The Development of Expert System Shell with Constraint-Based Programming

Nittaya Kerdprasop, Kanjana Intharachatorn, Kittisak Kerdprasop

Data Engineering Research Unit, School of Computer Engineering, Suranaree University of Technology, 111 University Avenue, Nakhon Ratchasima 30000, Thailand
nittaya@sut.ac.th

Abstract. Expert system is a computer program that is different from other conventional computer programs in that it incorporates specific knowledge, which can be human judgment, experience, and expertise, in order to provide knowledgeable advice to users. The main difference is caused from the structure of an expert system that contains the two loosely coupled parts: knowledge inference engine and knowledge base, instead of one tightly integrated structure. The loose coupling allows knowledge base contents to be dynamically added, removed, modified, or even completely changed to another subject area, whereas the inference engine remains intact and needs no modification. We refer to the knowledge inference engine as an expert system shell because it can be viewed as an outer layer program to infer knowledge advice from the inner knowledge base core. The development of expert system shell can be efficiently achieved through the support of logic-based language such as Prolog. In this paper, we propose and demonstrate a different scheme of expert system shell development using a constraint-based paradigm with the ECLiPSe constraint system. The experimental results reveal that the constraint-based paradigm uses less memory during execution and provides more concise form of knowledge representation than the logic-based paradigm.

Keywords: Expert system shell, Constraint programming, ECLiPSe constraint system.

1 Introduction

Expert system can be viewed as an intelligent system that acts as a human expert giving advice in some specific application domains. The field of expert system has emerged since the mid-1960s [10] as a successful implementation of artificial intelligence technology. The later coined term knowledge-based system has a close meaning to the expert system, but the main emphasis is on decision support.

Despite its variety in application domains, most expert systems are composed of two main separate modules: a knowledge base and an inference engine. The knowledge base stores expertise in a particular domain. The inference engine deduces advice from the stored knowledge with reasoning mechanism, and also generates explanation to users regarding the inferred advice. The separation of knowledge base
and inference engine allows users to connect a single inference engine to multiple knowledge bases [6], [7], [8]. Building an expert system shell with logic-based language such as Prolog [4] is not a difficult task because the language system provides knowledge representation formalism suitable for the rule-based expert system. We are, therefore, interested in developing the shell with a more advanced scheme of constraint logic programming. Our main focus is to compare efficiencies of both schemes.

Constraint programming has emerged four decades ago as a programming paradigm to solve constraint satisfaction and optimization problems [1], [5], [9]. It has been recently applied to solve biological [2], genomic [11], pattern mining [3], and software testing [12].

We present in this paper a constraint-based method to implement the rule-based expert system shell. In section 2, we briefly discuss background concepts of constraint logic programming. In section 3, we explain the details of our implementation of the constraint-based expert system shell. We then conclude the paper in section 4.

2 Preliminaries on Constraint Programming

Constraint programming is a programming paradigm normally applied to solve combinatorial search problems such as flight scheduling, crew roster assigning, logistic planning, and many more of this kind. The main steps of constraint programming are: (1) Users specify a problem by defining the variables together with their associated domains and constraints on these variables, and (2) The search procedure and constraint solver find solutions, which are values assigned to the specified variables such that all constraints are satisfied.

It is obvious from the program structure that constraint programming has been designed to solve constraint satisfaction problems that have been extensively studied in artificial intelligence. The efficiency of constraint programs is basically due to the constraint propagator feature in a constraint solver. The function of constraint propagator is to reduce the domains of variables by inferring from the existing constraints and then to prevent the search procedure from visiting parts of the search tree that do not contain any solution.

A constraint propagator takes as input a domain $D$ from which a variable can be assigned its value, and a set of constraints $C$. The output of the propagator is a reduced domain $D'$. For instance, given that $X, Y, Z$ are variables, the domains:

$$D(X) = \{a, c, d\}, \quad D(Y) = \{a, b, c, d\}, \quad D(Z) = \{c\},$$

and a set of constraints $C = \{ X=Y \land Y \neq Z \}$, the output of the constraint propagator are: $D'(X) = D'(Y) = \{a, d\}$, and $D'(Z) = \{c\}$.

The repeated application of propagator can lead to increasingly stronger (that is, smaller) domains. The propagator continues until it reaches a fixed point in which the domains cannot be further reduced. At this stage, the search procedure (either global or heuristic-based) can efficiently start assigning possible value to each variable.

At present, there are several constraint systems that provide functions to specify (or model) the problems and maintain the constraint consistency efficiently. They are
called constraint programming systems if they are based on procedural languages. The systems are classified as constraint logic programming systems if they are based on logic programming languages. The main benefit of constraint logic programming scheme are two folds: (1) the declarative style allows users to specify a problem itself, instead of specifying how to solve the problem, and (2) a high level of knowledge representation facilitates the inclusion of new knowledge that is highly dynamic.

Most constraint logic programming systems provide a large set of predefined constraints (such as 'alldifferent') and powerful search commands (such as 'labeling') to solve the combinatorial problems. The predefined constraints and exhaustive depth-first search procedure aim at solving a general class of constraint satisfaction problems. These facilities ease the development of knowledge intensive task such as expert system shell rapid prototyping.

3 The Development of a Constraint-Based Expert System Shell

The design of constraint-based expert system shell (as shown in Figure 1) is slightly different from the logic-based method. Constraint features have been added in the user interface, inference engine, and knowledge base parts. The flowcharts of ‘solve’ and ‘why’, which are modules to solve the user’s query and then explain recommendation, respectively, are given in Figure 2.

The knowledge base content in our constraint-based implementation has been changed from the rule-based format ‘IF-THEN’ to the constraint clauses, as shown in Figure 3. This simple knowledge base contains preferences of tourists. The ‘menuask’ module is also modified to include domain constraint. A comparative implementation of this module is demonstrated in Figure 4. Example of program running using the tourist attraction site knowledge base can be illustrated in Figure 5.

Fig. 1. Modules of a constraint-based expert system.
Fig. 2. The ‘solve’ flowchart.

```plaintext
% more Code Here
P &= ['resort', 'beach', 'pearl', 'coral'],
K &= ['resort', 'forest', 'roasting chicken', 'stone castle'],
B &= ['hotel', 'shopping'],
L &= ['hotel', 'hot spring', 'ceramic bowl', 'coal mine'],
type(Sin),
(if ord_subset([Sin],B)
  then
    (type(Lin),
      if ord_subset([Sin,Lin],B)
        then
          Vout &= 'Bangkok'
        else
          %more Constraint Here
   end).
```

Fig. 3. Knowledge base of the constraint expert system shell.
4 Conclusion

We present in this paper a rapid prototyping the expert system shell using constraint-based programming paradigm. The main purpose is to observe program coding difficulty, running and memory usage behavior. The insight understandability is expected to be fundamental knowledge for designing constraint solver that is more appropriate for the expert system shell development.

Knowledge base contents are conventionally constructed by knowledge engineers who are not an expert of the specific domains. With constraint-based method, the knowledge elicitation task is expected to be less error prone. The constraint paradigm is also planned to be used as automatic knowledge extraction scheme to learn specific knowledge from stored experiences and expertise.

Acknowledgments. This work has been supported by grants from the National Research Council of Thailand (NRCT) and Suranaree University of Technology.
References