KNOWLEDGE BASED TRANSLATION AND PROBLEM SOLVING IN AN INTELLIGENT INDIVIDUALIZED INSTRUCTION SYSTEM

Namho Jung and John E. Biegel
Intelligent Simulation Laboratory
Department of Industrial Engineering and Management Science
University of Central Florida
Orlando, Florida 32816

namho@oak.ists.engr.ucf.edu
biegel@oak.ists.engr.ucf.edu

ABSTRACT

An Intelligent Individualized Instruction (I^3) system is being built to provide computerized instruction. We present the roles of a translator and a problem solver in an intelligent computer system. The modular design of the system provides for easier development and allows for future expansion and maintenance. CLIPS modules and classes are utilized for the purpose of the modular design and inter module communications. CLIPS facts and rules are used to represent the system components and the knowledge base. CLIPS provides an inferencing mechanism to allow the I^3 system to solve problems presented to it in English.

INTRODUCTION

The Intelligent Individualized Instruction (I^3) system is an intelligent teaching system that makes possible the knowledge transfer from a human to a computer system (knowledge acquisition), and from the system to a human (intelligent tutoring system (ITS)). The ITS portion of the I^3 system provides an interactive learning environment where the system provides self-sufficient instruction and solves the problem presented in a written natural language. Self-sufficient instruction means that no instructor is required during the learning cycle. Solving problems written in a natural language means that the system is able to 'understand' the natural language: It is not an easy task, especially without any restriction on the usage of vocabulary and/or format.

Two I^3 system modules, a Translator and an Expert (a problem solver and a domain glossary), understand a problem presented to it in English by translation and keyword pattern matching processes. The I^3's pattern matching method uses the case-based parsing [Riesbeck and Schank 1989] that searches its memory (knowledge base) to match the problem with stored phrase templates. Unlike other case-based parsers (e.g., CYC project [Lenat and Feigenbaum, 1989]) the I^3 system does not understand the problem statement as a human does. A human uses common sense or other background knowledge to understand it. Rather the I^3 system 'understands' enough about the problem for the system to be able to solve the problem. We will discuss how the system 'understands' the human language.

AN INTELLIGENT INDIVIDUALIZED INSTRUCTION (I^3) SYSTEM

The I^3 system is a Knowledge Based System that is composed of domain dependent modules, domain independent modules, and a User interface module. The domain dependent modules (the Domain Expert and the Domain Expert Instructor) carry the domain expertise that enables the other modules remain domain independent. The separation of these domain dependent modules from the rest of the system makes the system reusable. Whenever the I^3 system is applied to another domain, only the domain-dependent knowledge of the new domain is needed.

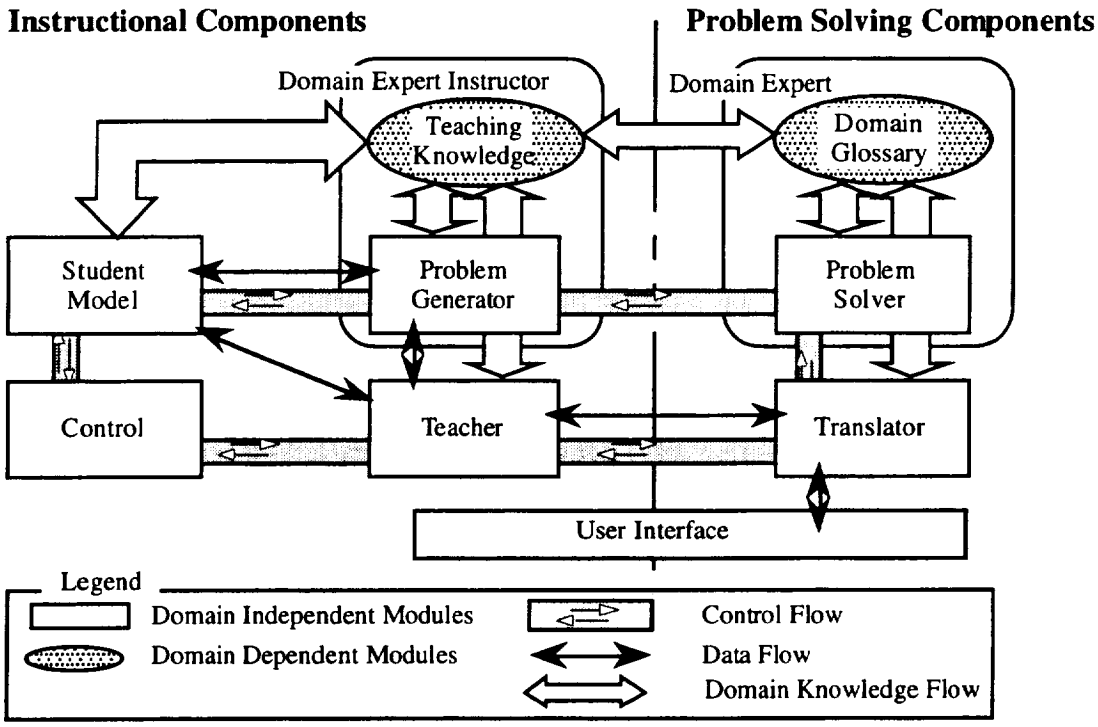

Figure 1. System architecture of I³'s intelligent teaching system

The goal of the I³ system is to provide a student with individualized learning to attain competency [Biegel 1993]. The I³ knowledge presentation subsystem generates self-sufficient instructions. The I³ system contains instructional components (Student Model, Teacher, Domain Expert Instructor, Control, and User Interface) and problem solving components (Translator and Domain Expert).

- The Student Model module evaluates and maintains the trainee's performance overall or on individual lessons.
- The Teacher module contains the knowledge about generic didactic strategies to customize each lesson by selecting and sequencing the instruction material.
- The Domain Expert Instructor (DEI) module represents the domain dependent teaching methodology. The DEI module provides the Teacher module with the teaching strategies, and the Student Model module with the evaluation criteria for the individual and/or overall lesson.
- The Control module manages the operation and maintains the modules' communications within the system.
- The User Interface module handles all communications between the user and the system.
- The Translator module parses a trainee's input and translates it into a system-understandable format.
- The Domain Expert (DE) module contains a knowledge base representing the problem-solving knowledge of a human domain expert.

DESIGN OF THE TRANSLATOR AND THE PROBLEM SOLVER

The Translator module and the Domain Expert module provide the user with the proper interpretation of the problem and the correct solution. Together, they allow a system to apply domain expert heuristics to solve problems by pattern matching. Pattern matching allows the

system to 'understand' a problem statement within the domain for which the system has been built. The problem statement can be translated into a set of rules and facts. Since the problem solver cannot translate English (or other natural language) directly into computer readable code, it relies on a translator to provide a communication mechanism between the user and the computer.

The Translator provides a full duplex communication medium between the human and the computer. The user will not be constrained in the format of the input problem and will not be required to do any parsing or be restricted to a limited syntax. The Translator module translates the domain jargon of an input problem into a system understandable format.

The major tasks of the translator are: (1) to convert text into computer readable code, and (2) to provide a knowledge base conversion and problem representation process. The text conversion process includes checking for correct spelling and removing all unnecessary symbols. The knowledge base conversion process includes (1) the conversion of a written number to a numeric value, (2) filtering out unnecessary words, and (3) replacing words with stem (root) or exemplar words. The knowledge based process uses a number conversion list, a list of words unnecessary for problem solving, and a domain thesaurus. The Intelligent Individualized Instruction system separates the knowledge base (domain glossary) which is the collection of the domain thesaurus, the domain template dictionary, and the unnecessary word listing, from the translation process.

The Domain Expert interprets the translated input using a domain vocabulary as a reference, selects a suitable method from a list of solution methods, and finds a solution. The Domain Expert (DE) consists of three parts, a Domain Glossary (DG), a set of Managers, and a Problem Solver (PS). The DE imitates the expert's methodologies of problem solving: The DG acts as the expert's memory by providing the necessary problem solving knowledge. Each Manager acts as the expert's procedural knowledge by solving a routine of the problem in one specific area. The PS acts as the scheduling and reasoning processes by controlling and scheduling the Managers in proceeding toward the solution of a problem.

The Domain Glossary represents the domain expertise. The DG consists of a domain template dictionary, a domain thesaurus, an unnecessary word listing, a domain symbol list, and a domain theory list. The DG represents relational knowledge (e.g., one year is twelve months), factual knowledge (e.g., 'interest rate' means domain variable i), and a list of words and symbols (e.g., '%' has a special meaning of interest rate in the Engineering Economy domain).

Special dictionaries for the domain provide benefits such as faster look up access than a general dictionary, and a higher priority to find the correct interpretation. The problem solver does not have to consider all different combinations of words' variables. It searches its own smaller dictionary that contains the necessary information in the application domain. English text (problem statement) is interpreted/translated by using the domain thesaurus without knowing the general meaning of the word. The thesaurus contains all words relevant to the domain. Each word is connected to a list of possible interpretations. Each word in the statement is looked up in the thesaurus and replaced by all relevant symbols.

The number of words in the domain vocabulary will vary between domains, but a vocabulary of 500 or so words and symbols will most likely cover most of the undergraduate engineering domains.

Each Manager handles one specific area of problem solving. It is a self-sufficient object that contains procedural knowledge (rules) and factual knowledge (facts) attached to it, and that knows how to handle situations upon request. When activated, a Manager searches the input statement for matched patterns in its templates. If a match is found, the Manager processes or interprets that portion of the problem statement. For example, a Date Manager knows how to interpret key terms that are related to the date, how to compare two date instances, and how to calculate the period

from two date instances. When a problem statement contains "... invest $5000 on January 1, 1994, ... will be accumulated in 5 years hence ... ", the Date Manager replaces the statement with " ... invest $5000 on [D1] ... will be accumulated in [D2] ... " where [D1] and [D2] are instances of a Date class and are represented as:

```
([Date::D1] is a Date                ([Date::D2] is a Date
    (year        1994)                   (year        5)
    (month  1)                           (month  0)
    (day         1)                      (day         0)
    (base        none) )                 (base        [Date::D1]) )
```

Some mangers handle both domain dependent and independent situations based on the factual knowledge they have. Communication among managers can be made through dedicated communication channels, such as CLIPS class objects or templates.

The Domain Expert module is of a modular design and maintains the separation of strategic knowledge from factual knowledge. The domain expertise can be categorized into three levels: high level control knowledge (a Problem Solver), middle level procedural knowledge (Managers), and low level factual knowledge (a Domain Glossary). By nature, the low level factual knowledge tends to be domain specific, and the high level control knowledge tends to be a domain independent. Any addition to the knowledge base can be accomplished by adding a Manager and its associated knowledge into the DG.

PROBLEM SOLVING IN THE I^3 SYSTEM

The problem solver applies a separate-and-solve method that breaks a problem statement into several small blocks, interprets each block, and then logically restructures them. The problem solving steps include interpretation of the problem statement, selection of the formula, and mathematical calculation. The steps are depicted in Figure 2 in which boxes on the left hand side represent the changes of the problem statement from input English text to the answer. The middle ovals show the problem solving processors. The right hand side boxes represent the domain expertise of the domain glossary. The problem solving process is generic so it can be used in other domains if the new domain expertise is available.

The I^3 system problem solving routine is performed by the problem solving components: the User Interface, the Translator, and the Domain Expert (the Problem Solver, the Managers, and the Domain Glossary). The routine includes initial domain independent processes (translating and filtering an input problem), and main domain dependent processes (interpreting the problem, selecting a solution method, and deriving an answer).

A user enters an engineering economics problem through the User Interface, as shown in Figure 3. The Translator performs filtering process by checking correct spelling using its English dictionary. The Translator converts the input problem statement into system understandable format; plural words to singular; past tense to present; uppercase words to lowercase; verbal numbers to numeric values; and separates symbols from numeric values (Figure 4). For example, part of the problem statement "If *$10,000* is *invested* at 6% interest *compounded* annually" becomes "if $ 10000 is invest at 6 % interest compound annual". Now, all the elements in the problem statement are known to the system.

The problem statement is divided into several blocks in order to distribute the complexity of the problem (Figure 5). Each block is a knowledge unit that contains a domain variable and a numeric value. The knowledge unit contains necessary as well as unnecessary information for interpreting the problem statement. Any unnecessary word such as 'at' must be removed before reasoning, because they only required overhead on the system during the process of reasoning. As an

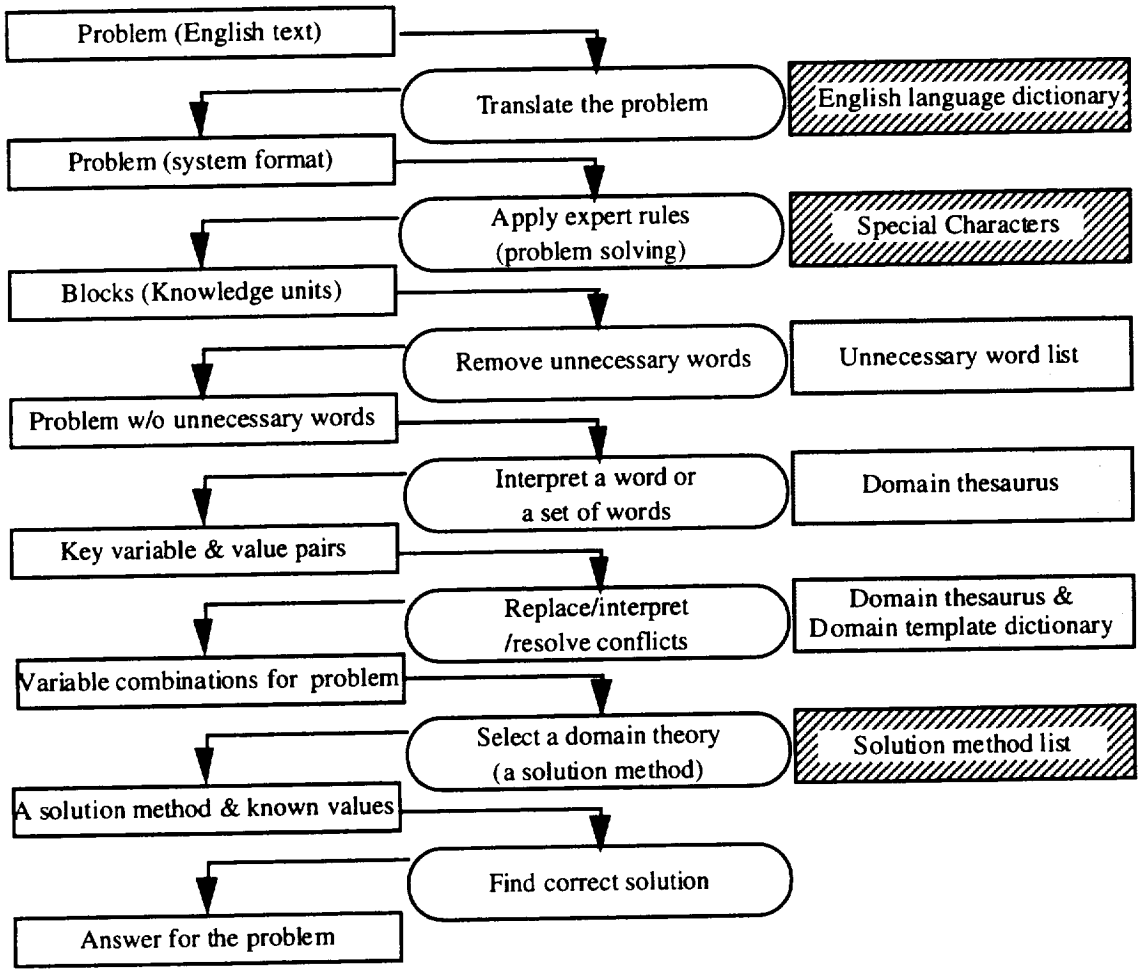

Figure 2. problem solving process

How much money will you have accumulated three years from now if $10,000 is invested at 6% interest compounded annually?

Figure 3. A sample problem in engineering economics

Plural to singular:	years => year
past tense to present tense:	invested => invest compounded => compound accumulated => accumulate
Upper case to lower case:	How => how
verbal number to numeric number:	three => 3
remove comma within a number:	10,000 => 10000
separate symbol from number:	$10000 => $ 10000 6% => 6 %

Figure 4. Conversion process

Block (Knowledge unit)	Unnecessary word
1. how much money will you have accumulate	you have
2. three year from now	
3. if $10,000 is invest	if, is
4. at 6% interest compound annually	at

Figure 5. Removing unnecessary words from each knowledge unit

Knowledge unit	Interpretation
1. how much money will ... accumulate	Find F
2. 3 year from now	N = 3
3. ... $ 10000 ... invest	P = 10000
4. ... 6 % interest compound annual	i = 6%

Figure 6. Knowledge Units of the Sample Problem

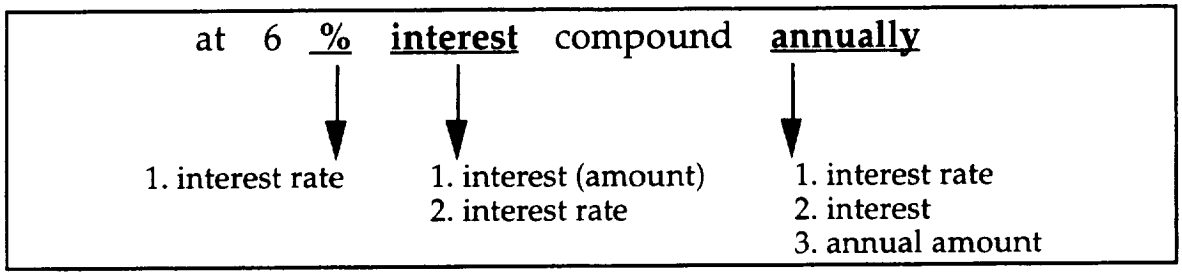

Figure 7. Domain Thesaurus Interpretation Example

Given P, i, N, and Find F.
Solution strategy is F = P (F / P, i%, N))

Figure 8. Finding a Solution Strategy

instance, a block "if $10000 is invest" will be interpreted as a present worth "P = $10000" because 'if' and 'is' are unnecessary, 'invest' is used previously as present worth, and '$10000' is a value of the variable. Unnecessary words can be found in all problems in the domain, but not in the list of domain templates. A domain template is a sequence of words, a knowledge unit, that is used to interpret a domain variable. The unnecessary word is not used uniquely: it could be found in the templates for all different variables.

The Problem Solver interprets each knowledge unit by applying the domain thesaurus and domain template dictionary. For example, when a text block, 'at 6% interest compound annual' is given to the system, the knowledge base provides interpretation of the block: 1) word by word: The word 'at' is an unnecessary word for solving the problem. Next word, '%', will be interpreted as 'interest rate,' and 'compound' as 'interest rate.' The word 'interest' has two meanings: 'interest' (amount of money) and 'interest rate' (rate). The last word 'annual' could be represented in three different variables: 'interest,' 'interest rate,' and 'annual value' (Figure 7). 2) as a template 'interest compound annual' meaning 'interest rate.' Such conflicts will be resolved by selecting an interpretation with the highest priority among all different possibilities. The knowledge base provides the necessary knowledge to determine which one has higher priority.

The Problem Solver sends the interpretation of the problem statement to the domain theory selector. The interpretation of the problem (for example, P = $10000, i = 6%, N = 3 year, and F is unknown) is used to select an appropriate solution method (F = P (F / P, i%, N)) (Figure 8). The system applies the interpretation of the problem to the solution method (F = 10000 (F / P, 6%, 3)). The solution found is presented to the user through the User Interface.

CONCLUSION

The Translator and the Problem Solver in the I^3 system have demonstrated that the knowledge based interpretation of natural language is feasible. Modular design of the Problem Solver provides the system's expandability and reusability. Expanded problem solving capability of the system can be accomplished by adding more knowledge to the Domain Glossary. Reusability can be enhanced by replacing or adding managers to the Problem Solver without reprogramming other parts of the system. Combining rule based processing with objects (or an integration of object oriented system with an intelligent system) makes it possible to define domain knowledge about the application further than with rules alone.

The I^3 system is being developed on an IBM compatible 486 machine using the C/C++ programming language (Microsoft Visual C++) and CLIPS 6 (C Language Integrated Production System, by NASA Lyndon B. Johnson Space Center, a forward chaining expert system shell based on the Rete algorithm).

REFERENCE

1. Biegel, John, "I^3: Intelligent Individualized Instruction," PROCEEDINGS OF THE 1993 INTERNATIONAL SIMULATION CONFERENCE, San Francisco, CA., OMNIPRESS, Madison, Wisconsin, November, 1993, 243-249.

2. Chung, Minhwa, and Moldovan, Dan, "Applying Parallel Processing to Natural-Language Processing." IEEE EXPERT INTELLIGENT SYSTEMS & THEIR APPLICATIONS, IEEE Computer Society, Vol. 9, Number 1, February, 1994, 36-44.

3. Martin, Charles, "Case-based Parsing," INSIDE CASE-BASED REASONING, Riesbeck Christopher K. and Schank, Roger C., Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey, 1989, 319-352.

4. Lenat, Douglas, and Feigenbaum, Edward, "On the Thresholds of Knowledge," APPLICATIONS OF EXPERT SYSTEMS, Vol. 2, Ed. Quinlan, J. Ross, Addison-Wesley Publishing Company, Sydney, Australia, 1989, 36-75.