The Jodie programming language

Tibor Csáki and Krisztián Veréb

One of the main policies of the Artificial Intelligence (AI) research is the creation of different paradigmatical languages supporting the solutions of AI problems. These developments have resulted the functional LISP and the declarative Prolog programming languages, among others. In spite of its many advantages, the declarative paradigm has the disadvantage of lacking the procedural programming equipment. This programming equipment is essential for large and complex developments. One possible solution to this problem is the creation and use of hybrid languages.

The usual method of creating a multiparadigmatical language is to extend a declarative one. The results of this method are declarative languages with a mixed structure, but they are not as universal as needed. The other possibility is to create an absolutely new language which is expressive enough. However, such a language could be difficult to learn.

To avoid these difficulties, we have decided to have well-known and easy-to-learn languages as the basis of our newly created language, Jodie. The concept of Jodie is to establish a link between a declarative (Prolog) and an imperative (C) programming language. For this establishment it is necessary to introduce new grammatical elements and make compromises. The main goal of Jodie is to separate the elements of different paradigms. We have solved this requirement with the help of new types which work like functions and we can operate on these special functions using 'conventional' C function calls. There are pure Prolog codes in the bodies of these AI functions. The communication has been realised with the help of parameters between the C and the Prolog functions. The parameters transferring information between the called Prolog routine and the calling C syntactical unit have a special type, namely the query type. This type appears only by its literals. Variables with this type are not allowed.

As a result of these steps, it was obvious to integrate programming objects of Automata Theory (e.g., Turing machine, Lindenmayer System, Finite State Automata) like we did it with Prolog before. There are also possibilities of integrating other programming objects, as well. To establish the usage of automata we introduced new AI constant and AI function types to make their description possible.

Since Jodie is easy to learn, contains a whole real Prolog and automata codes can be implemented with its help as well, this language is practical to use in the education of AI and Automata Theory.

Keywords: Artificial Intelligence, Automata Theory, Multi Paradigmatical Languages, Prolog, C