4 Artificial Intelligence
Programming Languages

4.1 Introduction

4.1.1 Syntax and semantics

Programming languages are formalisms for describing problems or solutions to problems in a way understandable (i.e. executable) by computers. A programming language is defined through its syntax and semantics.

In natural languages (e.g. English), the syntax specifies which words are part of the language and how words can be combined into correct sentences; in programming languages the syntax plays exactly the same role, only it is much simpler. Also note that in formal languages (and hence in computer languages) words are often referred to as symbols, sentences as expressions.

A precise formalism for describing the syntax of a broad class of languages is introduced in the next section. Such a formalism is important for two reasons:

- syntactic analysis of expressions has many potential applications to economics (see Chapter 7 for an application)
- it is useful for describing programming languages, as in Sections 3 - 6 of this Chapter

Semantics deals with the meaning associated to sentences (expressions) and words (symbols). One way whereby to describe semantics of a language is to translate it into a language whose semantics is known. In the following section, the meaning of the programs will be informally stated in English. This approach to semantics is adequate for the purposes of this work, although this is not theoretically acceptable, due to the ambiguities inherent in any natural language.
4.1.2 AI programming languages

In Chapter 3 it was stressed how the complexity of real-world problems stimulated the development of paradigms for knowledge representation. Computer languages of interest to AI encompass one or more of such knowledge representation criteria.

Since knowledge representation paradigms are very heterogeneous, AI programming languages do not share as many features as, say, COBOL-type languages. However, two characteristics are common to most of them. First, they use symbols as the basic data type. Second, they are interactive. Each of these topics is briefly addressed in the two following paragraphs.

4.1.3 Symbols and symbolic expressions

A symbol is the representation in a formal language of an elementary entity (an object, a concept, and so on) in the problem context; symbols are not decomposable.

Example 1: Symbols

American-currency
A33
gfwystr

are all symbols. In computer languages, symbols are usually defined as strings of characters and numbers. Although the first symbol has a clear interpretation in English, this interpretation is irrelevant for the execution of formal manipulations.

Languages based on symbols (so-called symbolic languages) include methods for symbolic manipulations. Typical elementary manipulations on symbolic structures are:

- creation, deletion or modification of symbolic structures of varying complexity
- retrieval, addition, substitution or deletion of substructures or in case symbols from the structures
- comparison between structures or substructures (this problem has a special relevance in AI and is called pattern-matching)
- "ad-hoc" processing of particular symbols (e.g. arithmetic operations on numeric symbols, logical operations on the symbols "true" and "false")

Different knowledge representation formalisms make different hypotheses on how symbols should be combined into structures and about the kind of manipulations allowed over the structures (corresponding to different hypotheses on how intelligent entities acquire, store and effectively use knowledge, see Chapter 3.1); the programming languages are designed to support the different symbolic structures and manipulation methods.

4.1.4 Interactivity and language interpreters

A language is interactive if it is designed so that expressions are executed as soon as they are entered by the user. An interactive language consists of an interpreter, i.e. a program which reads an expression typed in by the user or from a file, evaluates it, returns the result to the user, and waits for a new expression. All languages described in this chapter are interactive.

Example 2: A simple formal symbolic language

Consider the following toy formal symbolic language:

i. allowable symbols are English words (for instance, all words listed in the English dictionary). Such symbols have obviously no meaning to the