The Thirty-First International Florida
Artificial Intelligence Research Society Conference (FLAIRS-31)

Combining Qualitative and Quantitative Reasoning for Solving Kinematics Word Problems

Savitha Sam Abraham, Sowmya S. Sundaram
Indian Institute of Technology Madras
Chennai 600036
savithas@cse.iitm.ac.in

Abstract

This paper describes a system that combines qualitative and quantitative reasoning to solve kinematics word problems that are expressed in a simplified form of English. Such an integrated approach is useful in identifying the equations required to solve the problem and to infer certain implicit details in the problem scenario. The system also generates self explanatory solutions that can assist a student in mastering the concept involved. We created a dataset of 30 word problems from this domain. Such word problems have not been addressed in recent times.

1. Introduction

Kinematics is the study of motion where the cause of motion is not considered. Word problems from kinematics domain describe scenarios where there are single or multiple objects in motion and the value of one of the parameters of motion has to be computed. A word problem may have many details hidden in the description which may turn out to be significant in solving it.

The system developed integrates qualitative and quantitative reasoning. The need for this integration can be explained through an example problem:

Problem 1: *Thomas kicks a ball at an angle of 43 degrees with a velocity of 40 m/s. There is an obstacle at a height of 23.3843 m and at a distance of 28.2843 m from the initial position of the ball. What is the position of the ball when it hits the ground?*

The final position will depend on whether the ball hits the obstacle or not. In case, it hits the obstacle, the position and the velocity of the ball at that point is also crucial in determining the final position. Also, we should know the effects of such a collision. Though the problem statement does not mention anything about a hit, a person reading it understands that there are multiple behaviours possible. A human solving the problem then uses appropriate equations and numerical facts to eliminate all inconsistent behaviours from the set of possible behaviours. In the same way, our system identifies all the possible behaviours using qualitative reasoning (qualitative simulation) and then identifies the actual behaviour by eliminating inconsistent behaviours through quantitative reasoning. This results in identifying the equations required to solve the problem. Our system generates the answer $-106.6969\ m$ for the above problem with an explanation for it.

We made use of three types of knowledge to solve the above problem: qualitative, quantitative and knowledge about events in the domain. Quantitative knowledge is the knowledge about the different quantities in the domain and the equations connecting them. Qualitative knowledge is the knowledge about how quantities change with time and how the change in one quantity affects the others. For example, the fact that *acceleration is the rate of change of velocity with time* is part of qualitative knowledge. Qualitative knowledge can be used to generate the behaviour of a process in terms of the direction of change of the relevant quantities with time. Knowledge about events include details like the preconditions and effects of an event. This knowledge is encoded in the form of rules and this allows reasoning about events. The occurrence of an event results in discrete changes in the value of parameters. Hence, the system uses qualitative knowledge to reason about continuous change and event knowledge to reason about discrete changes in the physical system.

2. Background and Motivation

(Clark et al. 2007) described a problem solver where only quantitative knowledge about the domain was used to solve a problem. The user who posed the question had to make all implicit details in the question explicit as their system had no notion of trajectory. For example, in *Problem 2*, it should be explicitly stated that the vertical component of velocity is 0 m/s at maximum height. It is important to know this fact to solve the problem.

Problem 2: *Thomas kicks the ball at an angle of 43 degrees to the horizontal. Its initial velocity is 15 m/s. What is the maximum height reached by the ball?*

In order to infer such implicit details, qualitative reasoning can be used. The use of qualitative reasoning in solving physics problems was first emphasised in (De Kleer 1975). Later, (Pisan and Bachmann 1998) described an algorithm that combined qualitative and quantitative reasoning to solve problems from dynamics domain. Given a problem, the system used the initial state description to generate the initial qualitative state, following which attainable envisionment (Forbus 1997) was performed to generate the set of possible successor states. Inconsistent states were pruned using quantitative facts given in the problem. The equations required to solve the problem were identified as part of this pruning process.

This idea is used in our work, but we extend this by considering knowledge about events in the domain. If the system has knowledge about the preconditions and effects of an event, it can identify scenarios where there is a possibility of event occurrence and also know how it is going to change the parameters. We also make use of semi-quantitative reasoning (Kuipers and Berleant 1988) to prune inconsistent behaviours.

3. Proposed System

Input

The input is template based. The templates used are of the form 'The ***quantity name*** of the ***object name*** is ***numerical value*** at ***state description***.'. The parameters or quantities identified for our domain are horizontal position (hp), horizontal velocity (hv), horizontal acceleration (ha), vertical position (vp), vertical velocity (vv), vertical acceleration (va) and angle of projection. A ***state description*** can in turn be given by the value of some parameter or by the occurrence of some event. The example problem, *Problem 2*, gives facts about two states *start* and *maximum height*. The problem is restated as:

Problem 2: *The velocity of the ball is 15 m/s at the start. The angle of projection of the ball is 43 degrees at the start. What is the vertical position of the ball at the maximum height of the ball?*

The input problem is processed and the facts are stored state wise. A state has the following details in it:

Time: *t* (can be an instant or interval)
Object to Parameter-Value Map
  Object: Object-1
  Parameter-Value Map
Event: events occurring at *t* if any
Equation: equations that holds true for this state
Next: pointer to successor of this state

A state of the physical system is described by the state of the individual objects in it. State of the objects is described using its parameters' values. A state represents the properties of the system at a particular time instant or during a time interval. A state also has details about the events that occurred and the list of equations that are true at that state. For example, if the event $hit(O_1, O2)$ is true at a state, then the the equation slot of the state has the equation $position(O_1) = position(O_2)$. A state also has a pointer to the state that follows it once it terminates.

Every parameter has three types of values associated with it - quantitative, qualitative and interval value. The qualitative value of the parameters define the object's qualitative state. A parameter takes one of the three possible values M, Z, P, where 'M', 'P' and 'Z' means that the quantitative value is negative, positive and zero respectively. The interval value is an interval in which the quantitative value of the parameter lies. The interval value of a parameter is initialized depending on its qualitative value and later it gets updated as more quantitative facts are inferred as in (Kuipers and Berleant 1988). As interval is updated, its width becomes narrower and finally when the quantitative value of the parameter becomes known (say, a value *v*), the interval becomes [*v*,*v*].

The set of state descriptions given in the input is stored into a state representation. Since the description of a state given in input is only a partial description, all the slots in the state class may not have fillers. Let the list of input state descriptions $\{I_1, I_2 ... I_f\}$, be called ***I***. In *Problem 2*, there are two state descriptions $\{I_1, I_2\}$, where I_1 is the state with facts related to *start* state and I_2 is the state with facts related to *maximum height*.

Generation of Behaviour

Qualitative behaviour of a system is a sequence of qualitative states through which the system passes. A **qualitative state** of the system is identified using qualitative value of each parameter and the direction in which they are changing[1]. We start by constructing the initial qualitative state. The successors of this state is then generated. If there are multiple successors, the actual successor is identified by pruning inconsistent successors using the numerical facts given. We then check for events at this state using knowledge about pre-conditions of an event. If there is an event occurrence, the next state is generated based on effects of the event. The successor generation continues until all facts given in the problem are identified in the behaviour being generated.

Generation of initial state and successors: The partial description I_1 is used to construct the initial qualitative state of the system named, s_1. s_1 is a complete description of I_1. If qualitative values of some parameters are missing, their default values are used. The direction of change of each parameter is also inferred from the qualitative relations between parameters. The possible successors of a state is computed using the standard QSim algorithm (Qualitative Simulation (Kuipers 1986)) [2].

Pruning inconsistent states: QSim algorithm can return multiple possible successors for a state. There can be only one consistent successor state for a state. The consistent successor is identified with the help of numerical values of the parameters available and the equations of motion. Algorithm 1 describes the pruning process. The quantitative reasoner used for solving equations is described in later sections.

[1]If {va = M, vv = P, vp = P} is current state, then it is inferred that vv is decreasing as va is negative ($d(vv)/dt = va$) and vp is increasing as vv is positive ($d(vp)/dt = vv$). Next state: va is 'M' as $d(va)/dt = 0$, vv reduces to 'Z' from 'P' and vp is P.

[2]Java implementation by Dan Dvorak for QSim from http://www.oursland.net/projects/qsim/ is used in this work.

Algorithm 1: prune(): Algorithm to check consistency

```
Input: behaviour generated so far $B = \{s_0, s_1,..s_i\}$, list of
       possible successors of $s_i$, next[] = $\{s_{i1}, s_{i2}..\}$.
1  for each possible successor $s_j$ in next[] do
2      Create a copy of the behaviour $B$, $Bcopy$.
3      Assume $s_j$ is the successor of $s_i$ in $Bcopy$
4      for each parameter in $s_j$ do
5          Compute value of parameter
6          if value obtained is not agreeing with the
             qualitative value or is an imaginary value then
7              return $s_j$ is inconsistent
8          end
9      end
10     Substitute all the values computed in the set of
         equations available
11     if the system of equations is consistent then
12         Return $s_j$ is consistent
13     end
14     else
15         Return $s_j$ is inconsistent
16     end
17 end
```

Mapping partial state descriptions to states in the behaviour being generated: As each state is generated, it is checked whether the state can be a complete description for a partial state description in input. Such a mapping is allowed if it is consistent to assume the facts in the partial description as facts in the state generated. Once all the partial descriptions in ***I*** are mapped to some state in the behaviour, the behaviour generation terminates.

Checking for events: As each state is generated, it is checked whether there is a possibility of event occurrence in the state. If there is an event occurrence, the next state that the system goes into is not generated by QSim algorithm as the event can interrupt the continuous change in the parameters. The next state in that case will depend on the effects of that event on the parameters of the system. The event in our domain is *meet (object-1, object-2)*. The first level of checking is a qualitative check where the interval value of position is used. If the intervals of horizontal and vertical position for the two objects overlap, there is a possibility for the two objects to meet. This is encoded in the following rule (*S* is the state where event occurrence is checked).

$holdsin(overlap(O1,\ O2),\ S)\ -\ >\ possible(meet(O1,\ O2),\ S)$

If the first level of checking succeeds, the second level of numerical checking is done. This confirms the presence or absence of the event.The numerical check involves checking the consistency of the equations, that are pre-conditions of the event, at state *S*:

$$(hp, O1, S) - (hp, O2, S) = 0$$
$$(vp, O1, S) - (vp, O2, S) = 0$$

The effects of the event is also encoded in the form of rules. These rules are called the effect axioms (Shanahan 1999). Effect of *meet (object-1, object-2)*, in cases like the scenario in *Problem 1*, is that it reverses the direction of horizontal velocity of the object (assuming that the coefficient of restitution is 1). All that is not affected by the event continues to hold in the next state as well.

Solving equations

The previous section explained how qualitative and quantitative reasoning collaborates to generate the behaviour of the system. This section describes the quantitative reasoning module developed by us that is used in computing value of an unknown parameter or to check consistency of a system of equations.

Every equation has details about the list of parameters in the equation, the states being considered as initial and final and the formula connecting these parameters. If the behaviour generated is $B = \{s_0, s_1, s_2...s_i\}$, equations are generated by considering a pair of instantaneous states, as initial and final according to their temporal ordering.

Given a list of equations and a parameter whose value is to be computed (the unknown), the system first constructs a search tree with the unknown parameter at the root. The nodes in the tree are variables and the edges represent equations. A sample search tree is shown in (A) of figure 1. Each node in the tree has details like: the parameter/variable at node, a tabu list that is a list of parameters that cannot be visited again from this node, a pointer to its children nodes and cost of the node. In the figure, the root node expands to give two children through edges with equations $var\_1 + var\_2 + var\_3 = k1$ and $var\_1 + var\_2 = k2$, where $k1$ and $k2$ are constants. This means that var_1 can be computed from the first equation if var_2 and var_3 are known or from the second equation if var_2 alone is known. A node is marked *solved*, if there is an equation that can be solved to get the value of the parameter at that node. The cost of such a node is set as 1 and the node is not expanded further. Another case where a node is not expanded further is when there are no more equations left that does not contain any of the variables in the tabu list of that node. The least cost path is followed in computing the root parameter. If none of the leaf nodes are marked solved, then it indicates that the problem requires solving simultaneous equations.

Solving simultaneous equations is done through a sequence of substitutions. (B) of figure 1 shows a part of a sample search tree that is constructed by our system. The search is done to find an equation with only one variable in it which is the unknown parameter to be computed. The nodes in the tree are equations. The first level of this tree has all the equations that are originally available. The second level of equations are those generated by combining two equations and the third level by combining three equations and so on. Two equations are combined if they have a common variable. Combining two equations means isolating common variable in one equation to get an expression, and then substituting the expression in the other equation in place of the common variable. When an equation is generated whose variable list has only one element which is the unknown parameter, the formula for the equation is computed. For example, if y is the unknown, in the figure, when equation $E10$ with y as the only variable is generated, the

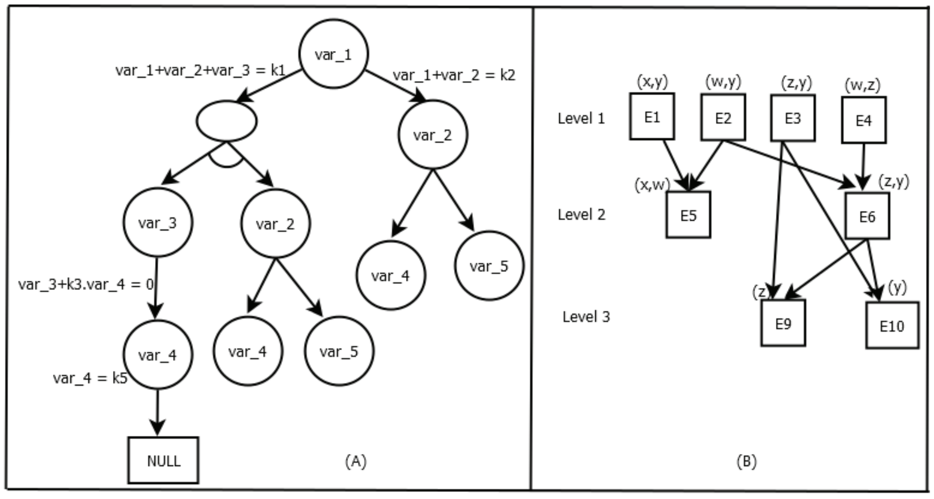

Figure 1: (A) shows a sample search tree with variables at nodes and equations at edges (B) shows a part of the tree that searches for equations.

following processing is done:

Get parents of $E10$: $E3$ and $E6$ with common variable z
Get formula for parent equations. Since $E3$ is an originally available equation, its formula is available. $E6$ is a newly generated equation, hence its formula has to be computed.
Get parents of $E6$: $E2$ and $E4$ with common variable w
Use $E2$ to isolate w and get an equation of the form $w = expression$. Substitute this expression in $E4$ in place of w to get the formula for $E6$
Use formula for $E3$ and $E6$ similarly to get the formula for $E10$
If the formula is valid (it is not of the form $0 = 0$), it is used to solve for y

As $E10$ is processed, we get the sequence of substitutions done to finally arrive at the solution.

Generation of explanation

An explanation is generated from the behaviour. A state in the behaviour is mapped to a description in English. For example, a state with vertical position $0\ m$ and time 0 is mapped to a description "The object is initially on ground". The system goes through the states in the behaviour generated to generate an explanation in English.

Results and Discussion

The proposed system is implemented in Java. We solved a set of around 30 problems from kinematics domain collected from various sources in internet. As compared to (Clark et al. 2007), our solver handles more scenarios (we can handle problems with more than one object) and is able to make inferences from the facts in the problem statement because of the knowledge support it has. Also, the equation solver implemented returns the sequence of substitutions done to arrive at the solution which is easier for students to understand. The problems we could not solve are because of the restriction in input we had. For example, a problem where difference between positions of two objects at some time point is given could not be represented. This is because our input template expects value of a parameter rather than relation between parameters.

Conclusion and Future Scope

A problem solver was developed for kinematics domain that used a combination of qualitative and quantitative reasoning to solve problems expressed in simplified English.

In future, we would like to make the interface more user friendly. We hope to make the input more expressive. Also, we hope to extend this work by building a question answering system around the solver. This would require formal representation of the domain knowledge.

References

Clark, P.; Chaw, S.-Y.; Barker, K.; Chaudhri, V.; Harrison, P.; Fan, J.; John, B.; Porter, B.; Spaulding, A.; Thompson, J.; et al. 2007. Capturing and answering questions posed to a knowledge-based system. In *Proceedings of the 4^{th} international conference on Knowledge capture*, 63–70. ACM.

De Kleer, J. 1975. *Qualitative and quantitative knowledge in classical mechanics*. MIT, Artificial Intelligence Laboratory.

Forbus, K. D. 1997. Qualitative reasoning.

Kuipers, B., and Berleant, D. 1988. Using incomplete quantitative knowledge in qualitative reasoning. In *AAAI*, volume 88, 324–329.

Kuipers, B. 1986. Qualitative simulation. *Artificial intelligence* 29(3):289–338.

Pisan, Y., and Bachmann, A. 1998. Using qualitative reasoning to solve dynamic problems. In *Proceedings of the 12^{th} International Workshop on Qualitative Reasoning*, 167–173.

Shanahan, M. 1999. The event calculus explained. In *Artificial intelligence today*. Springer. 409–430.