An assumption-based truth maintenance system, or ATMS, makes this type of context-switching between hypothetical worlds particularly efficient. In a JTMS, the maintenance of justifications allows you to move quickly from one state to another by making a few retractions and assertions, but at any time only one state is represented. An ATMS represents all the states that have ever been considered at the same time. Whereas a JTMS simply labels each sentence as being in or out, an ATMS keeps track, for each sentence, of which assumptions would cause the sentence to be true. In other words, each sentence has a label that consists of a set of assumption sets. The sentence holds just in those cases in which all the assumptions in one of the assumption sets hold.

Truth maintenance systems also provide a mechanism for generating explanations. Technically, an explanation of a sentence \( P \) is a set of sentences \( E \) such that \( E \) entails \( P \). If the sentences in \( E \) are already known to be true, then \( E \) simply provides a sufficient basis for proving that \( P \) must be the case. But explanations can also include assumptions—sentences that are not known to be true, but would suffice to prove \( P \) if they were true. For example, one might not have enough information to prove that one's car won't start, but a reasonable explanation might include the assumption that the battery is dead. This, combined with knowledge of how cars operate, explains the observed nonbehavior. In most cases, we will prefer an explanation \( E \) that is minimal meaning that there is no proper subset of \( E \) that is also an explanation. An ATMS can generate explanations for the "car won't start" problem by making assumptions (such as "gas in car" or "battery dead") in any order we like, even if some assumptions are contradictory. Then we look at the label for the sentence "car won't start" to read off the sets of assumptions that would justify the sentence.

The exact algorithms used to implement truth maintenance systems are a little complicated, and we do not cover them here. The computational complexity of the truth maintenance problem is at least as great as that of propositional inference—that is, \( \text{NP-hard} \). Therefore, you should not expect truth maintenance to be a panacea. When used carefully, however, a TMS can provide a substantial increase in the ability of a logical system to handle complex environments and hypotheses.

### 12.7 The Internet Shopping World

In this final section we put together all we have learned to encode knowledge for a shopping research agent that helps a buyer find product offers on the Internet. The shopping agent is given a product description by the buyer and has the task of producing a list of Web pages that offer such a product for sale, and ranking which offers are best. In some cases the buyer's product description will be precise, as in \textit{Canon Rebel XTi digital camera}, and the task is then to find the store(s) with the best offer. In other cases the description will be only partially specified, as in \textit{digital camera for under $300}, and the agent will have to compare different products.

The shopping agent's environment is the entire World Wide Web in its full complexity—not a toy simulated environment. The agent's percepts are Web pages, but whereas a human
Section 12.7. The Internet Shopping World

Example Online Store
Select from our fine line of products:
- Computers
- Cameras
- Books
- Videos
- Music

<html>
  <h1>Example Online Store</h1>
  <i>Select</i> from our fine line of products:
  <ul>
    <li><a href="http://example.com/computers">Computers</a></li>
    <li><a href="http://example.com/cameras">Cameras</a></li>
    <li><a href="http://example.com/books">Books</a></li>
    <li><a href="http://example.com/videos">Videos</a></li>
    <li><a href="http://example.com/music">Music</a></li>
  </ul>
</html>

Figure 12.8  A Web page from a generic online store in the form perceived by the human user of a browser (top), and the corresponding HTML string as perceived by the browser or the shopping agent (bottom). In HTML, characters between `<i>` and `</i>` are markup directives that specify how the page is displayed. For example, the string `<i>Select</i>` means to switch to italic font, display the word Select, and then end the use of italic font. A page identifier such as http://example.com/books is called a uniform resource locator (URL). The markup `<a href="url">Books</a>` means to create a hypertext link to `url` with the anchor text `Books`.

Web user would see pages displayed as an array of pixels on a screen, the shopping agent will perceive a page as a character string consisting of ordinary words interspersed with formatting commands in the HTML markup language. Figure 12.8 shows a Web page and a corresponding HTML character string. The perception problem for the shopping agent involves extracting useful information from percepts of this kind.

Clearly, perception on Web pages is easier than, say, perception while driving a taxi in Cairo. Nonetheless, there are complications to the Internet perception task. The Web page in Figure 12.8 is simple compared to real shopping sites, which may include CSS, cookies, Java, Javascript, Flash, robot exclusion protocols, malformed HTML, sound tiles, movies, and text that appears only as part of a PEG image. An agent that can deal with all of the Internet is almost as complex as a robot that can move in the real world. We concentrate on a simple agent that ignores most of these complications.

The agent’s first task is to collect product offers that are relevant to a query. If the query is "laptops," then a Web page with a review of the latest high-end laptop would be relevant, but if it doesn’t provide a way to buy, it isn’t an offer. For now, we can say a page is an offer if it contains the words "buy" or "price" or "add to cart" within an HTML link or form on the