

*This paper will appear as Chapter 3 in AI and Automation (Bourbaki ed.), pp. 28-47, October, 1995.*

## **ON THE DEEP STRUCTURES OF WORD PROBLEMS AND THEIR CONSTRUCTION**

FERNANDO GOMEZ  
Department of Computer Science  
University of Central Florida  
Orlando, Fl 32816  
USA

### **ABSTRACT**

The relation between problem-solving and comprehension is studied. It is shown that word problems have a deep structure, which needs to be revealed if these problems are to be solved in a principled manner. It is also indicated that the comprehension of a word problem is not a frame-based process, but rather is a constructive process by means of which the deep structure is built from the sentences describing the problem, in a way similar to the construction of a house from the building materials. This constructive process is done by means of integration rules, which build the deep structure on the basis of the semantics of the logical forms of the sentences and on the deep structure under construction.<sup>1</sup>

---

<sup>1</sup>This paper is an extended abstract of a technical report with the same title.

# 1 Introduction

This paper deals with an aspect of the problem of automating the comprehension of expository texts, namely understanding word problems. It will be shown that problems have a *deep structure* similar to the deep structure of a sentence, and that the task of the comprehender is to reveal and build this deep structure. It will be indicated that the deep structure built from the natural language description of the problem has in it all elements necessary to answer a simple information retrieval question, an arithmetic question or an algebra question. The idea of deep structure of a problem is not entirely new. It can be traced back to an earlier paper by Paige and Simon [13] in which they study the behavior of subjects solving word algebra problems and compare it to Bobrow's STUDENT program. They distinguish between "direct" and "auxiliary" translations of natural language expressions. By "direct" translation the authors refer to the direct mapping of an English sentence into an algebraic expression. For instance, the sentence *Three times a number plus 18 is equal to 78* will be directly mapped into  $3x + 18 = 78$ . This is the behavior that characterizes STUDENT and some of the subjects they studied. However, some kinds of problems require "auxiliary" representations such as the construction of diagrams that capture the physical situation described by the problem. Later in the paper, they say:

Let us propose a hypothesis. We suppose that the subject constructs an *internal representation* of the problem situation from the problem statement. We do not insist that this representation be "visual" in any literal sense, but we do require that it contain in implicit form the same relations that are implicit in the diagram (p. 273).

The authors clearly establish that some subjects use internal representations that are functionally equivalent to diagrammatic representations depicting the physical situation described by the problem. The notion of internal representation of a problem, now called *deep structure* of a problem, is further discussed in chapters 2 and 3 of *Human Problem-Solving* [11]. In chapter 3, the authors assert:

In contrast [ ... to the theories of deep structure sketched by linguists], the internal structures we shall postulate for problem solving situations generally constitute large, complex, interrelated contexts that do not factor out in any simple way into components that are isomorphic with single sentences.

It is clear that the authors maintain that their notion of internal structure goes beyond the meaning of individual sentences. The internal structure the authors postulate for problem solving is the notion of *problem-space*. One of the main points of this paper, however, will be that the internal representation of a problem stated in natural language is a structure that *mediates* between the English description of the problem and the *problem-space* representation, and that it cannot be identified with the *problem-space* representation of a problem. In our opinion, the *problem-space* representation

derives from the deep representation, which is semantically based. The *problem-space* representation underlies many problem-solving methods: logic, hill-climbing, means-end analysis, generate-and-test, etc. In a *problem-space* representation, the semantic content of the terms that make up the problem has been removed. A river is not any longer a river in a *problem-space* representation, but just a token. Paraphrasing Hilbert in his description of the formalist program for the foundations of mathematics, one may say that the meaning of the terms in the *problem-space* have been reduced to stains of ink. Even in a domain as clearly semantic as medical diagnosis [3], many problem-solving methods have been also syntactical. In most expert systems, terms such as “fever,” “inflammation,” etc., are just tokens with no meaning for the problem-solver. The mapping of a problem stated in natural language into any of these *problem-space* methods is not only a hard task, but one that in most cases is only possible if one knows a solution prior to constructing the *problem-space* representation. It is also extremely interesting that different people provide very diverse representations within a given method; e.g. different axioms in the case of logic or different operators in the case of means-ends analysis. This, in my opinion, indicates that the underlying structure of the problem cannot be identified with the *problem-space* representation. It is the internal representation what makes possible the construction of the *problem-space*.

The other main point of this paper is that the comprehension of a word problem is not a script-based process, but is a process consisting of constructing the deep structure of the problem from the logical forms of the sentences in a way similar to the construction of a house from the building materials. It is a constructive process, not filling holes in a pre-established structure. This paper is organized as follows. Section 2 argues that the comprehension of word problems is not a schema-based process. Section 3 provides examples and representations of deep structure based on the concept of *classification*. Section 4 explains the *deep structure* of motion problems based on the notion of *subevent* and other temporal links. Section 5 shows how the *deep structure* of motion problems can be constructed from the logical form of the sentence, by using integration rules. Section 6 outlines how the solution of the problem can be obtained from the *deep structure*. Finally, section 7 gives our conclusions.

## 2 Schema-Based Understanding

The AI paradigm for understanding problems stated in natural language has been the same as that for understanding narratives, namely frames, scripts or schemas. Schema-based understanding is a top-down process with two main components: schema recognition and schema instantiation. Once a relevant schema is recognized, understanding proceeds by instantiating the slots of the schema with the concepts in the sentence. Several programs have been built using the notion of schema-based understanding [12, 4, 14]. There are serious limitations with this approach to understanding problems. First of all, the recognition of the relevant schema and the actual instantia-

tion of the slots are problems for which solutions have been found only in the simplest situations. Schema-based understanding can be considered - with serious reservations - as a model of the understanding of a problem by an expert. The programs that embody schema-based understanding have compiled in them a tremendous amount of knowledge, not only about how to integrate different parts of the sentence into the schema, but also about the final representation of the problem on the basis of just a few initial clues in the first sentence of the problem. It is certainly the case that a good student who has solved a considerable number of word problems can recognize a type of problem by reading the first sentence of a problem. Yet, it is also true that the same student can be taken aback by the formulation of a problem, realizing only in the last sentences that he/she is mistaken about the type of problem. Certainly, experts are not just a compiled mass of knowledge, but they are able to deal with cases that deviate from stereotypical situations. This is true not only of problem-solving, but also of comprehension.

Although the understanding of some of these problems can be done by identifying a relevant schema and instantiating it, the comprehension of most problems can not be schema-based. There are several reasons for this. First of all, the clue identifying the relevant schema may occur too late in the problem to be of any value. Consider the following problem describing an overtake situation:

Leesburg is on the route from Orlando to Tampa. The distance from Orlando to Leesburg is 200 miles. The distance from Leesburg to Tampa is 100 miles. Peter left Orlando to go to Tampa by car at 55 mph. Mary left Leesburg to go to Tampa at the same time. She drove at 25 mph. Will Peter overtake Mary before she gets to Tampa ?

Note that until the question is posed, there has not been a good way to find out what schema this is. Perhaps, it could have been suspected in the fifth sentence, but that is too late to make any sense of the first four sentences. Redundant information is another aspect of problem comprehension which would be very hard to accommodate in a schema-based system. Children are frequently given problems with irrelevant information in order to test their problem-solving skills. These “noisy” problems pose serious difficulties for the children. The reason for this is that these problems deviate from stereotypical problems. If a child has learned a rigid schema which is good for at most three or four problems, the child is unable to match redundant or irrelevant information to the schema.

Our research on the comprehension of expository texts has centered on the investigation of understanding in the face of limited knowledge. In [5, 6], we have proposed a knowledge-lean model of comprehension. The comprehender of our model is a layman who reads the texts with limited knowledge about their content. Since the reader is acquiring new concepts, the schema-based understanding model can barely be applied in this context. Instead, comprehension is a bottom-up process by means of which new concepts are built. In our model, concepts are constructed by means of *formation*

rules, which build concepts from the logical form of the sentence. Once these concepts are built, a process, called *recognition*, is invoked in order to check if the concepts that have been built are already in long-term memory (LTM). Those concepts that remain unrecognized are passed to an *integration* phase, which is responsible for integrating them in LTM. The front end of SNOWY, the system that incorporates these ideas, is a general purpose parser, called WUP (word usage parser), based on the notion of syntactic usage of a word. WUP produces a shallow parse of a sentence, leaving the attachment of prepositions and relatives, the determination of the verbal concept and thematic roles to the *interpretation* phase of SNOWY. The *formation* phase is also a general purpose algorithm producing an output that is the same for a variety of tasks in the comprehension of expository texts (e.g. acquiring knowledge for an expert system [7], understanding an elementary scientific text, solving a word problem, etc.). The *recognition* algorithm is also the same for all these tasks. The *integration* algorithm is the only one that changes with a given task. These changes are not major ones, and are mainly due to the different way in which events, e.g. *Peter went to the theater*, and general atemporal facts, e.g. *All whales live in the sea* are integrated.

### 3 The Deep Structure of Problems

Consider the following problem:

Problem C1: All students who went to the theater went to the game.  
 Twenty students went to the theater. Did at least twenty students go to the game?

In spite of its simplicity, this problem can hardly be solved by using production rules that would match the surface terms of the sentences. The problem has an underlying structure that needs to be revealed if it is to be solved in a principled manner. The underlying structure of this problem is one of *classification* in a sense explained below. Consider the first sentence of problem C1 *All students who went to the theater went to the game*. In [8], we have shown how to represent restrictive relative clauses as classification hierarchies. The clause *Students who went to the theater* is represented, or formed, by creating a concept with a dummy name, say *x1*, but containing a slot, called *cf*, that identifies the concept by providing necessary and sufficient conditions. The representation of this concept is:

```
X1
  (cf(is-a(student) actor(x1(q(all)))) primitive(ptrans)
    theme(x1(q(all))) to-loc(theater-a)))
```

We will use the primitive *ptrans* throughout the paper to mean the transfer from a location to another location by an animate being [15]. The content of the *cf* slot says

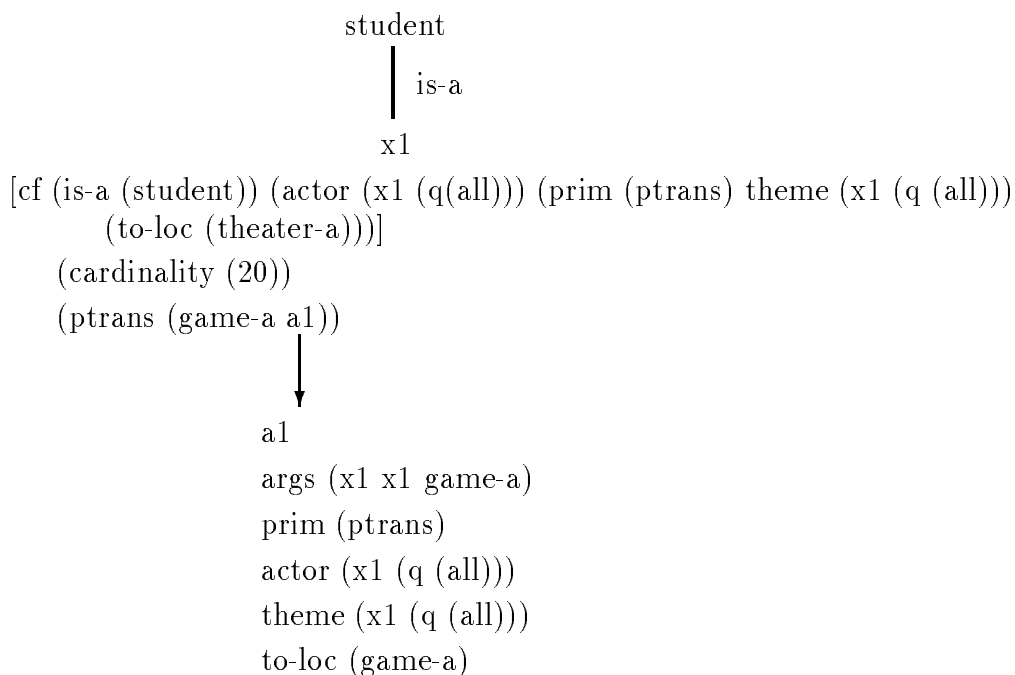


Figure 1: Deep structure of problem C1.

that  $x1$  is a subset of students who went to a theater. The precise meaning of the *cf* representation in FOPC is:

$$\forall(x)(X1(x) \iff Student(x)andR1(x))$$

where R1 is the relation “x went to the theater.” During *formation*, the name  $X1$  replaces the restrictive relative clause in the main clause, which, in our case, yields *All x1 went to the game*. This in turn is represented (formed) by building a relation predicated from all X1 as indicated in Figure 1. Concepts denoting entities (student, book, x1) are called *object-structures*. Conceptual relations denoting actions or events are represented in the *object-structures* by indicating the relation followed by the *theme* (if any) and a pointer to the representation of the action structure. In Figure 1,  $a1$  is an action structure. The conceptual relation is fully represented in  $a1$ . The action structures can be connected to other action structures. The letter  $q$  is the quantifier of the concept immediately following it. Concepts that are not followed by a quantifier are individuals or constants, e.g. theater-a, game-a, Orlando, etc. The scope of the quantifiers is from left to right. (See [8] for an explanation of the meaning of these structures using FOPC.)

Existentially quantified sentences, e.g., *Some people like music* and also sentences whose *subject* or *actor* has a numeral in it are also represented as classification hierarchies. Hence, the sentence *Twenty students went to the theater* is represented by

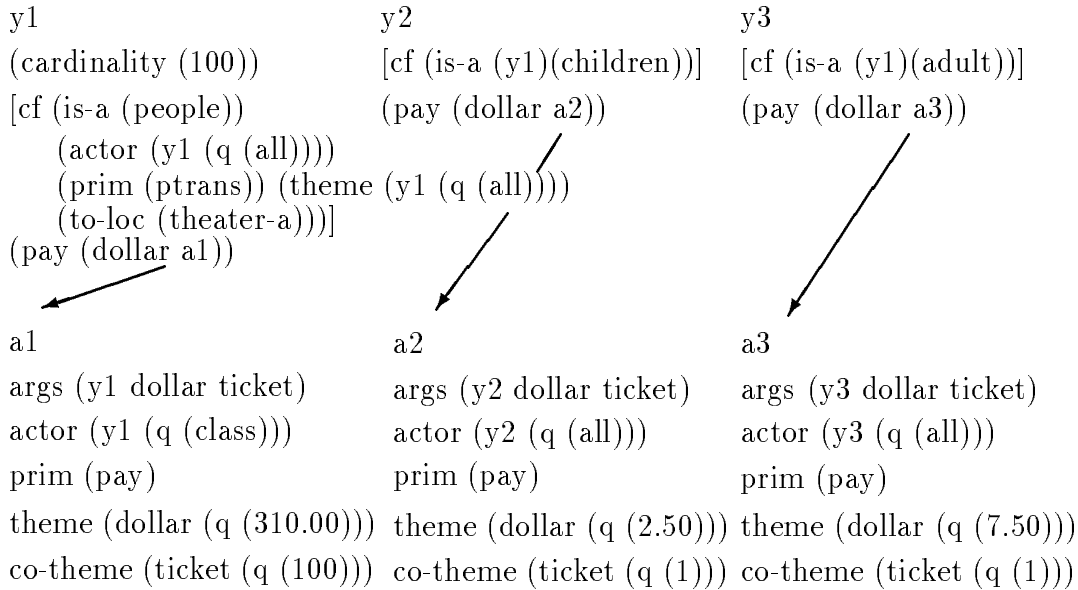


Figure 2: Structure definitions of problem C2.

creating a class, say X2, whose characteristic features are “is-a student and went to the theater,” and the cardinality of the class is twenty. Formally:

```
X2
  (cf(is-a(student) actor(x2(q(all))) primitive(ptrans)
      theme(x2(q(all))) to-loc(theater-a)))
  (cardinality(20))
```

When X2 is going to be integrated in memory, the *recognizer* algorithm is activated and realizes that X2 already exists in memory, namely the concept X1 created above. Hence, the result of integrating X2 in memory is simply to insert the slot *cardinality* with the value twenty in the concept X1. Now, the answer to the question *Did at least twenty students go to the game?* is obtained by descending to the concept X1 from the concept *student* and noticing that the quantifier of X1 is a universal quantifier and the cardinality of the class X1 is greater than or equal to twenty. Note that if the first sentence of the problem had been *Twenty students went to the theater*, the structures built would have been the same. Classification is the deep structure of many word algebra problems. Consider the problem below that is one or two levels more complex than the one just explained, but it has a similar deep structure.

Problem C2: A group of 100 people consisting of children and adults went to the theater. They paid a total of \$310.00 for 100 tickets. Each child paid \$2.50 per ticket, and each adult paid \$7.50 per ticket. How many children went to the theater?

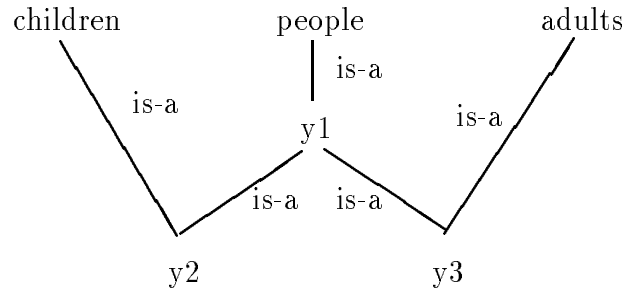


Figure 3: Deep structure of problem C2.

The representation built for this problem is depicted in Figure 2 and 3.  $Y1$  represents the concept “people who went to the theater.” Note the quantifier *class* in the relation *pay*, structure *a1*, under the concept  $Y1$ . This is a collective quantifier. Its meaning in this sentence is that they paid as a group or class \$310.00. Note that “all” or “100” are wrong ways to quantify  $Y1$  in the relation “pay.”  $Y2$  represents the concept “children who went to the theater” and  $Y3$  represents the concept “adults who went to the theater.” Figure 3 depicts the classification relations between these concepts, providing the deep structure of the problem. The answer to the question is obtained by the following algorithm.

An answer to a question of the form *How many x relation?* is obtained by searching for the quantifier of  $x$  if  $x$  is the *theme* of the relation and by searching for the *cardinality* of  $x$  if  $x$  is the *actor* of the relation. Let us consider the latter case. If  $x$  has a *cardinality* slot, the value of that slot is the answer. If  $x$  does not have it, the cardinality slot of the superconcept of  $x$  is examined. Then, the algorithm tries to obtain the answer by finding the cardinality of the brothers of  $x$  and by applying some arithmetic. That would be the case if the cardinality of the brothers of  $x$  is known. A simple subtraction would solve the problem. In this case, the algorithm tries to solve the problem by setting up an equation of two unknowns. The algorithm asks the information retrieval component to find the algebraic relations that exist between a superconcept and its subconcepts. One of these relations is that the sum of the cardinality of the subconcepts is equal to the cardinality of their superconcept. For the problem we are discussing, this will result in producing the equation “ $Y2 + Y3 = 100$ .” The information retrieval prints that equation with the following warning “I am assuming that  $Y2$  and  $Y3$  are the only subclasses of  $Y1$  and that they are disjoint classes.” The second equation that the information retrieval would produce is based on the following general rule. Let  $arg1$  denote an argument of a relation:

- If (a) the argument, say  $arg1$ , of a relation in a concept,  
     say  $x$ , is quantified with a class quantifier, and  
 (b) the same relation is predicated of all subconcepts of  $x$ ,  
 (c) and the quantifier of  $arg1$  is “all” in every

subconcept of  $x$ , then:

- (d) an arithmetic or algebraic relation exists between the superconcept of  $x$  and its subconcepts.

Once this relation is established, a procedure that matches the other arguments in the relation produces the final equation that in our case is “ $2.50 * Y2 + 7.50 * Y3 = 310.00$ .” Note that this rule is independent of the relation “pay.” For instance, similar relations will be established for problem C3 below, which is clearly *isomorphic* to the one just explained because the two have the same *deep structure*.

C3: One hundred people consisting of faculty and students ate 300 cakes. Each faculty member ate 2 cakes and each student ate 4 cakes. How many cakes did the students eat?

## 4 The Deep Structure of Motion Problems

We proceed, now, to analyze the deep structure of motion problems. Children at school are told that they must construct a graphical representation of these problems prior to trying their actual solution. For instance, consider the following problem:

Problem M1: Peter traveled from  $a$  to  $b$  through  $a_1$ . He used a train and a plane. The train going at 30 mph took 2 hours longer than the plane. The plane traveled at 150 mph. He traveled a total of 1400 miles. How long did the train take?

One possible graphical representation of problem M1:

```
train: 30 mph;duration = 2 + x  plane:150 mph;duration = x
<-----a1----->
a----- 1400 mi -----b
```

There is a considerable body of evidence indicating that many children and also adults have trouble in solving these problems because they do not understand their linguistic formulation. By asking them to build the graphical representation, they are forced to fully comprehend the description of the problem. Of course, what the diagrammatic representation does is to make apparent the deep interrelations that define the problem [10]. In other words, the graphical representation is one way to expose the deep structure underlying the problem. For instance by looking at the graphical representation above, one can *see* that it is about a motion problem describing a trip consisting of two subtrips. A question such as *How far is  $a$  from  $b$ ?* can be directly answered by looking at the figure. The answer to the question *How long did the trip by train take?* is also facilitated by the graphical representation, because it shows clearly that the distance traveled by the train plus the distance traveled by the plane is equal to 1400 miles.

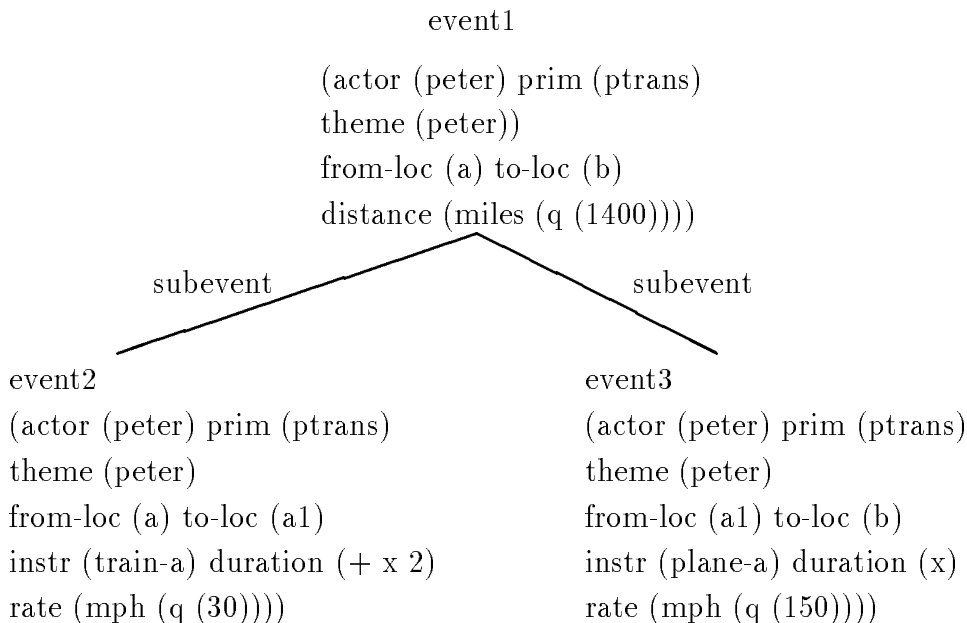


Figure 4: Deep structure of problem M1.

The diagram is a kind of mental model of the representation of the problem. The analysis of word algebra problems about motion revealed to us that if a program is to solve these problems in a principled manner, it needs to construct a representation that will be at least homomorphic to the graphical representation. The representation built by our system is a propositional representation that is homomorphic to the graphical representation. The deep structure corresponding to problem M1 is depicted in Figure 4.

The representation in Figure 4 says that problem M1 is an instance of a problem of a trip in stages, having three events. Event1 represents the event of going from place *a* to place *b*. Train-a and train-b are internally generated names that stand for instances of trains.

Event2 and event3 are linked to event1 by the relation *subevent*. Event e1 is a subevent of e2 iff the occurrence of e1 is a necessary condition for the occurrence of e2 and e1 occurs *in* the time interval of e2. The concept of *subevent* can be defined formally using Allen's temporal logic [1] as follows. First, we need the following Allen's definitions for intervals. *STARTS*(t1,t2): time interval t1 shares the same beginning as t2, but ends before t2 ends; *FINISHES*(t1,t2): t1 shares the same end as t2, but begins after t2 begins; *DURING*(t1,t2): t1 is fully contained within t2; We need also the predicate *IN*, meaning that one interval is wholly contained in another, as follows:

$$\forall(t1, t2)(IN(t1, t2) \iff DURING(t1, t2) \vee STARTS(t1, t2) \vee FINISHES(t1, t2))$$

Another of Allen's predicates is that of OCCUR. This predicate takes an event,  $e$ , and a time interval,  $t$ , and is true if the event happened over the time interval  $t$  and there is no subinterval of  $t$ , say  $t'$  *prime*, over which  $e$  is true. This is captured in the axiom:

$$\forall(e, t, t')(OCCUR(e, t) \wedge IN(t', t) \implies \neg OCCUR(e, t'))$$

Then, a formal definition for the concept of *subevent* is:

$$\forall(e1, e2)(Subevent(e1, e2) \iff \exists t1 \exists t2(OCCUR(e1, t1) \wedge OCCUR(e2, t2) \wedge NECESSARY(e1, e2) \wedge IN(t1, t2)))$$

The predicate *necessary* is an essential element of the definition of *subevent*, since potentially infinite events occur in the time interval of the event, say, *Peter drove to Tampa*. But only just a few of them are preconditions for the event *Peter drove to Tampa* to occur. Some of those could be *Peter opened the door of the car*, *Peter entered the car*, etc. The meaning of *necessary* in this definition is logic necessity. That is, "event  $e1$  is a necessary condition for event  $e2$ " means that  $e2$  does not occur unless  $e1$  occurs, or  $\neg occur(e1) \implies \neg occur(e2)$ , or, by contraposition,  $occur(e2) \implies occur(e1)$ . Thus, if one considers the representation of the event *Peter went from a to b through a1* depicted in Figure 4, event *event2 Peter went from a to a1* and event *event3 Peter went from a1 to b* are necessary conditions for the event in the root node to occur. This is so because, although Peter could have gone from  $a$  to  $b$  through many other routes, once the event *Peter went from a to b through a1* took place, events *event2* and *event3* become necessary conditions for event *event1* to occur.

The deep structure is not only a computationally efficient representation, but, more importantly, one that is required if a solution is to be found, unless a subject uses rote memorization for the solution of a problem. Of course, this does not mean that a subject needs to be aware of the deep structure of a problem when he/she is solving it. Note that the representation that has been built does not classify the type of problem as being an overtake problem, round trip problem, etc. Consider the problem:

M2: At 2 pm, a train left Tampa to go to Orlando at 40 mph. Another train left Orlando to go to Tampa at the same time at 60 mph. The distance from Tampa to Orlando is 300 miles. When will the two trains meet?

The representation built for this problem and depicted in Figure 5, consists of two events connected by the relation *init-same-time* meaning that the two events initiated at the same time. This relation does not indicate, in any way, which event is going to end or finish first.

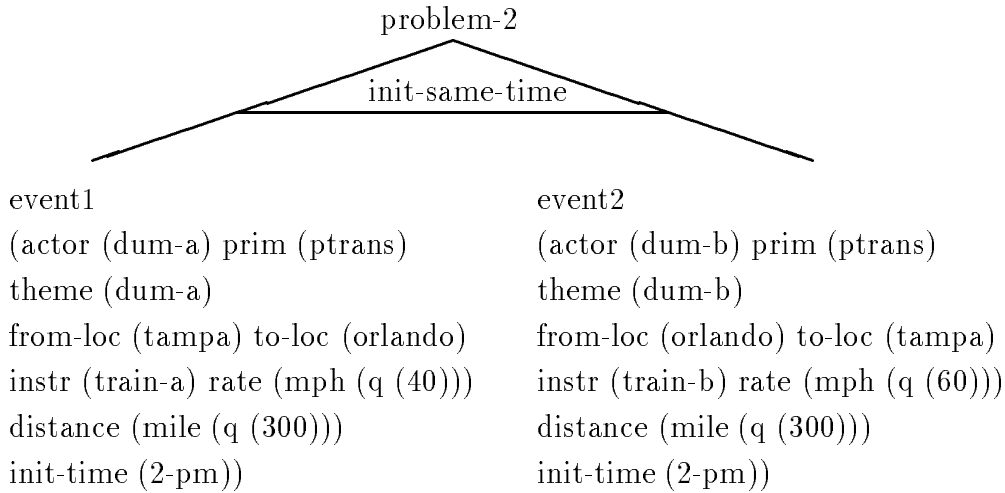
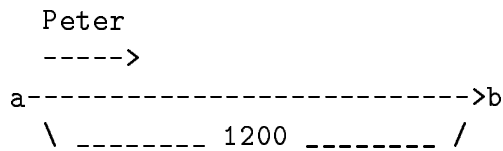


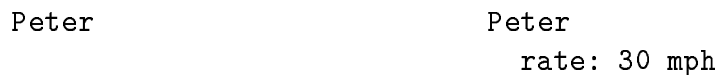
Figure 5: Deep structure of problem M2

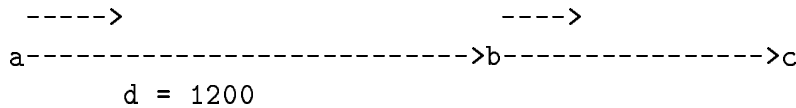
## 5 The Construction of the Deep Structure of Motion Problems

The deep structure of a motion problem is built by using *integration* rules similar to those explained for the classification problems. As for those rules, these rules access two data bases, one formed by the logical form of the sentence, and another one consisting of the deep structure under construction. These rules encode common sense knowledge about the physical constraints in the logical forms and in the representation being built. In the description of a problem, initial sentences set up a context and subsequent ones need to fit in this context. This may be illustrated with the following sentence: *Peter traveled a distance of 1200 miles from a to b*. This sentence produces the following initial representation (we use the diagrammatic representation):



Subsequent sentences need to fit in this initial representation, which already imposes constraints on their semantics. For instance, suppose that the next sentence is *Peter returned to c at 20 mph*. This sentence does not make any sense because there is not a location called “c” in the initial model, and it cannot be integrated in the representation. If the next sentence is *He continued from b to c at a speed of 30 mph*, “b” is found in the model and this representation can be produced:





The representation also deals with cases of ambiguity. For instance, suppose that the next sentence is *It took Peter two hours*. This sentence is ambiguous only in the context of the last diagram, because it has two segments to which the duration can be attached. However, the sentence is not ambiguous in the first representation that consists only of one segment. There are also rhetoric and discourse constraints imposed by the initial sentences of a problem [2].

Two types of integration rules are needed: those that *merge* an incoming event into an event already in the model and those that *reorganize* the events by connecting them with temporal links. We use DS to refer to the deep structure being built and I-EVENT to the event being integrated. The first integration rule says:

R1

If the DS is empty, then insert the I-EVENT in the model.

Then, if the first sentence read is *A train traveled from a to b at 24 mph*, the DS will contain:

```
(event1
  (actor(dum-a) primitive (ptrans) theme(dum-a) from-loc (a)
    to-loc(b) instrument (train-a) rate (mph (q(24))))))
```

Terms like dum-a, dum-b stand for a unknown *actor*, *theme*, *from-loc*, etc. We refer to them as dummy terms. Had the first sentence read been *Peter traveled 1400 miles*, then after applying rule R1, the DS would contain:

```
(event1
  actor(Peter) primitive (ptrans) theme(Peter)
  distance (miles(q(1400))))
```

Suppose that the first sentence read is *Peter traveled from a to b*. This sentence will produce the following representation in the DS:

```
(event1
  (actor (Peter) primitive (ptrans) theme (Peter)
    from-loc (a) to-loc (b)))
```

The following rule merges information from the I-EVENT into the events in the model.

R2

If the I-EVENT and an event in the DS match then

- a. remove the event from the DS
- b. create a new event by forming the union of the two events.
- c. integrate the union event in the DS.

Two events, say e1 and e2, *match* if all slots that appear in both events have identical fillers.

$$\text{Match}(e1,e2) \iff \forall(x)\forall(y) (\text{Slot}(x,e1) \text{ and } \text{Slot}(y,e2) \text{ and } \text{Equal}(x,y) \implies \text{Equal}(\text{filler}(x), \text{filler}(y)))$$

Suppose that the DS consists only of the following event:

```
(event1
  (actor(Peter) primitive (ptrans) theme (Peter)
    from-loc(Tampa) to-loc(Orlando)))
```

Now, the next sentence is: *He traveled by train.* The logical form produced for this sentence is:

```
(actor (Peter) primitive (ptrans) theme (Peter)
  instrument (train-a))
```

Rule R2 will fire since the event in the DS and this one will *match*, because every slot that appears in both events has the same filler. The union of the two events will result in the event:

```
(event1
  (actor(Peter) primitive (ptrans) theme (Peter)
    from-loc(Tampa) to-loc(Orlando) instrument (train-a)))
```

Since rule R2 has removed the only event in the model from it, the DS is empty. As a consequence, the event above will be integrated in the DS by rule R1. Let us assume that the first two sentences of a problem are *Peter went from Tampa to Orlando. He used a train and a plane.* The representation built for the first sentence is:

```
(event1
  (actor(Peter) primitive (ptrans)
    theme(Peter) from-loc (Tampa) to-loc(Orlando)))
```

The parser output for the second sentence is:

- 1.a (subject ((pron he)) verb ((main-verb use) (tense sp))  
 object ((udet a) (noun train)))
- 1.b (subject ((pron he)) verb ((main-verb use) (tense sp))  
 object ((udet a) (noun plane)))

Interpretation rules resolve the referent of “he” and create unique names for the indefinite noun groups “a train” and “a plane.” This produces:

- 2.a (subject (Peter) verb (used) object (train-a))
- 2.b (subject (Peter) verb (used) object (plane-a))

Now, it becomes necessary to determine the meaning of “used.” One of the interpretation rules for finding the meaning of “use” in a sentence consisting only of an *subject* and an *object* says: Find the most recent logical form, say LF1, containing an action primitive and an *actor* that matches the *subject* of the sentence with “use.” If the rule succeeds, as in this case, the subject of the verb “use” becomes the *actor*, the verb “use” is replaced with the primitive of LF1, and the *object* slot is changed to an *instrument* slot. This rule will produce:

3.a (actor (peter) primitive (ptrans) instrument (train-a))  
3.b (actor (peter) primitive (ptrans) instrument (plane-a))

The logical form 3.a is integrated using rule R1. However, R1 does not fire in integrating logical form 3.b. This logical form is integrated using one of the rules that inserts events in the DS and reorganizes it. In this case, the rule is:

R3

If the I-EVENT and an event in the DS match, say event e1,  
except for the instrument slots then:  
a. create a new event in the DS.  
b. connect the I-EVENT and e1 with a subevent link.

The application of this rule results in the structure depicted in Figure 6. The rule is encoding the piece of common sense knowledge that says that if somebody has traveled using two vehicles, then he/she has made a trip in stages. The rule produces the correct hierarchy with an event at the top level and two subevents connected to it. The integration rules are independent of the surface form of the sentences. One can see that the rules above will produce the same deep structure if the sentence is *Peter traveled from Tampa to Orlando using a train and a plane*. Consider the following more complex example:

Problem M3: (1) Peter went from Tampa to Orlando using a train and a plane. (2) He traveled by train from Tampa to Leesburg and (3) traveled by plain from Leesburg to Orlando. (4) He also went from Orlando to Miami.

Sentence (1) will create the structure depicted in Figure 6. The integration of sentences (2) and (3) will result in inserting “Leesburg” into the slots *to-loc* and *from-loc* of event2 and event3. This is done by using rule R2. Slots with fillers dum-a, dum-b, etc. are ignored by the matching algorithm. The integration of sentence (4) is done by the following subevent rule.

R4

If the I-EVENT and an event in the DS, say e1, have the same actor and the same primitive and the from-loc slot of the

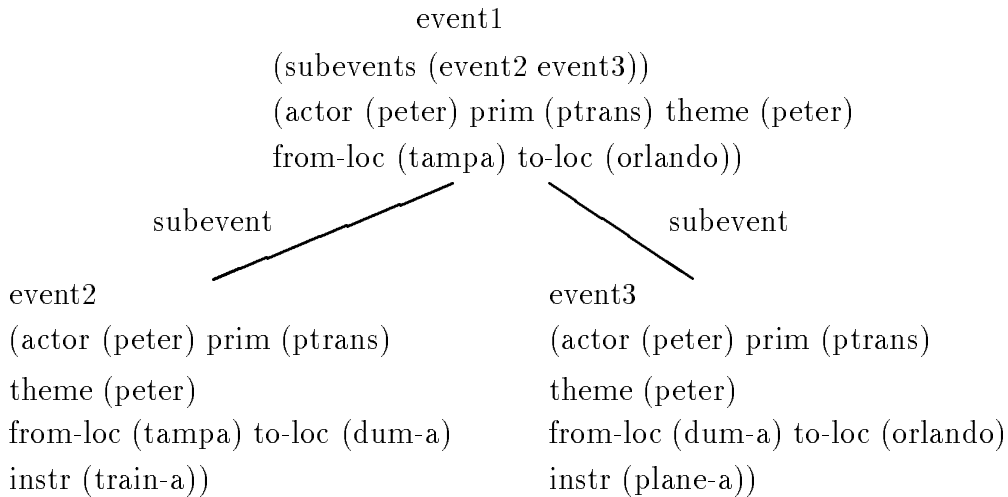


Figure 6: Structure built for “Peter went from Tampa to Orlando. He used a train and a plane.”

I-EVENT is the same as the to-loc slot of e1, then:

- a. create an superevent, say e2, whose from-loc is the from-loc of e1 and the to-loc is the to-loc of the I-EVENT
- b. one subevent of e2 is e1
- c. the other subevent of e2 is the I-EVENT

This rule is applied in a top-down fashion starting with the root node of the hierarchy of events and descending. The application of this rule stops as soon as an event in the DS matches. The structure created is depicted in Figure 7. If the sentences describing problem M3 are followed by:

The trip from Tampa to Miami took 7 hours. The trip from Tampa to  
Leesburg took 2 hours, and the trip from Leesburg to Orlando, 3 hours.

These sentences will be integrated by rule R2. The definite description “the trip” in the sentence *The trip from Tampa to Miami took 2 hours* is handled by applying rule R2 to the logic form below:

```
(actor(dum-a) primitive (ptrans) from-loc(Tampa) to-loc(Miami)
  duration(hour(q(2))))
```

Note that only one event in the DS matches. However, rule R2 will find more than one match if the sentence were *The trip took 2 hours*. In this case, the definite description cannot be resolved, and an assumption needs to be made about its reference.

If none of the rules fires for an event, that event is inserted in the DS by a catch-all rule, which inserts the event in the DS without connecting it to any of the events already in the DS. Hence, we have the rule:

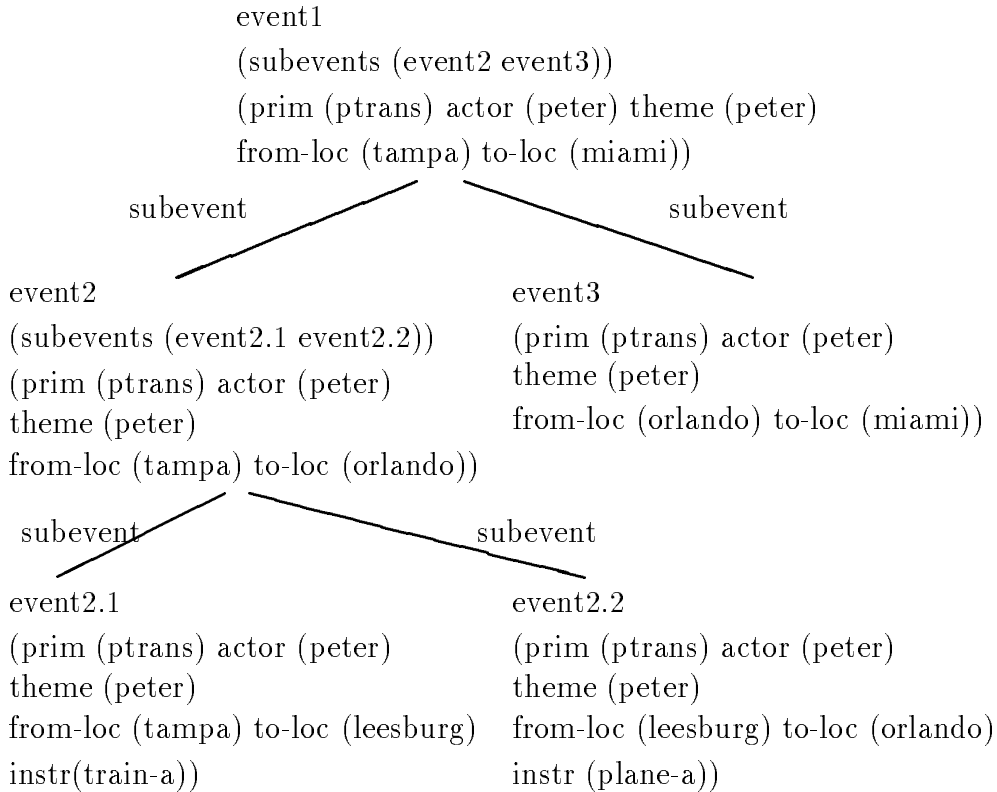


Figure 7: Representation produced for problem M3

R4

If no rule fires, then insert that event in the DS without connecting it to any of the other events in the DS.

Consider the sentences *Peter traveled from Tampa to Orlando. Mary left from Tampa toward Orlando at 2 p.m.* Let us call e1 and e2 the events produced by the first and the second sentence, respectively. Event e1 will be integrated by rule R1, while event e2 will be integrated by rule R4. No link will connect those two events in the DS. Suppose that the next sentence is *Peter left Tampa at 2 p.m.* In this case, rule R2 will fire modifying event e1 that will become:

```
(actor(peter) primitive(ptrans) init-time(2-pm) from-loc(Tampa)
  to-loc(Orlando))
```

The slot *init-time* recording the time in which the event starts has been built by rule R2. As the reader may remember, when rule R2 merges knowledge into an event, it removes that event from the model and activates the rules that connect the events in the DS in order to integrate that event. In this case, the following rule integrates the event in the DS:

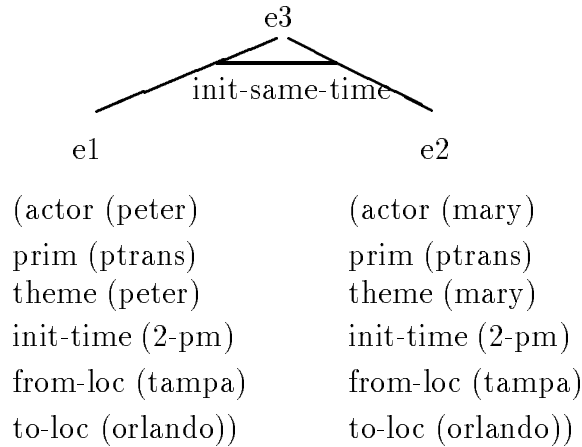


Figure 8: The representation produced for “Peter traveled from Tampa to Orlando. Mary left from Tampa toward Orlando at 2.p.m. Peter left Tampa at 2 p.m.”

R5

If the I-EVENT and an event in the DS have an init-time slot with the same filler, then:  
connect the two events with an init-same-time link.

The representation produced for this last example is depicted in Figure 8.

## 6 The solution of motion problems

The solution of motion problems represented in a hierarchy of subevents are solved in a way similar to problems whose deep structure is organized using an *is-a* hierarchy. Each type of question has an associated hierarchy of problem-solving methods that will provide an answer to it. For instance, consider the question *How long did Peter take going from A to B*. The first problem-solving method in the hierarchy tries to find an answer to the question by accessing the appropriate slot in the right event in the DS. This is simple knowledge retrieval. If this fails, a slightly more complex problem solving method that tries to apply some arithmetic knowledge is activated. This problem-solver method checks to see if the events in the DS are organized by the *subevent* relation. Let us assume that the DS consists of event2 and event3 which are subevents of event1. The problem-solver method tries to achieve the subgoal:  $\text{duration}(\text{event1}) = \text{duration}(\text{event2}) + \text{duration}(\text{event3})$ . This subgoal is attempted by directly accessing the “duration” slot in those events, that is, by activating the information retrieval method. If the arithmetic problem-solver fails because the duration of event2 or the duration of event3 is unknown, then the problem is tried by using some algebra, that is, by invoking the algebraic problem-solver method. The subgoal

activated is  $\text{distance}(\text{event1}) = \text{distance}(\text{event2}) + \text{distance}(\text{event3})$ , which is equal to  $\text{rate}(\text{event1}) * \text{duration}(\text{event1}) = \text{rate}(\text{event2}) * \text{duration}(\text{event2}) + \text{rate}(\text{event3}) * \text{duration}(\text{event3})$ . The execution of this subgoal is again tried by activating the information retrieval method, which fills the content of “rate” and “duration.” Each slot acts like a question to the information retrieval method, e.g., find the content of “rate” in event1, etc.

## 7 Conclusions

All the ideas described in this paper have been implemented. The program that deals with motion problems contains only about fourteen general organization rules in order to integrate events describing motion problems. Besides these rules, there are some integration rules which are specific to some lexical terms. For instance, the verb “return” has a special rule to integrate it. Expressions such as “upstream” and “downstream” also trigger specific integration rules.

In the body of the paper, we say that the understanding of a word problem is similar to the construction of a house from scratch. The house in our metaphor corresponds to what we have called the *deep structure* of a problem, and the building materials are the events describing the problem. In the construction of a house, there are minor actions like putting a wooden frame in a window, and major actions like setting a wall. The minor actions correspond to the actions performed by our *merge* rules, and the major actions correspond to the actions performed by our event insertion and organization rules. This is clearly in contrast with the schema-based understanding paradigm in which the house is already pre-built, and the only things left to do are minor actions.

Where do problem-spaces come from? This is one of the most fundamental questions in the field of human problem-solving. It is generally accepted that if a problem-space is provided for a problem, the solution follows without much difficulty since it reduces to search, which is a well-understood technique to which massive parallel methods, to use the favorite phrase, can be applied. However, the solution of many problems hinges not as much on *how* to search, but on *what* and *where* to search, that is, on the problem-space itself. In this paper, we have gathered some evidence showing how the problem-space for some kinds of word problems is based on and derives from the *deep structure* of the problem.

## References

- [1] J. Allen. (1984). Towards a General Theory of Action and Time. *Artificial Intelligence*, **23**, pp. 123-154.

- [2] J. Batali. (1991). *Automatic Acquisition and Use of Some of the Knowledge in Physics Texts*. Ph.D Dissertation, Massachusetts Institute of Technology, Artificial Intelligence Laboratory.
- [3] F. Gomez. (1981). *On General and Expert-Based Methods in Problem-Solving*. Ph.D Dissertation, The Ohio State University, Department of Computer Science.
- [4] F. Gomez. (1982). Towards a Theory of Comprehension of Declarative Contexts. Proceedings of the 20th Meeting of the Association of Computational Linguistics, pp. 36-43.
- [5] F. Gomez. (1985). A Model of Comprehension of Elementary Scientific Texts. Proceedings of the Workshop on Theoretical Approaches to Natural Language Understanding. Halifax, Nova Scotia, pp. 70-81.
- [6] F. Gomez and C. Segami. (1989). The Recognition and Classification of Concepts in Understanding Scientific Texts. *Journal of Experimental and Theoretical Artificial Intelligence*, **1**, pp. 51-77.
- [7] F. Gomez and C. Segami. (1990). Knowledge Acquisition from Natural Language for Expert Systems Based on Classification Problem-Solving Methods. *Knowledge Acquisition*, **2**, pp. 107-128.
- [8] F. Gomez and C. Segami, (1991). Classification-Based Reasoning. *IEEE Transactions on Systems, Man and Cybernetics*, **1**(3), pp. 398-405.
- [9] F. Gomez. (1992). Representing Biological Systems as Temporal Event Hierarchies. Technical Report, Dept. of Computer Science, UCF, Orlando, Florida.
- [10] J. Larkin and H. Simon, (1987) Why a Diagram is Sometimes Worth Ten Thousand Words, *Cognitive Science*, **11**, pp. 65-99.
- [11] A. Newell and H. Simon. (1972). *Human Problem-Solving*. Englewood Cliffs, N.J., Prentice-Hall.
- [12] G. S. Novak. (1976). Computer Understanding of Physics Problems Stated in Natural Language. *Journal of Computational Linguistics*, Microfiche 53.
- [13] J. M. Paige and H. Simon. (1979) Cognitive Processes in Solving Word Algebra Problems. In H. Simon, *Models of Thought*, Yale University Press.
- [14] C. H. Rapp. (1986). Algebra Reader: An Expert Algebra Word Problem Reader. TR-86-30-6, Computer Science Department, Oregon State University.
- [15] R. Schank. (1975) *Conceptual Information Processing*. North Holland, Amsterdam.