On Developing Case-Based Tutorial Systems with Conceptual Graphs

Pak-Wah Fung and Ray H. Kemp

Institute of Information Sciences and Technology
Massey University
Palmerston North, New Zealand
{P.W.Fung, R.Kemp}@massey.ac.nz

Abstract. In this paper, a case-based intelligent tutoring system (CBITS) using conceptual graphs to represent the cases is described. Among others, two core issues to be addressed by CBITS developers are indexing and how learning activities can be constructed from the cases. Addressing the former, the authors discovered that the minimum common generalization of a set of graphs is a powerful means of embedding different tutorial primitives into the cases. This approach provides an extremely rich indexing vocabulary for the cases and relieves the developers from speculative assignment of indexes. For constructing learning activities an operational semantics for the case graphs is defined and with this semantics the instructor can reason about the graph structure and provide intelligent guidance to the students in exploring the case contents. Newtonian mechanics is the testing domain of the proposal but its underlying methodology should be equally applicable to other subject domains.

1 Introduction

Intelligent tutoring systems (ITS) is an exciting area and the development of this field will no doubt have a lot of impact to our educational systems. One of our long term ambitions is to develop an open-ended campus-wide learning resource which may include online exercises, worked examples, past students' projects or real-world case studies, etc. From anywhere and at any time, the students can call upon a domain specific tutorial agent to provide individualized guidance on solving problems or explaining the case contents in terms of fundamental principles. This paper reports our progress in representing the collected cases and the mechanism of making the case contents usable in tutorial settings.

We adopted the position of case-based tutoring which is chiefly based on the experimental discovery (e.g. [1], [10], [11], [17] and [21]) that experts excel mainly in their own domains, and there is very little evidence that a person highly skilled in one domain can transfer that skill to another. The excellence of competent problem solvers is due to their possession of a good deal of domain-specific knowledge, an achievement that requires many years of intensive experience over many cases. This phenomenon applies to learning music, to science and to chess playing. No one can reach genius levels of performance without at least ten years of practice. Therefore it is well-grounded to draw the conclusion that the expertise of competent problem solvers is developed through their exposure to very many cases pertaining to their own particular subject domain. One of the key differences in domain knowledge between a more competent problem-solver and a less competent one is the number and range of cases to which they have been exposed. The remaining questions are: what structure best represents the cases and how can we help the students acquire the domain knowledge embedded in the cases by developing an ITS? Cognitive psychologists (e.g. [14] and [20]) suggest that expert knowledge is in schemata form and that schemata are induced or abstracted from many specific examples [15], [9]. To reflect these finding, the case representation language must offer an abstraction mechanism for generalizing different cases. Conceptual graphs [19] provides such a mechanism.

2 On Indexing Case Features

In case-based tutoring environments, the learning activities are designed around a specific situation, i.e. by tackling problems specified within particular contexts. It is expected that the students acquire their generic knowledge structure through generalizing the cases they have experienced and the knowledge can be transferred to a larger class of similar and yet novel problems. It, therefore, follows that case representation is critical and operations on the representation must support indexing, retrieval and comparison or adaptation.

Superficially, encoding the problem context and the solution methods seems to constitute the necessary case components adequately but the system may be held back when facing an unusual user request which is beyond the scope of the indexes chosen by the system developer. In our subject domain of Newtonian mechanics, a typical mechanics case such as "finding the acceleration of a block resting on an inclined plane" may be indexed under the surface feature - a block resting on an inclined plane, as well as under the deep feature - Newton's second law of motion. Indexing cases in this way is also in accord with the 'mainstream' approach (i.e. problem description, solution methods and outcomes) delineated by [12].

The cases can be used in two ways within a tutorial setting: (1) as worked examples by providing detailed solutions or (2) initially hiding the answers and helping the students to discover it via a guidance procedure. However, supposing a student is attempting to solve a problem that is about a skier rushing down a ski slope. In actual fact the worked example just mentioned is entirely relevant to the present problem but its tutorial utilities will be lost if the system is unable to recognize the similarity between the stored case (i.e. a block resting on an inclined plane) and the student's current problem (i.e. a skier rushing down a ski slope). Better still, the system should be smart enough to classify the case as relevant when facing 'unexpected' requests such as "a boy plays on a playground slide" or "resolution of vector into two perpendicular components", etc.

Addressing this problem demands the system to be capable of manipulating the cases at different levels of abstraction and relieving the developers from 'guessing' what sort of future situations the case can be considered as relevant. A case not only represents the application of general knowledge (say Newton's second law of motion) in a specific context (finding the acceleration of a block resting on an inclined plane), it should also allow the detection of cross-contextual similarities across difference cases.

3 Benefits of Representing Cases with Conceptual Graphs

We are not the pioneers (e.g. [16]) of exploiting the advantages of representing cases with graphs. In [16], the case details are encoded using directed graphs but directed graphs can only denote binary relationships and are therefore inadequate or cumbersome in representing relationship among three or more objects. Conceptual graphs (CG) was chosen as the knowledge representation scheme not only because of its linguistic and logical basis, but also because of its separate consideration of the relation node allows us to represent n-ary relationship among objects. Fig.2 below shows an example 3-ary relations between three concept nodes in our subject domain.

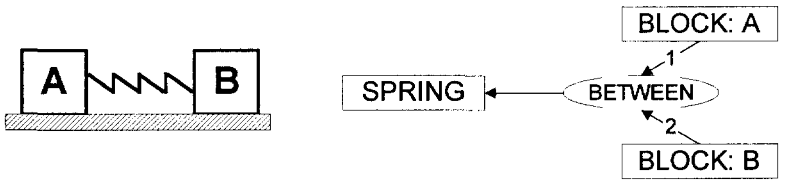


Fig. 1 A 3-ary relationship among 3 concept nodes

The graph denotes the physical configuration of 3 objects: a spring being placed between two blocks with referent labels 'A' and 'B' respectively. In linear format, it is represented as:

```
[SPRING] ← (BETWEEN) -
        1 ← [BLOCK: A]
        2 ← [BLOCK: B]
```

Another advantage of using CG is the global concept type hierarchy and attaching concept nodes with referent have elegantly represents concept subsumption and concept instantiation. The explicit use of 'inst-of' and 'isa' links in directed graphs becomes unnecessary.

4 Cases: Instantiation of Knowledge Schemata

One central theme of case-based learning is learning by induction, i.e. the generic knowledge structures, namely schemata in psychological terms, are induced from multiple cases. This assertion implies a plausible correspondence between the problem representation of the cases. If we intend to teach the students a physical principle via studying cases or solving problems, the respective cases must be derivable from the schema representing the law in generic terms. Any computer emulation of case-based tutoring must thus contain the following features:

1. The domain knowledge involved in the cases can be represented as semantic schemata
2. There are algorithms to find correspondences between abstract structures

If we are representing cases with graph structures, the first feature requires a specialization and generalization mechanism on the structures and the second one builds on the fact that the structures are computational. These notions have corresponding elements in the CG model.

4.1 Specialization and Generalization of Conceptual Graphs

What does it mean to assert that two problems, say P1 and P2, are instantiation of a physical principle? Generally, physics principles like Newton's laws of motion or conservation of energy are abstract descriptions of how the physical world behaves. When a problem solver successfully applies a particular physics principle to solve a problem, this implies that by instantiating the physics principle he/she can make sense

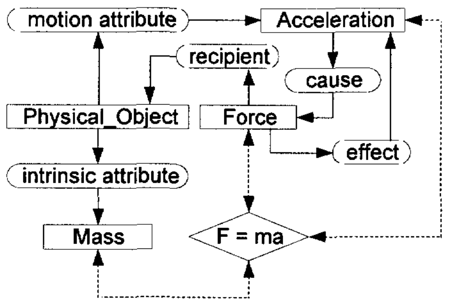


Fig. 2 The canonical graph for Newton's 2nd law

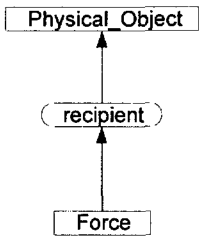


Fig. 3 A typical schema

of the problem situation and is able to draw inferences. In the CG perspective, the case graph is derivable from the well-formed graphs, namely canonical graphs, that representing the modelled world in an abstract sense. The canonical graph for Newton's second law of motion (i.e. whenever a force is exerted on a physical object the object accelerates) is depicted in Fig.2. Fig.3 illustrates the schema for the general situation of a force acing upon an object.

For the cases that were encoded using CG, as, for example, the one shown in Fig. 4, one can see that it is a specialization of the generic principle denoted by the graph in Fig. 3. In other words, Fig. 3 is both an abstract description of and a generalization of Fig. 4. Supposing we want to retrieve some cases about "a force acting on an object" for the students to practise. This task has been transformed to one of matching the target graph, i.e. Fig. 3, with the case graphs stored in the case library. By all means the case of Fig. 4 is matchable in this circumstance and can be retrieved for further processing. Of particular importance in this approach is that the explicit effort of indexing the case becomes unnecessary because the case features have been implicitly embedded.

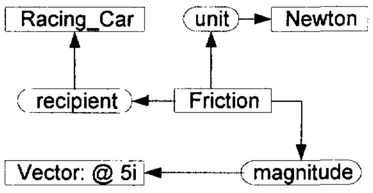


Fig.4 A typical case

4.2 Schema Induction: Finding Common Generalization across Cases

In theory, generalization can occur with just one case but this seldom happens in reality. Teaching experience informs us that most students need to be exposed to multiple cases before a generic knowledge structure can be induced. So the question is: what constitutes a correspondence between two (or more) problem representations? It is based on the notion of common generalization of the CGs representing the corresponding problems. As there can be more than one possible generalization, all these generalizations can be ordered in a generalization hierarchy.

The minimum common generalization (MCG) in the generalization hierarchy represents both situations in the least abstract way. The more extensive the MCG between two cases, the more features they have in common. In our application domain, mechanics, wherever we say that two problems can be solved by a similar physics principle, the MCG of the problems must consist of one of the canonical graphs representing the classic laws such as Newton's laws of motion, conservation of energy, conservation of momentum, and so forth.

Besides having a thorough understanding of domain specific principles, expert problem-solvers are often endowed with some powerful domain independent problem-solving strategies such as the principle of decomposition or the principle of invariance. Canonical graphs representing these principles are also potential candidates for the MCG graphs. All of these are being considered as tutorial primitives. For real cases which often comprise more than one tutoring primitive, the attractiveness of using CG lies with the integration of all primitives into one single coherent unit (see Fig. 5) and obviously the more extensive a case graph is, the more tutoring primitives it may contain.

It seems that everything discussed is relating to the domain of physics, but in fact some primitives are domain independent and can be used in many other areas. Consider the following example: our current learning objective concerns the exponential increase of some entities after a certain period of time. The system can retrieve cases ranging from the domain of banking (principal plus compound interest) to ecology (growth of fish population). That is why we claim that CG-based case representation allows cross-contextual matching and retrieval. The case developer does not need to speculate on what future scenarios the cases can be utilized as the indexing vocabulary had implicitly been encoded in the graph. Essentially speaking, each node is an index by itself and all other [concept]→(relation)→[concept] tuples form another index class.

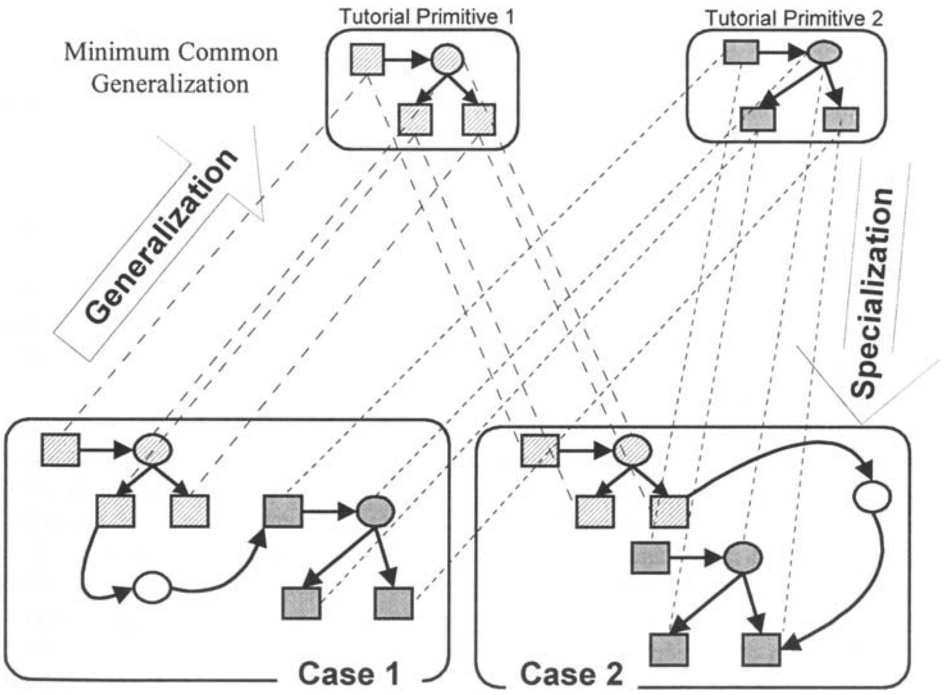


Fig. 5 A generic framework for embedding primitives into cases

5 An Operational Semantics for Conceptual Graphs

In the previous sections, we addressed the issue of retrieving cases based on the notion of matching case graphs with the tutorial primitive graph. The next question is: "Now we have a case, but so what?" The pedagogical utilities of a case are very limited if the system is not endowed with the ability to convey the knowledge embedded in the case to the users. A typical physics case is generally set in the context of problem-solving with some initially given physical quantities. During the process, the knowledge base of the problem-solvers is invoked and inference procedures are executed to derive the unknown physical quantities sought. To be useful in a tutoring context, a case should contain the knowledge components which reveal such a process explicitly to the users. In our testing domain of mechanics, mathematical computation plays a significant role so merely consider symbolic relationships among objects is inadequate. Besides the usual concept and relation nodes, we also need to consider the third class of nodes, the *actors*, described in [19]. Although our problem-solving cases can be faithfully represented by CG, the construction of the graph remains in a 'black-box' so far and the world model is represented declaratively. That is to say the user has no way of inspecting the internal processes to see how the graph is constructed. To be useful in a tutorial context, the knowledge components of a case should be made available to the students. By available, we mean the system should be able to justify each problem-solving step to be shown to the students. In other words, the procedural mechanism of achieving the solution must be made explicit but the issue is how can this mechanism be defined and executed within a graph-structured framework. In Sowa's original formulation, the actor nodes are an attachment of a CG, and the group of actors constitute a *data flow graph*. The execution mechanism of the data flow graph is like the firing process in a Petri Net [13]. In the semantics to be defined here, the CG and the data flow graph are synthesized into one single global graph, instead of treating them as two separate graphs. A case has three types of nodes: *concept* nodes, *symbolic relation* nodes and *mathematical relation* nodes (called actors in [19]).

Two assumptions were made in making the synthesis. First, human cognitive functions in studying a concrete case are viewed as a graph construction process. Relevant concept nodes are created and linked to each other via some appropriate relation nodes (symbolic or mathematical). A case represented by a graph consists of sets of concept nodes and relation nodes, but to what extent the students grasp the graph contents remains unknown until some observable actions are seen. The second assumption is a computational perspective on the graph building process and is based on the notion of concept node *marking*. Initially, the sets of nodes in a case are all transparent to the users because they are not yet marked. The set of nodes representing the initially given physical quantities are marked first. Each problem-solving step is viewed as generation of new graph nodes, but they are implemented as the nodes states change from unmarked to marked. To mark a set of nodes, the mathematical relation nodes (simply called operators) which link the marked and the unmarked nodes have to be *fired*. The procedures of solving the problem are defined as the firing sequence for marking the target concept nodes. The subgraph associated with a particular fired node represents the semantics of the knowledge behind its firing.

The semantics is derived from Sowa's original formulation and from a class of high-level Petri Nets called Predicate Transition Networks (PTN) [8]. In PTN, each input place can carry a number of pre-defined typed object called *tokens*, which are said to be marked whenever the tokens reside in the place. When all the input places of a transition are marked (a condition which *enables* the transition), the transition can be fired at any time and the token will be deposited to its corresponding output place(s) after the firing. This token-fire notion is borrowed for our model as there are a lot of similarities between a net and a graph. Whenever a concept node is instantiated to a specific individual, it is said to be marked. Firing a mathematical relation node (i.e. executing the built-in mathematical operations) will put a token (a referent for a concept node) into the concept node which is being pointed at by the head of the directed arc. The formulation may be considered as a remnant of the classic general problem-solver GPS [4] which considers problem-solving as a process of searching the problem space. The initial state of the problem is described by a set of marked concept nodes, and the goal is to mark the sought concept nodes, while the set of situations on the way to the goal are the intermediate states. Every time an operator is fired and marks new concept nodes, it is like invoking a problem-solving operator to transform the problem state from one state (an old marking) to another (a new marking).

5.1 Formal Definition of Case Contents

Definition 5.1.1: A Case $\mathcal{C}$ consists of:

1. A finite, directed graph, defined by the tuple $<C, R, R_m, E, E_m>$;
2. C, R and R_m are three disjoined sets of vertices (i.e. C ⬆ R = ĕ ; C ⬆ R_m = ĕ ; R ⬆ R_m = ĕ and C ⬆ R ⬆ R_m = ĕ) where C represents the set of concept nodes, R represents the set of symbolic relation nodes, and R_m represents the set of mathematical relation nodes (or simply operators);
3. E is a set of directed arcs, each arc connecting a concept c ï C to a symbolic relation r ï R or vice versa, i.e. E í (C à R) ğ (R à C) ; and
4. E_m is another set of directed arcs, each arc connecting a concept c ï C to a mathematical relation r_m ï R_m or vice versa, i.e. E_m í (C à R_m) ğ (R_m à C).

Definition 5.1.2: Input/Output Sets, Arcs and Concepts.

1. There are four functions: two *input* functions I_c , I_r and two *output* functions O_c and O_r.
2. I_c : C ® R_m and O_c : C ® R_m are mappings from concepts to sets of mathematical relations.

3. I_r : R_m ® C and O_r : R_m ® C are mappings from mathematical relations to sets of concepts.
4. For each concept *c* ï *C*, an input set $I_c(c)$ and an output set $O_c(c)$ are defined as:
   - $I_c(c)$ = { r_m ï R_m ♦ (r_m , *c*) ï E_m }; (r_m , *c*) is called the input arc of *c*, and r_m is called the input mathematical relation of *c*.
   - $O_c(c)$ = { r_m ï R_m ♦ (*c* , r_m) ï E_m }; (*c* , r_m) is called the output arc of *c*, and r_m is called the output mathematical relation of *c*.
5. For each mathematical relation r_m ï R_m, an input set $I_r(r_m)$ and an output set $O_r(r_m)$ are defined as:
   - $I_r(r_m)$ = { *c* ï *C* ♦ (*c* , r_m) ï E_m }; (*c* , r_m) is called the input arc of r_m, and *c* is called the input concept of r_m.
   - $O_r(r_m)$ = { *c* ï *C* ♦ (r_m , *c*) ï E_m }; (r_m , *c*) is called the output arc of r_m, and *c* is called the output concept of r_m.

Definition 5.2.3: Marking of Concept Nodes

1. For every *c* ï *C*, if *referent(c)* ← * and *referent(c)* ï *I*, i.e. *c* is being instantiated to a specific individual by having an individual marker, then *c* is *marked.*
2. A marking *m* of a graph *G* = < *C* , *R* , R_m , *E* , E_m > is a function mapping from the set of concepts *C* to the set of Boolean variables. i.e. *m* : *C* ® *Boolean.*
3. The marking *m* can be represented by a *n-vector*: *m* = $(m_1, m_2, \ldots, m_n)$, where *n* = ♦*C*♦ and each m_i ï { *T* , *F* }, *i*= *1, 2, ... , n.*
4. A marked graph *M* = < *G* , *m* > is a graph with marking *m*.

Definition 5.2.4: Enabling of Mathematical Relations

A mathematical relation node r_m ï R_m is *enabled* whenever each concept *c* ï $I_r(r_m)$ is marked. For the case graph in Fig.9, only r_{m1} is enabled at that marking.

Definition 5.2.5 Firing of Mathematical Relations

1. When a mathematical relation node is enabled, it can be fired at any time.
2. Every time a mathematical relation is fired, every *c* ï $O_r(r_m)$ will be marked.
3. For every *c* ï $O_r(r'_m)$, where r'_m is the fired mathematical relation, the referent of *c* is evaluated according to the formula(e) inscripted in the respective r'_m ï $I_c(c)$.
4. The definition of firing is different from the original PTN formalism in that no token is removed from the input concepts, because it makes little sense in our application of problem solving that a specific concept will become a general concept again after the firing.

An example case graph is shown in Fig. 6 in which only C_1 and C_5 are marked as initial conditions. At this marking, only r_{m1} can be fired to mark C_2 but either r_{m2} or r_{m3} can be fired subsequently.

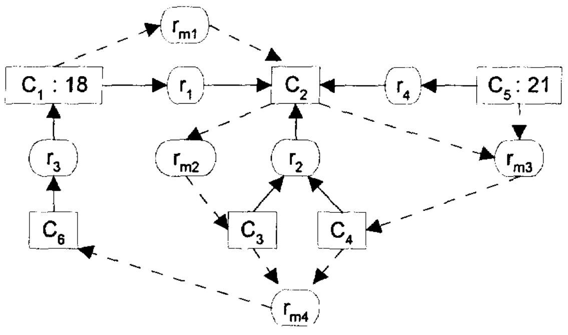


Fig. 6 A marked case graph with marking $m = (T, F, F, F, T, F)$.

5.2 Modelling Data-driven Tutoring

Having defined the essential constructs in representing procedural knowledge, we are in a position to show how they can be used in modelling different styles of reasoning. According to the finding reported in [18], novices tend to adopt a working back strategy, whereas more competent problem-solvers usually opt for a working forward strategy. A working forward approach operates from the initial set of data given by the problem statement, successively invoking equations which can be solved with the givens until the goal of finding values of the target physical quantities can be achieved. Let us attempt to capture such a mode of reasoning within our framework by referring to Fig. 6. In this case, we have six concepts (i.e. C_1, C_2, ..., C_6) which relate to each other symbolically via r_1 to r_4 and mathematically via r_{m1} to r_{m4}. The values of C_1 and C_5 are given at the beginning. The goal of the problem is to find out the value of the concept C_6. From the initial marking, we can only fire r_{m1} [giving (T, T, F, F, T, F)]. After firing, r_{m1}, r_{m2} and r_{m3} become enabled and therefore there are two alternatives available [i.e. firing r_{m2} gives (T, T, T, F, T, F) or firing r_{m3} gives (T, T, F, T, T, F)]. This process can be continued until a marked C_6 is obtained (i.e. the goal of the problem-solving activities). Fig. 7 below depicts the whole process of creating successive markings. Even this simple example indicates the student can gain access to quite a large solution space for him/her to explore but in the mean time the tutor can keep track of what can/cannot be done.

$$\begin{tikzcd}
 & \mu^{0} = (T, F, F, F, T, F) \arrow[d, "r_{m1}"'] & \\
 & \mu^{0,r_{m1}} = (T, T, F, F, T, F) \arrow[dl, "r_{m2}"'] \arrow[dr, "r_{m3}"] & \\
\mu^{0,r_{m1},r_{m2}} = (T, T, T, F, T, F) \arrow[d, "r_{m3}"'] & & \mu^{0,r_{m1},r_{m3}} = (T, T, F, T, T, F) \arrow[d, "r_{m2}"'] \\
\mu^{0,r_{m1},r_{m2},r_{m3}} = (T, T, T, T, T, F) \arrow[d, "r_{m4}"'] & & \mu^{0,r_{m1},r_{m3},r_{m2}} = (T, T, T, T, T, F) \arrow[d, "r_{m4}"'] \\
\mu^{0,r_{m1},r_{m2},r_{m3},r_{m4}} = (T, T, T, T, T, T) & & \mu^{0,r_{m1},r_{m3},r_{m2},r_{m4}} = (T, T, T, T, T, T)
\end{tikzcd}$$

Fig. 7 A complete marking tree showing the data-driven reasoning process

This section only shows how forward reasoning can be modelled but in fact it is much more powerful and flexible. Readers can refer to [5] to see how other problem-solving approaches can be modelled as well. These include goal-driven (or backward reasoning), bi-directional reasoning and hierarchical reasoning.

6 CLASP: A Case-based Learning Assistant System in Physics

At this stage, a prototype called CLASP, has been developed to test the idea. Thus far only two types of activities associated with examples have been identified: providing solutions for studying, and exercises with answers; hence the modes of interaction in the CLASP prototype are also designed around these two themes. When the users issue a request (in terms of the problem description of their own problems) the system will search through its whole case library and provide them cases which match their request. The style of presenting the case will follow the user's wishes, but only two modes of interaction (solution studying and guided-problem-solving) are available. This is to reflect the common way of using examples in physics textbooks.

In the study mode, the system presents the whole case (i.e. both the problem and solution statements) for the user to study. This looks like an electronic reference book and the student may browse through the relevant cases. However, a special agent called *the case-questioner* was developed to question the users on the contents of the examples. The motivation of questioning is to promote self-explanation [2] by the students on the example solutions. In the guided-problem-solving mode (the term 'interactive mode' was used), the system only presents the problem situation to the users, but appropriate system guidance will be provided in solving the problems. Schematically, the overall architecture of the current edition of the CLASP prototype is summarized in Fig. 8. The students interact with the CLASP system with the support of the back-end knowledge base.

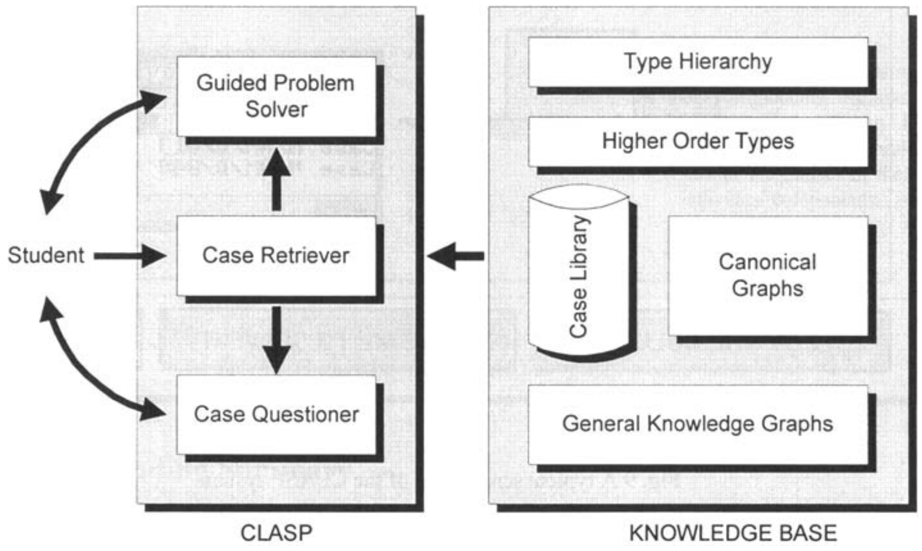
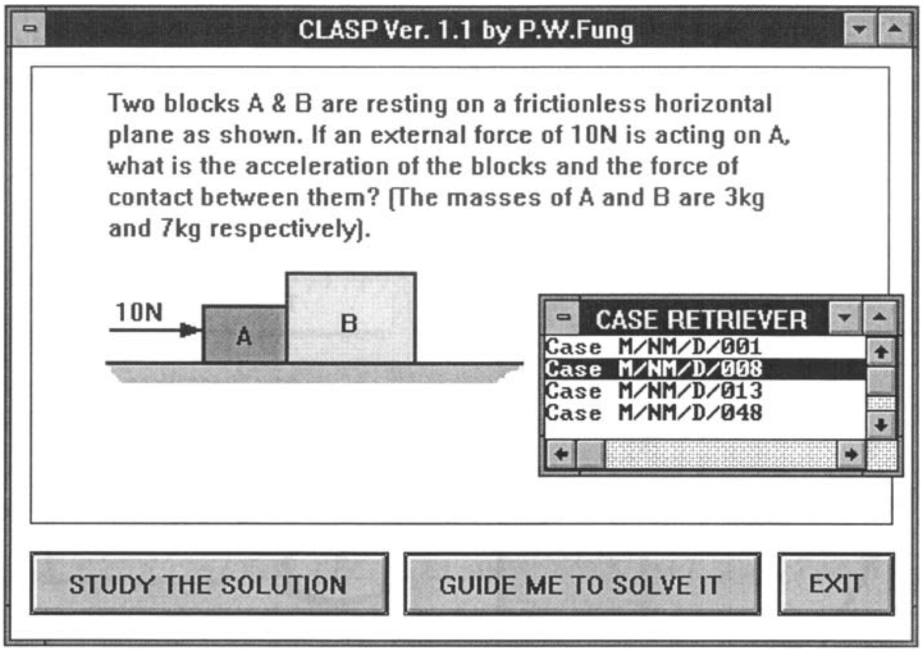


Fig. 8 Schematic description of the CLASP architecture

The deliberate separation of the knowledge base from the main CLASP system enables the flexibility and convenience of adding new knowledge, new cases, new type definitions and so forth without needing to know how the system functions. Different inference engines are developed separately and embedded in the guided-problem-solver, the case-retriever and the case-questioner respectively. These modules have independent rights of access to the knowledge base.

Once a particular case is selected, the case's problem statement will be displayed (see Fig. 9 for an example). At this stage of development, the user has three choices: studying the complete solution (i.e. study mode); asking the system to guide him/her to solve the problem (i.e. interactive mode); or to leave the system.



Two blocks A & B are resting on a frictionless horizontal plane as shown. If an external force of 10N is acting on A, what is the acceleration of the blocks and the force of contact between them? [The masses of A and B are 3kg and 7kg respectively].

Case M/NM/D/001
Case M/NM/D/008
Case M/NM/D/013
Case M/NM/D/048

Fig. 9 A typical screen shot of the CLASP system

7 Guided Problem Solving

In CLASP, problem-solving is modelled as a graph search. When a problem situation, such as the one in Fig. 9, is encountered, the initial data are represented as concept nodes being instantiated to specific values and they are displayed to students on the working pad (Fig. 10). Now the problem-solver can start tackling the problem by searching through the graph and seeing what additional information can be inferred from the initial given data. For the system to perform the tasks, the expertise has already been encoded in the conceptual graphs, therefore the next step to be taken is modelled by searching the graph to find out which operators can be fired. The inferred steps may be unfolded or kept hidden for a while as a hint to advise the student. The intelligence of the system's problem-solving ability comes from its inference engine, being implemented as different graph search methods.

The explanatory capability of the system comes from the matching of the input-operator-output nodes with the consequences of the general knowledge graphs. Whenever an operator is fired, the associated nodes will be matched against the consequences of the general knowledge graphs. If one is found, and it should be, then that particular graph will be tagged. If the student requests a justification of the step taken, the system can explain the graph in general terms. For example, the firing of an algebraic summation operator on the values of masses of two physical objects will match the consequence of the general knowledge graph in Fig. 8 so the whole graph can be retrieved for explanation. The working pad, showing the problem space, and the explanation combinations supply the integration of what and why the step happened and the whole process becomes transparent to the student.

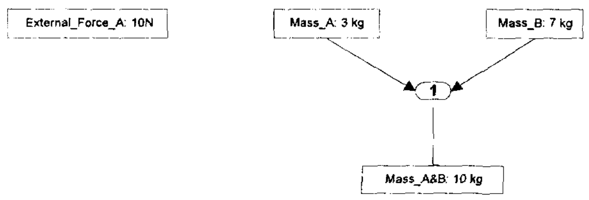
Problem Space	System's Comments:
 External_Force_A: 10N Mass_A: 3 kg Mass_B: 7 kg 1 Mass_A&B: 10 kg	Step 1: For a system comprising two components, the mass of the whole system is evaluated by the algebraic sum of the masses of their individual components. $M_{A\&B} = M_A + M_B$

Fig. 10 The problem space and the system's comments

8 Concluding Summary

CGs have been widely used in other knowledge-based applications but very few in building instructional systems. The work reported here may serves to strengthen the utility of CG in this area. The contribution of the project can be roughly divided into two categories: theoretical and application. On the theoretical side, we have developed an alternative perspective on matching cases in case-based reasoning. What constitutes the case contents was also formally defined and an operational semantics was defined on a combined structure between a CG and a data flow graph. Regarding the application achievements, the operational semantics was used in implementing four tutoring methodologies. The methodologies are very powerful in guiding students solving problems. Some other features of the system were not mentioned due to space limitation. They include generating different categories of questions from a case graph [7] to promote self-explanation from the students. The model proposed in this paper can also perform qualitative reasoning [6] and causal order between system variables can be represented succinctly. In the short term, we plan to test run the system with evaluation from students. The issue of complexity was temporary sidelined in developing the graph matching algorithm because of the initial objectives of the project were mainly educational. Once the students are satisfied with the tutoring capability of the system, we will look into the issue of efficiency before deploying it in the campus-wide network.

References

1. Chase, W.G. & Simon, H.A. (1973). Perception in Chess. *Cognitive Psychology*, Vol. 4, pp. 55-81.
2. Chi, M.T.H., Bassok, M., Lewis, M.W., Reimann, P. & Glaser, R. (1989). Self-Explanation: How Students Study and Use Examples in Learning to Solve Problems, *Cognitive Science*, Vol. 13, pp.145-182.
3. Collins, A., Brown, J.S. & Newman, S. (1989). Cognitive Apprenticeship: Teaching the crafts of Reading, Writing and Mathematics. In L.B. Resnick (Eds.*). Knowing, Learning and Instruction: Essays in honours of Robert Glaser*. Hillsdale, NJ: Lawrence Erlbaum Associates.
4. Ernst, G.W. & Newell, A. (1969). *GPS: A Case Study in Generality and Problem Solving.* NY: Academic Press.
5. Fung, P.W. (1996). *Designing an Intelligent Case-based Learning Environment with Conceptual Graphs*. PhD Thesis, Dept. of Computation, UMIST, England.
6. Fung, P.W. (1997). Generating Qualitative Predictive Problems in a Case-based Physics Tutor. In B. du Boulary & R. Mizoguchi (Eds.) *Artificial Intelligence in Education; Frontiers in Artificial Intelligence and Applications Vol. 39*. Amsterdam: IOS Press.
7. Fung, P.W. & Adam, A. (1996). Questioning Students on the Contents of Example Solutions. In *Proceedings of European Conference on Artificial Intelligence in Education*, Lisbon.
8. Genrich, H.J. & Lautenbach, K. (1981). System Modelling with High-Level Petri Nets. *Theoretical Computer Science*, Vol. 13, pp. 109-136.
9. Gick, M. & Holyoak, K.J. (1983). Schema Induction and Analogical Transfer. *Cognitive Psychology*, Vol. 15, pp.1-38.
10. Glaser, R. & Chi, M.T.H. (1988). Overview. In M.T.H. Chi, R. Glaser & M.J. Farr (Eds.) *The Nature of Expertise*. Hillsdale, NJ: Lawrence Erlbaum Associates.
11. Hayes, J.R. (1985) Three Problems in Teaching General Skills. In J. Segal, S. Chipman & R. Glaser (Eds.), *Thinking & Learning, Vol.2*. Hillsdale, NJ: Lawrence Erlbaum Associates.
12. Kolodner, J. (1993). *Case-Based Reasoning.* San Mateo, CA: Morgan Kaufmann Publishers.
13. Peterson, J.L. (1981). *Petri Net Theory and the Modeling of Systems.* Englewood Cliffs. NJ: Prentice-Hall Inc.
14. Rumelhart, D.E. (1980). Schemata: The Basic Building Blocks of Cognition. In R. Spiro, B. Bruce & W. Brewer (Eds.), *Theoretical Issues in Reading Comprehension.* Hillsdale, NJ: Lawrence Erlbaum Associates.
15. Rumelhart, D.E. & Norman, D.A. (1981). Analogical Processes in Learning. In J.R. Anderson (Ed.), *Cognitive Skills and their Acquisition.* Hillsdale, NJ: Lawrence Erlbaum Associates.
16. Sanders, K.E., Kettler, B.P. & Hendler, J.A. (1997). The Case for Graph-Structured Representations. In D.B. Leake & E. Plaza (Eds.) *Case-based Reasoning Research and Development. Lecture Notes in Artificial Intelligence, Vol. 1266*, pp. 245-54. Springer-Verlag.
17. Schank, R.C. & Cleary, C. (1995). *Engines for Education.* Hillsdale, NJ: Lawrence Erlbaum Associates.
18. Simon, D.P. & Simon, H.A. (1978). Individual Differences in Solving Physics Problems. In R. Siegler (Ed.), *Children's Thinking: What Develops?* Hillsdale, NJ: Lawrence Erlbaum Associates.
19. Sowa, J. (1984). *Conceptual Structures: Information Processing in Mind and Machine.* Reading, MA: Addison-Wesley.

20. Thorndyke, P.W. (1984). Applications of Schema Theory in Cognitive Research. In J.R. Anderson & S.M. Kosslyn (Eds.), *Tutorials in Learning and Memory.* San Francisco: Freeman.
21. Voss, J.F. & Poss, T.A. (1988). On the Solving of Ill-Structured Problems. In M.T.H. Chi, R. Glaser & M.J. Farr (Eds.), *The Nature of Expertise.* Hillsdale, NJ: Lawrence Erlbaum Associates.