\
m_{33} &\leftrightarrow (\neg a_{11} \wedge \neg a_{12} \wedge \neg a_{13} \wedge \neg a_{21} \wedge a_{22} \wedge a_{23} \wedge \neg a_{31} \wedge a_{32})
\end{aligned}$$

A moment's reflection will show that the collection of these eighteen statements capture all we know about mini minesweeper. What changes from one game to the next is the adjacency information for the start square. In some situations (more precisely, in five of the possible initial models of the world), the axiom

$$\neg a_{11}$$

holds. In other cases (the remaining three models of the world), the opposite axiom holds instead:

$$a_{11} \ .$$

In the first case, inference ought to let us conclude that it is safe to sweep all of the neighbors of the start square. In the second case, we know that there is a mine in one of the three neighbors of the start square, but we do not know which, and we have one chance out of three to blow up at the first move.

This fact shows us that minesweeper (in its "mini" as well as in its full version) is not just a game of logical reasoning: if a_{11} is true, all we can do is to pick one of the neighbors of the start square, and hope that it does not contain a mine. How do we model this in our inference system? Suppose that upon seeing the axiom a_{11} we somehow manage to infer $m_{12} \vee m_{21} \vee m_{22}$ (more on how to do this later): the mine is in one of the squares adjacent to the start square. Since logic does not take us any further, we simply pick one of the three propositions m_{12}, m_{21}, m_{22}, say, m_{21}, and we ask the game keeper whether this is true or false. In other words, we admit that we do not have enough axioms to proceed, and we ask for one of the game keeper (the computer program, for instance). Of course, this is a dangerous action, because if the answer is "m_{21} is true," then we have lost the game. This business of asking for new axioms is of course not a part of the logical inference process, but just a rule of the game of minesweeper.

Returning now to logical inference, what constitutes a sufficient list of inference rules and logical equivalences? It is easiest to think of "general purpose" rules, that are likely to be useful in many situations. First, it is convenient to have "rewriting rules" (logical equivalences) that rearrange conjunctions or disjunctions, cancel double negations, and switch between conjunction and disjunction according to De Morgan's laws seen earlier:

E.1: $S \wedge U \leftrightarrow U \wedge S$, E.2: $\neg\neg S \leftrightarrow S$, E.3: $\neg S \vee \neg U \leftrightarrow \neg(S \wedge U)$, E.4: $\neg S \wedge \neg U \leftrightarrow \neg(S \vee U)$

(logical equivalence rules are labelled E.x for later reference). Then, we need inference rules that break down each of the adjacency relationships into two implications:

$$\text{I.1:} \ \frac{S \leftrightarrow U}{\begin{array}{c} S \rightarrow U \\ U \rightarrow S \end{array}} \ ,$$

take apart their right-hand sides:

$$\text{I.2:} \ \frac{S \rightarrow (U \wedge W)}{\begin{array}{c} S \rightarrow U \\ S \rightarrow W \end{array}} \ ,$$

and eliminate true propositions from a conjunction:

$$\text{I.3:}\quad \frac{\neg S}{\dfrac{S \vee U}{U}}\ .$$

(check that these inference rules are all sound). Inference rules are labelled I.x for later reference.

With these rules, we can take any actual adjacency snapshot at any stage of the game, and reduce the axioms to statements that only refer to the adjacency variables whose values are known.

We need one more inference rule that is more specific to minesweeper, namely, a way to rule out the presence of a mine if any one of the corresponding adjacency statements is violated:

$$\text{I.4:}\quad \frac{\neg U}{\dfrac{S \rightarrow U}{\neg S}}\ .$$

As a glimpse of how inference works, consider the situation in Figure 3. In our previous discussion, we had stated a few *ad hoc* axioms, and shown how to reason from there. How can we handle the same situation starting from our new, more general axioms and equivalence and inference rules?

In Figure 3, we know that a_{12} is true and a_{13} is false (and also that a_{11} is true, although we did not use this fact in our previous reasoning), but we know nothing direct about the values of the other adjacency variables. In this case, we can reason by exclusion. We start from the existence axiom, which says that the mine must be somewhere other than the start square:

$$m_{12} \vee m_{13} \vee m_{21} \vee m_{22} \vee m_{23} \vee m_{31} \vee m_{32} \vee m_{33}\ .$$

Then, we take each of the adjacency axioms in whose right-hand sides at least one of the known adjacency values a_{12} and $\neg a_{13}$ is violated. These are the axioms whose left-hand side contain $m_{22}, m_{23}, m_{31}, m_{32}, m_{33}$. From the elimination rule, we can infer that all these variables are false, and eliminate them from the existence axiom. In addition, since we are at this stage, we must have previously proven (or have been told by the game keeper) that both m_{12} and m_{13} are false. We are now left with

$$m_{21}$$

which tells us where the mine is. From the axiom $m_{21} \rightarrow \neg(m_{12} \vee m_{13} \vee m_{22} \vee m_{23} \vee m_{31} \vee m_{32} \vee m_{33})$ we can now infer that all other squares are safe, and we have won the game.

This high-level view of a proof strategy needs to be fleshed out in detail given the available axioms and inference rules. It is generally useful to first understand or develop a proof in its outline, and then fill in the details: thinking of the overall structure while also crossing all the tees and dotting all the eyes is very difficult, and it becomes easier to lose sight of where we want the proof to go. Something similar happens when you are planning a trip, say from Durham to Monticello, VA. You first determine where Monticello is relative to Durham, then you find one or two major roads that connect the general areas of the two locations, then you plan a path from where you are to the major roads and from there to Jefferson's house in Monticello. If you tried to plan a way out of your house first, you would soon be lost, unless you know the general direction in which to travel.

The nitty-gritty is more tedious. As an example, the following steps eliminate m_{31} from the disjunction

$$m_{12} \vee m_{13} \vee m_{21} \vee m_{22} \vee m_{23} \vee m_{31} \vee m_{32} \vee m_{33}$$

to obtain the following intermediate theorem (an intermediate theorem is also called a *lemma*):
Lemma: Given the mini mine sweeper axioms and a_{12}, the following holds:

$$m_{12} \vee m_{13} \vee m_{21} \vee m_{22} \vee m_{23} \vee m_{32} \vee m_{33}\ .$$

Proof:

Inference rule I.1 applied to the following adjacency axiom

$$m_{31} \leftrightarrow (\neg a_{11} \wedge \neg a_{12} \wedge \neg a_{13} \wedge a_{21} \wedge a_{22} \wedge \neg a_{23} \wedge a_{32} \wedge \neg a_{33})$$

transforms a biimplication into two separate implications:

$$\begin{aligned} m_{31} &\rightarrow (\neg a_{11} \wedge \neg a_{12} \wedge \neg a_{13} \wedge a_{21} \wedge a_{22} \wedge \neg a_{23} \wedge a_{32} \wedge \neg a_{33}) \\ m_{31} &\leftarrow (\neg a_{11} \wedge \neg a_{12} \wedge \neg a_{13} \wedge a_{21} \wedge a_{22} \wedge \neg a_{23} \wedge a_{32} \wedge \neg a_{33}) \ . \end{aligned}$$

The first of these two statement together with inference rule I.2 yields

$$\begin{aligned} m_{31} &\rightarrow \neg a_{11} \\ m_{31} &\rightarrow (\neg a_{12} \wedge \neg a_{13} \wedge a_{21} \wedge a_{22} \wedge \neg a_{23} \wedge a_{32} \wedge \neg a_{33}) \ . \end{aligned}$$

Applying inference rule I.2 again to the second of these expressions yields

$$\begin{aligned} m_{31} &\rightarrow \neg a_{12} \\ m_{31} &\rightarrow (\neg a_{13} \wedge a_{21} \wedge a_{22} \wedge \neg a_{23} \wedge a_{32} \wedge \neg a_{33}) \ . \end{aligned} \tag{1}$$

This isolates the implication (1) from the rest of the adjacency axiom. From a_{12} and equivalence E.2 we obtain

$$\neg\neg a_{12}$$

which together with (1) and inference I.4 yields

$$\neg m_{31} \ . \tag{2}$$

Equivalence rule E.1 applied to the existence axiom

$$m_{12} \vee m_{13} \vee m_{21} \vee m_{22} \vee m_{23} \vee m_{31} \vee m_{32} \vee m_{33}$$

(with the fifth '$\vee$' playing the role of the same symbol in E.1) yields

$$m_{31} \vee m_{32} \vee m_{33} \vee m_{12} \vee m_{13} \vee m_{21} \vee m_{22} \vee m_{23}$$

and this brings m_{31} to the front of the expression. Applying inference rule I.3 to (2) and the latter expression yields

$$m_{32} \vee m_{33} \vee m_{12} \vee m_{13} \vee m_{21} \vee m_{22} \vee m_{23}$$

in which m_{31} has disappeared. Finally, equivalence E.1 applied to the previous expression (with the second '$\vee$' playing the key role) rearranges the disjunction as follows:

$$m_{12} \vee m_{13} \vee m_{21} \vee m_{22} \vee m_{23} \vee m_{32} \vee m_{33} \ .$$

This is the desired statement. Δ

The Δ on the last line marks the end of the proof. Other proofs are of course possible. Chipping away each m_{ij} from this lemma in a similar way will eventually lead to the desired theorem

$$m_{21} \ .$$

The complete proof of this theorem is left as an exercise, if you have the patience for it. Of course, doing this by hand is extremely tedious, but a theorem prover written in software could automate the process. The difficult part of this task is to implement strategies that find the proper sequence of statements and rules to apply. This is the goal of the branch of artificial intelligence called "automated theorem proving" or "automatic logical reasoning."

9 Limits of Propositional Logic

Mini minesweeper can be automated with a relatively modest number of axioms and inference rules (even fewer than were listed earlier, if one pays more attention to efficiency). However, the number of axioms and rules needed for more complex reasoning quickly escalates beyond what is practical if only propositional logic is available.

Even a more realistic game of minesweeper gets soon out of control. If for instance there is a 9×9 array with ten mines on it (the default beginner version in Microsoft Windows), then our nine axioms that express the possible positions for a single mine are replaced by

$$\left(\begin{array}{c} 80 \\ 10 \end{array} \right) \approx 1.6 \times 10^{12}$$

analogous statements, one for each way to pick ten possible mine positions out of 80 squares (the start square is disallowed). That is more than one trillion axioms! It would be very hard to store these, let alone work with them. The situation with adjacency axioms is even more hopeless.

More fundamentally, propositional logic does not allow situations with an infinite or even just unknown number of possibilities. This problem becomes immediately apparent in games or other problems that involve time. In minesweeper, the game evolves monotonically, in that it is only possible to sweep new squares (we cannot "unsweep" a square we have visited). If you think of a game like dungeons and dragons, or some other exploration game, it becomes important to keep track of whether you have been through a room before or not, or of how many arrows you have in your quiver. What happens in response to some action will be different depending on whether you are in some room on your way to defeating a wizard, or on your way back. If we need a time variable to distinguish what happens in different circumstances, then we would need to replicate every possible proposition for every possible value of the time variable. This is not just hard, but typically impossible, because we do not know ahead of time how long a game will last.

The main problem with propositional logic is that its basic "sentences" are individual symbols devoid of any structure. For instance, something as complex as "there are mines in three of the squares adjacent to the square in row 4, column 7" is represented as a single symbol in propositional logic. A very closely related sentence, "there is a mine in one of the squares adjacent to the square in row 5, column 2," is represented by a different symbol, and the inference mechanism is completely unaware of the similarity of meaning of these two symbols. Please do not be misled by the fact that in the examples in the previous section similar sentences had similar symbols, such as m_{12} and m_{31}. This similarity was for our own convenience, and logical inference made no use whatsoever of this similarity.

We clearly need a more powerful formalism, one that lets us consider an infinite amount of possibilities with simple, concise sentences. In this formalism, we should be able to describe similar configurations (such as the adjacency information for different parts of the minesweeper board) with the same sentence but different "parameters," rather than with one sentence for each possible board configuration.

The formalism we need is called *predicate logic*. Its "parameters" are called *variables*, and its sentences are called *predicates*. Similar situations are described by the same predicate with different values for its variables. The positive result is that the representation of interesting worlds becomes concise. The negative result is that inference becomes harder.

10 Use of Predicate Logic

We can describe all situations in minesweeper if we can talk about where mines are (and where they are not), and how many mines are adjacent to what square. In fact, we wrote eighteen propositions that capture all we need to know for mini minesweeper. Let us try to summarize these into fewer, more concise English statements. Existence can be expressed thus:

> There is a mine in some square, but not in the start square.

Uniqueness can be stated as follows:

> If the mine is in one square, then it is not in any other.

Adjacency is captured by the following sentence:

> If there is a mine at row i, column j, then the mine is adjacent to every square whose row and column indices differ by at most one from i and j respectively, but not to square i, j itself.

To formalize these, we first construct the two *predicates* $M(i,j)$ and $A(i,j)$ in the *variables* i and j. The first one means "there is a mine in the square at row i and column j." The second means "there is a mine in a square adjacent to the square at row i and column j. We also need to specify what values i and j can possibly take, that is, their *domain*. The domain of both of them is the set $\{1,2,3\}$.

By themselves, these predicates are neither true nor false. It is not just that we do not know if at any point in the game they are true or false. The problem is more serious: asking whether, say, $M(i,j)$ is true or false makes no sense, unless we know the values of the variables i and j that the predicates refer to. We can specify these values, that is, we can *bind* the variables, in one of three ways:

Instantiation: Each variable is replaced with a value in its domain. For instance, $M(2,1)$ is a particular instantiation of $M(i,j)$. We may still not know the truth value of $M(2,1)$, but the question makes sense: $M(2,1)$ is either true or false: either there is a mine in the square at row 2, column 1, or there is none.

Universal Quantification: We assert that the predicate must hold for *all* values of the variable. We write

$$\forall i\ \forall j\ M(i,j)$$

and read "for all i and for all j, $M(i,j)$ holds." This statement is defined to be equivalent to the conjunction $M(1,1) \wedge M(1,2) \wedge M(1,3) \wedge M(2,1) \wedge M(2,2) \wedge M(2,3) \wedge M(3,1) \wedge M(3,2) \wedge M(3,3)$. Again, the question of truth makes sense. In this case, we know that this statement is false no matter what: it is never true that there is a mine in every square of the array.

Existential Quantification: We assert that there exists a set of values for the variables for which the predicate holds (we need not know what these values are). We write

$$\exists i\ \exists j\ M(i,j)$$

and read "there exists i and there exists j such that $M(i,j)$ holds." This statement is defined to be equivalent to the disjunction $M(1,1) \vee M(1,2) \vee M(1,3) \vee M(2,1) \vee M(2,2) \vee M(2,3) \vee M(3,1) \vee M(3,2) \vee M(3,3)$. We can again ask the question of truth. In this case, we know that this statement is true no matter what: it is always true that there is a mine in some square of the array.

Once all the variables in a predicate are bound in one of these ways, the question of truth or falsehood becomes meaningful, so the predicate becomes a *proposition*.

Our knowledge about the minesweeper game is then expressed by the following propositions:

$$\exists i\ \exists j\ (M(i,j) \wedge (i \neq 1 \vee j \neq 1))$$
$$\forall i\ \forall j\ \forall i'\ \forall j'\ (M(i,j) \wedge (i \neq i' \vee j \neq j')) \rightarrow \neg M(i',j')$$
$$\forall i\ \forall j\ \forall i'\ \forall j'\ (M(i,j) \wedge (|i-i'| \leq 1) \wedge (|j-j'| \leq 1) \wedge \neg(i = i' \wedge j = j')) \rightarrow A(i',j')$$
$$\forall i'\ \forall j'\ (\forall i\ \forall j\ (|i-i'| \leq 1) \wedge (|j-j'| \leq 1) \wedge \neg(i = i' \wedge j = j') \rightarrow \neg M(i,j)) \rightarrow \neg A(i',j')$$

Let us read them in English:

- There is a number i and a number j in their respective domains such that there is a mine at row i and column j and i is not equal to 1 or j is not equal to 1.
- For every choice of numbers i, j, i', j' in their respective domains, if there is a mine in row i, column j, and either i is different from i' or j is different from j' (or possibly both), then there is no mine at row i', column j'.
- For every choice of numbers i, j, i', j' in their respective domains, if there is a mine in row i, column j, and i differs from i' by at most one unit, and j differs from j' by at most one unit, and either i is different from i' or j is different from j' (or possibly both), then there is a mine in a square adjacent to the square at row i', column j'.

- For every choice of numbers i', j' in their respective domains, assume that for every choice of numbers i, j in their respective domains the fact that i differs from i' by at most one unit and j differs from j' by at most one unit implies that there are no mines at row i, column j. Then, under this assumption, there is no mine adjacent to the square at row i', column j'.

The last two sentences spell out the meaning of the predicate A and its negation $\neg A$.

It is now straightforward to check that these propositions are equivalent to the eighteen propositions given earlier. The interesting fact is that the same three propositions describe minesweeper even with a bigger array. All we need to change is the domain of the variables.

If we want to allow for K mines instead of 1 (where K is a fixed, known positive integer), the statements become somewhat more involved. For notational simplicity, we only consider the case $K = 2$ here. One complication is that the adjacency predicate $A(i, j)$ now needs to be replaced with a ternary predicate $A(i, j, k)$ where the domain of k is $\{0, 1, 2\}$, and whose meaning is "there are mines in k of the squares adjacent to the square at row i, column k. One separate axiom then must be given for each of the three possible values of k. We then have the following set of axioms:

$$\exists i_1\ \exists j_1\ \exists i_2\ \exists j_2\ (M(i_1, j_1) \wedge M(i_2, j_2) \wedge (i_1 \neq 1 \vee j_1 \neq 1) \wedge (i_2 \neq 1 \vee j_2 \neq 1) \wedge (i_1 \neq i_2 \vee j_1 \neq j_2))$$

$$\forall i_1\ \forall j_1\ \forall i_2\ \forall j_2\ \forall i'\ \forall j'\ (M(i_1, j_1) \wedge M(i_2, j_2) \wedge (i_1 \neq i' \vee j_1 \neq j') \wedge (i_2 \neq i' \vee j_2 \neq j')) \rightarrow \neg M(i', j')$$

$$\forall i'\ \forall j'\ (\forall i\ \forall j\ (|i - i'| \leq 1) \wedge (|j - j'| \leq 1) \wedge \neg(i = i' \wedge j = j') \rightarrow \neg M(i, j)) \rightarrow A(i', j', 0)$$

$$\forall i_1\ \forall j_1\ \forall i'\ \forall j'\ (M(i_1, j_1) \wedge (|i_1 - i'| \leq 1) \wedge (|j_1 - j'| \leq 1) \wedge \neg(i_1 = i' \wedge j_1 = j') \wedge$$
$$(\forall i_2\ \forall j_2\ ((|i_1 - i'| \leq 1) \wedge (|j_1 - j'| \leq 1) \wedge (i_1 \neq i_2 \vee j_1 \neq j_2)) \rightarrow M(i_2, j_2))) \rightarrow A(i', j', 1)$$

$$\forall i_1\ \forall j_1\ \forall i_2\ \forall j_2\ \forall i'\ \forall j'\ (M(i_1, j_1) \wedge M(i_2, j_2) \wedge (|i_1 - i'| \leq 1) \wedge (|j_1 - j'| \leq 1)$$
$$\wedge(|i_2 - i'| \leq 1) \wedge (|j_2 - j'| \leq 1) \wedge \neg(i_1 = i' \wedge j_1 = j') \wedge \neg(i_2 = i' \wedge j_2 = j') \wedge \neg(i_1 = i_2 \wedge j_1 = j_2))$$
$$\rightarrow A(i', j', 2)$$

Conciseness has come at the price of more involved inference rules. These are of the same type of the rules used in propositional logic, but must account properly for the quantifiers. This is discussed in Section 1.5 of the textbook.