**Logic**

This section summarizes the fundamental concepts of logical representation and reasoning.

These beautiful ideas are independent of any of logic’s particular forms. We therefore postpone the technical details of those forms until the next section, using instead the familiar example of ordinary arithmetic. Knowledge bases consist of sentences. These sentences are expressed according to the **syntax** of the representation language, which specifies all the sentences that are well formed. The notion of syntax is clear enough in ordinary arithmetic:

“x + y = 4” is a well-formed sentence, whereas “x4y+ =” is not.

A logic must also define the **semantics** or meaning of sentences. The semantics defines the **truth** of each sentence with respect to each **possible world**. For example, the semantics for arithmetic specifies that the sentence “x + y =4” is true in a world where x is 2 and y is 2, but false in a world where x is 1 and y is 1. In standard logics, every sentence must be either true or false in each possible world—there is no “in between.” When we need to be precise, we use the term **model** in place of “possible world.” Whereas possible worlds might be thought of as (potentially) real environments that the agent might or might not be in, models are mathematical abstractions, each of which simply fixes the truth or falsehood of every relevant sentence. Informally, we may think of a possible world as, for example, having x men and y women sitting at a table playing bridge, and the sentence x + y =4 is true when there are four people in total. Formally, the possible models are just all possible assignments of real numbers to the variables x and y. Each such assignment fixes the truth of any sentence of arithmetic whose variables are x and y. If a sentence α is true in model m, we say that m **satisfies** α or sometimes m **is a model of** α. We use the notation M(α) to mean the set of all models of α.

Now that we have a notion of truth, we are ready to talk about logical reasoning. This involves the relation of logical **entailment** between sentences—the idea that a sentence *follows*

*logically* from another sentence. In mathematical notation, we write .

to mean that the sentence α entails the sentence β. The formal definition of entailment is this:

α |= β if and only if, in every model in which α is true, β is also true. Using the notation just

introduced, we can write 

(Note the direction of the ⊆ here: if α |= β, then α is a *stronger* assertion than β: it rules out *more* possible worlds.) The relation of entailment is familiar from arithmetic; we are happy

with the idea that the sentence x = 0 entails the sentence xy = 0. Obviously, in any model where x is zero, it is the case that xy is zero (regardless of the value of y). We can apply the same kind of analysis to the wumpus-world reasoning example given in the preceding section. Consider the situation in Figure 7.3(b): the agent has detected nothing in [1,1] and a breeze in [2,1]. These percepts, combined with the agent’s knowledge of the rules of the wumpus world, constitute the KB. The agent is interested (among other things) in whether the adjacent squares [1,2], [2,2], and [3,1] contain pits. Each of the three squares might or might not contain a pit, so (for the purposes of this example) there are 23=8 possible models.

The KB can be thought of as a set of sentences or as a single sentence that asserts all the individual sentences. The KB is false in models that contradict what the agent knows — for example, the KB is false in any model in which [1,2] contains a pit, because there is no breeze in [1,1]. There are in fact just three models in which the KB is true, and these are shown surrounded by a solid line in Figure 7.5. Now let us consider two possible conclusions: α1 = “There is no pit in [1,2].” α2 = “There is no pit in [2,2].”

We have surrounded the models of α1 and α2 with dotted lines in Figures 7.5(a) and 7.5(b),

respectively. By inspection, we see the following: in every model in which KB is true, α1 is also true.

Hence, KB |= α1: there is no pit in [1,2]. We can also see that in some models in which KB is true, α2 is false.

Hence, KB α2: the agent *cannot* conclude that there is no pit in [2,2]. (Nor can it conclude that there *is* a pit in [2,2].)



The preceding example not only illustrates entailment but also shows how the definition

of entailment can be applied to derive conclusions—that is, to carry out **logical inference**.

The inference algorithm illustrated in Figure 7.5 is called **model checking**, because it enumerates all possible models to check that α is true in all models in which KB is true, that is,

that M(KB) ⊆ M(α). In understanding entailment and inference, it might help to think of the set of all consequences of KB as a haystack and of α as a needle. Entailment is like the needle being in the haystack; inference is like finding it. This distinction is embodied in some formal notation: if an inference algorithm i can derive α from KB, we write which is pronounced “α is derived from KB by i” or “i derives α from KB.” An inference algorithm that derives only entailed sentences is called **sound** or **truth preserving**. Soundness is a highly desirable property. An unsound inference procedure essentiallymakes things up as it goes along—it announces the discovery of nonexistent needles.It is easy to see that model checking, when it is applicable,4 is a sound procedure.The property of **completeness** is also desirable: an inference algorithm is complete ifit can derive any sentence that is entailed. For real haystacks, which are finite in extent,it seems obvious that a systematic examination can always decide whether the needle is inthe haystack. For many knowledge bases, however, the haystack of consequences is infinite,and completeness becomes an important issue. Fortunately, there are complete inferenceprocedures for logics that are sufficiently expressive to handle many knowledge bases. We have described a reasoning process whose conclusions are guaranteed to be true in any world in which the premises are true; in particular, *if* KB *is true in the* real *world,* *then any sentence* α *derived from* KB *by a sound inference procedure is also true in the real* *world.* So, while an inference process operates on “syntax”—internal physical configurations such as bits in registers or patterns of electrical blips in brains—the process *corresponds* to the real-world relationship whereby some aspect of the real world is the case by virtue of other aspects of the real world being the case. This correspondence between world and representation is illustrated in Figure 7.6. The final issue to consider is **grounding**—the connection between logical reasoning processes and the real environment in which the agent exists. In particular, *how do we know* *that* KB *is true in the real world?* (After all, KB is just “syntax” inside the agent’s head.) This is a philosophical question about which many, many books have been written. A simple answer is that the agent’s sensors create the connection. For example, our wumpus-world agent has a smell sensor. The agent program creates a suitable sentence whenever there is a smell. Then, whenever that sentence is in the knowledge base, it is true in the real world. Thus, the meaning and truth of percept sentences are defined by the processes of sensing and sentence construction that produce them. What about the rest of the agent’s knowledge, such as its belief that wumpuses cause smells in adjacent squares? This is not a direct representation of a single percept, but a general rule—derived, perhaps, from perceptual experience but not identical to a statement of that experience. General rules like this are produced by a sentence construction process called **learning**. Learning is fallible. It could be the case that wumpuses cause smells *except on* *February 29 in leap years*, which is when they take their baths. Thus, KB may not be true in the real world, but with good learning procedures, there is reason for optimism.



**The propositional logic**

We now present a simple but powerful logic called **propositional logic**. We cover the syntax

of propositional logic and its semantics—the way in which the truth of sentences is determined. Then we look at **entailment**—the relation between a sentence and another sentence

that follows from it—and see how this leads to a simple algorithm for logical inference. Everything takes place, of course, in the wumpus world.

**Syntax**

The **syntax** of propositional logic defines the allowable sentences. The **atomic sentences**

consist of a single **proposition symbol**. Each such symbol stands for a proposition that can

be true or false. We use symbols that start with an uppercase letter and may contain other

letters or subscripts, for example: P, Q, R, W1,3 and North. The names are arbitrary but are often chosen to have some mnemonic value—we use W1,3 to stand for the proposition

that the wumpus is in [1,3]. (Remember that symbols such as W1,3 are *atomic*, i.e., W, 1,

and 3 are not meaningful parts of the symbol.) There are two proposition symbols with fixed

meanings: True is the always-true proposition and False is the always-false proposition. **Complex sentences** are constructed from simpler sentences, using parentheses and **logical**

**connectives**. There are five connectives in common use:

￢ (not). A sentence such as ￢W1,3 is called the **negation** of W1,3. A **literal** is either an atomic sentence (a **positive literal**) or a negated atomic sentence (a **negative literal**).

∧ (and). A sentence whose main connective is ∧, such as W1,3 ∧ P3,1, is called a **conjunction**; its parts are the **conjuncts**. (The ∧ looks like an “A” for “And.”)

∨ (or). A sentence using ∨, such as (W1,3∧P3,1) ∨ W2,2, is a **disjunction** of the **disjuncts** (W1,3 ∧ P3,1) and W2,2. (Historically, the ∨ comes from the Latin “vel,” which means“or.” For most people, it is easier to remember ∨ as an upside-down ∧.)

⇒ (implies). A sentence such as (W1,3∧P3,1) ⇒ ￢W2,2 is called an **implication** (or conditional). Its **premise** or **antecedent** is (W1,3 ∧P3,1), and its **conclusion** or **consequent** is ￢W2,2. Implications are also known as **rules** or **if–then** statements. The implication symbol is sometimes written in other books as ⊃ or →.

⇔ (if and only if). The sentence W1,3 ⇔ ￢W2,2 is a **biconditional**. Some other books write this as ≡.



**Semantics**

Having specified the syntax of propositional logic, we now specify its semantics. The semantics defines the rules for determining the truth of a sentence with respect to a particular model. In propositional logic, a model simply fixes the **truth value**—true or false—for every

proposition symbol. For example, if the sentences in the knowledge base make use of the

proposition symbols P1,2, P2,2, and P3,1, then one possible model is m1 = {P1,2 =false, P2,2 =false, P3,1 =true}. With three proposition symbols, there are 23 =8 possible models—exactly those depicted in Figure 7.5. Notice, however, that the models are purely mathematical objects with no necessary connection to wumpus worlds. P1,2 is just a symbol; it might mean “there is a pit in [1,2]” or “I’m in Paris today and tomorrow.”

The semantics for propositional logic must specify how to compute the truth value of *any* sentence, given a model. This is done recursively. All sentences are constructed from atomic sentences and the five connectives; therefore, we need to specify how to compute the truth of atomic sentences and how to compute the truth of sentences formed with each of the five connectives. Atomic sentences are easy:

• True is true in every model and False is false in every model.

• The truth value of every other proposition symbol must be specified directly in the model. For example, in the model m1 given earlier, P1,2 is false.

For complex sentences, we have five rules, which hold for any subsentences P and Q in any

model m (here “iff” means “if and only if”):



The rules can also be expressed with **truth tables** that specify the truth value of a complex sentence for each possible assignment of truth values to its components. Truth tables for the

five connectives are given in Figure 7.8. From these tables, the truth value of any sentence **s**

can be computed with respect to any model m by a simple recursive evaluation. For example, the sentence ￢P1,2 ∧ (P2,2 ∨ P3,1), evaluated in m1, gives true ∧ (false ∨ true) = true ∧ true =true. Exercise 7.3 asks you to write the algorithm PL-TRUE? (s,m), which computes the truth value of a propositional logic sentence s in a model m.

The truth tables for “and,” “or,” and “not” are in close accord with our intuitions about the English words. The main point of possible confusion is that P ∨Q is true when P is true or Q is true *or both*. A different connective, called “exclusive or” (“xor” for short), yields false when both disjuncts are true.7 There is no consensus on the symbol for exclusive or; some choices are .

The truth table for ⇒ may not quite fit one’s intuitive understanding of “P implies Q” or “if P then Q.” For one thing, propositional logic does not require any relation of *causation* or *relevance* between P and Q. The sentence “5 is odd implies Tokyo is the capital of Japan” is a true sentence of propositional logic (under the normal interpretation), even though it is a decidedly odd sentence of English. Another point of confusion is that any implication is true whenever its antecedent is false. For example, “5 is even implies Sam is smart” is true, regardless of whether Sam is smart. This seems bizarre, but it makes sense if you think of “P ⇒ Q” as saying, “If P is true, then I am claiming that Q is true. Otherwise I am making no claim.” The only way for this sentence to be *false* is if P is true but Q is false. The biconditional, P ⇔ Q, is true whenever both P ⇒ Q and Q ⇒ P are true. In English, this is often written as “P if and only if Q.” Many of the rules of the wumpus world are best written using ⇔. For example, a square is breezy *if* a neighboring square has a pit, and a square is breezy *only if* a neighboring square has a pit. So, we need a biconditional, where B1,1 means that there is a breeze in [1,1].

**A simple knowledge base**

Now that we have defined the semantics for propositional logic, we can construct a knowledge base for the wumpus world. We focus first on the *immutable* aspects of the wumpus world, leaving the mutable aspects for a later section. For now, we need the following symbols for each [x, y] location:

Px,y is true if there is a pit in [x, y].

Wx,y is true if there is a wumpus in [x, y], dead or alive.

Bx,y is true if the agent perceives a breeze in [x, y].

Sx,y is true if the agent perceives a stench in [x, y].

The sentences we write will suffice to derive ￢P1,2 (there is no pit in [1,2]), as was done

informally in Section 7.3. We label each sentence Ri so that we can refer to them:

• There is no pit in [1,1]:

R1 : ￢P1,1 .

• A square is breezy if and only if there is a pit in a neighboring square. This has to be

stated for each square; for now, we include just the relevant squares:

R2 : B1,1 ⇔ (P1,2 ∨ P2,1) .

R3 : B2,1 ⇔ (P1,1 ∨ P2,2 ∨ P3,1) .

• The preceding sentences are true in all wumpus worlds. Now we include the breeze

percepts for the first two squares visited in the specific world the agent is in, leading up

to the situation in Figure 7.3(b).

R4 : ￢B1,1 .

R5 : B2,1 .