Proceedings of the Twenty-Ninth International
Florida Artificial Intelligence Research Society Conference

Hybrid of Qualitative and Quantitative Knowledge Models for Solving Physics Word Problems

Savitha Sam Abraham and Deepak Khemani
Indian Institute of Technology, Madras
Chennai 600036
savithas@cse.iitm.ac.in, khemani@iitm.ac.in

Abstract

This paper describes a system that uses a hybrid of quantitative and qualitative knowledge to solve physics word problems. Such an integration of knowledge from two models is useful to come up with the correct solution for many of these problems. We have applied this hybrid model to solve word problems from projectile motion. These types of word problems have not been addressed in recent times. We have solved a set of 30 problems in this domain.

1. Introduction

Word problems are scenarios expressed in natural language that describe real world situations. Solving a word problem in physics requires the application of the physics concept that explains the situation. To solve such word problems, the students have to often read between the lines and identify the possible inferences that can be made. A machine which is to mimic this process should first understand the natural language description and then solve it using the knowledge that it has. Since the complexity of natural language understanding is high, especially for this domain, the proposed system works on a subset of natural language known as Controlled Natural Language (CNL).

Knowledge related to a physical system can be quantitative or qualitative. Quantitative knowledge is the description of different quantities and the equations relating them. A qualitative model describes the behaviour of a process in terms of the direction of change of the relevant quantities with time. The qualitative variables themselves take one of the three possible values {-, Z, +}. That is, a qualitative variable, for example speed, can be negative, zero or positive. To solve any word problem, we require a quantitative representation of the problem, that is, a set of "quantity=value" assignments. The values of the quantities can be directly substituted in an equation to get the unknown quantity. For some word problems, the quantity-value mapping is directly given. Hence it is easy to construct the quantitative representation of the problem. For other word problems, the quantitative representation cannot be directly obtained from the word problem. The system has to do some reasoning and make inferences from the given facts. This is where qualitative reasoning can help. But the qualitative models generated for a problem scenario are often ambiguous as it considers many different behaviours possible. These ambiguities are resolved using the quantitative information and other facts in the problem to get the right model for the problem. The qualitative model thus obtained is then used to make the implicit information in the word problem explicit.

2. Background and Motivation

(Clark et al. 2007) described a physics problem solver that solved word problems expressed in CNL. The question in CNL stated all the details required to solve the problem and the only job left was to substitute the values of the given quantities in the correct equation. Thus, the knowledge that was used to solve the problem was quantitative knowledge. But this cannot be extended to a certain class of problems. For example, consider the following problem and its CNL:

> **Problem:** *John kicks the ball with an angle of 53 degrees to horizontal. Its initial velocity is 10 m/s. What is the maximum height reached? Where is the ball at time 3 seconds?*
>
> **CNL:** *John kicks a ball. The angle of the ball is 53 degrees. The initial velocity of the ball is 10 m/s. What is the height of the ball after 3 seconds?*

This is the best CNL conversion available. There is no CNL construct for 'What is the maximum height reached?' as there is no notion of trajectory in the knowledge base. Also, some implicit information about the trajectory is missing. For instance, the vertical velocity is 0 m/s at the maximum height. So if it is given in the question that the ball is at maximum height, then it should also be inferred that the ball has vertical velocity 0 m/s at that point. This inference should be made explicit by the user while she reconstructs the question in CNL and it is not necessary that the user would be aware of it. The system described in (Clark et al. 2007) fails to give a solution in the absence of this inferred fact.

The above problem is solved by adding a qualitative reasoning module to the problem solver. Qualitative reasoning generates the quantitative representation of the problem after performing some reasoning. It helps increase the efficiency of CNL used as now the user does not have to make the implicit details in the original question explicit.

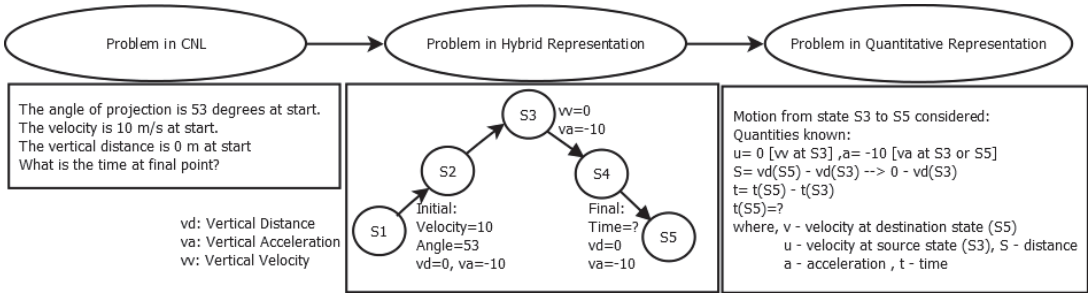

Figure 1: Flow of the entire process

The use of qualitative reasoning in solving physics word problems was first emphasized in (De Kleer 1975). (Pisan 1998) is a work related to ours where qualitative reasoning, plan formulation, algebraic reasoning and diagrammatic reasoning were used together to design a Thermodynamic Problem Solver (TPS). This paper presents a problem solving system that uses a hybrid model where there is a closer integration between the qualitative and quantitative information compared to the work in (Pisan 1998). Envisioning is a process of generating all possible qualitative behaviours of a physical system (Forbus 1997). There are two types of envisioning algorithms - attainable and total envisionment. Attainable envisioner produce all states reachable from a set of initial states, and total envisioners produce a complete envisionment of the system considering all initial states (Forbus 1997). (Pisan and Bachmann 1998) described a third type of envisioning called problem guided envisionment that performs attainable envisionment but stops when a state is found that either satisfies the final conditions given in the problem statement or contradicts with the problem statement. It always stops at the first state that satisfies the final conditions which may not be the actual final state. Hence, here we perform attainable envisionment and stop either when a state is found to be irrelevant to the given problem scenario or when no more transitions are possible from it. Ambiguities in attainable envisionment are then eliminated using domain knowledge and numerical facts in the question to get the actual behaviour/model for the problem. The partial state descriptions in the problem are then mapped to qualitative states in the model to get the hybrid representation of the problem which is then used for further processing.

3. Proposed Approach

The proposed approach for solving physics word problems uses a hybrid model of quantitative and qualitative knowledge to understand the question in CNL and to provide a mechanism for performing commonsense reasoning while solving the problem. The flow of the entire process is shown in Figure 1.

The user first restates the question in CNL. The CNL defined has the following format: "The *quantity* is *number unit* at *state*. What is the *quantity* at *state*?" where, *quantity* is some quantity in the physics concept like velocity, acceleration etc, *number unit* is the value of the quantity with the unit specified and *state* is the state at which quantity has the corresponding value. For example, a question in natural language and its CNL is given below. The states mentioned in the problem below are *start* state and *minimum velocity* state.

Problem 1: *A ball is launched at ground level with an initial velocity of 28 m/s at an angle of 30 degrees to the horizontal. At what time does the minimum speed occur?*

Problem 1 in CNL: *The angle of projection is 30 degrees at start. The velocity is 28 m/s at start. The vertical distance is 0 m at start. What is the time at minimum velocity?*

The hybrid representation of the problem is generated next by first performing attainable envisionment to generate the qualitative model for the given problem scenario, and then, representing the numerical facts given in the question in the model obtained. These two steps are described in detail in sections 3.1 and 3.2 respectively.

3.1. Performing Attainable Envisionment

Attainable envisionment is performed using the standard algorithm but the creation of invalid states during attainable envisionment is avoided using the knowledge in the domain theory and numerical facts in the question. Domain theory has knowledge about how the quantities change (for example, relations like $v = dS/dt$, $a = dv/dt$ where a, v, S and t are acceleration, velocity, displacement and time, are encoded as rules). These rules are used to generate successor states from a state. Domain theory also has knowledge about relations between quantities. One such relation is the Correspondence relation (Bredeweg et al. 2007), that describes the relation between quantities of the form: 'If quantity *Q1* has value *V1*, then quantity *Q2* has value *V2*'. Any state that violates these relations are invalid states. The information in a state of a qualitative model is the set of all quantities and the qualitative value they hold at that state. The initial state is created such that it is consistent with the initial conditions given in the problem[1]. The algorithm for attainable envisionment is given in Algorithm 1.

[1]For problem 1, initially t=Z, hd covered is zero, vd=Z (object is on ground), va=acceleration due to gravity, angle=+, v=+ (from their values 30, 28) and values of remaining quantities are inferred from these given quantities using correspondence relations in domain theory.

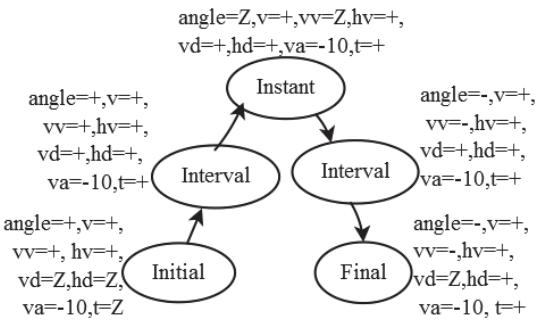

Figure 2: Due to negative va, vv decreases to Z and then negative. vd increases as long as vv=+ and then decreases. t, hd are increasing and hv is found to be constant.

Algorithm 1: Attainable Envisionment

Input: Initial State, Domain Theory
1 State current= initial
2 Next=possible successor states from current state (invalid states and states that are not relevant to the current problem scenario are eliminated)
3 **for** *each n in Next* **do**
4 Generate possible successors of n. If no successors can be generated mark it as final state. Else go to step 2.
5 **end**
6 Return graph representing attainable envisionment

The attainable envisionment for Problem 1 is shown in Figure 2, where, t stands for time, v for velocity, vd for vertical distance, hd for horizontal distance, vv for vertical velocity, hv for horizontal velocity and va for vertical acceleration.

Some quantities may not change continuously. For example, the horizontal component of velocity is constant during the flight of a projectile. But if the object hits a wall on the path, the direction of horizontal velocity gets inverted. Such changes occur as a result of some event and are considered only when there is a possibility of its occurrence, else such changes are irrelevant to the given problem scenario. If there is a wall mentioned in the question, then there is also a possibility for the event *hitting the wall* to occur and hence the system considers the changes of quantities due to this event at the time of producing successor states. This is represented as :

Quantity : Horizontal Velocity
Condition: Exists(Wall)
Change : + to -

There is a possibility of horizontal velocity switching from + to - if the condition Exists(Wall) is satisfied. If there is a wall mentioned in the question, then there are two possible ways in which the quantity horizontal velocity can change. Either it can remain the same (case where it is not hitting the wall) or it can change its direction (case where it is hitting the wall). For each case, a state will be generated.

Once the possible behaviours are generated, the next step is to identify the actual behaviour for the problem. For Problem 1, there is only one possible behaviour (Figure 2) as there are no ambiguities. This need not be the case always. There could be other cases of ambiguity as well, as shown below:

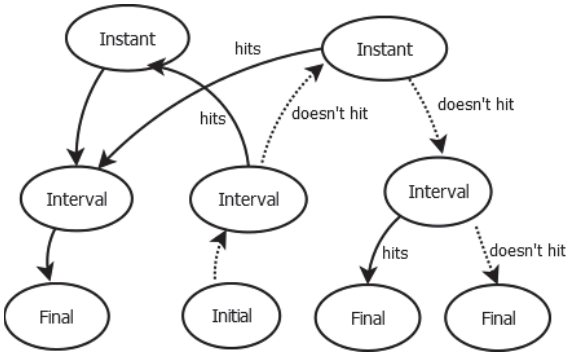

Figure 3: The wall can be anywhere in the path of the ball or could be beyond the range of ball. The ball could hit the wall or not hit the wall at all. The path in dotted lines shows the behaviour when the ball does not hit the wall.

Problem 2 in CNL: *The angle of projection is 45 degrees at start. The velocity is 40 m/s at start. The vertical distance is 0 m at start. There is a wall. The horizontal distance is 80 m at wall. The height of wall is 50 m. What is the horizontal distance at final point?*

The attainable envisionment for Problem 2 is shown in Figure 3. At each state there is a possibility of the next state being the state where the ball hits the wall. The final point on ground depends on where it hits the wall. These ambiguities are resolved after some numerical evaluations using the numerical facts in the question (Section 3.5).

3.2. Inserting Quantitative Information into Qualitative Model

Every numerical value and state mentioned in the problem is identified in the model that is generated after performing attainable envisionment. Consider the problem below:

Problem 3: *The angle of projection is 30 degrees at start. The velocity is 28 m/s at start. The vertical distance is 0 m at start. What is the vertical velocity at time 2 seconds?*

The above problem specifies 2 states: start state and the state where $time = 2$ seconds. While 'start' state corresponds to the initial state in the model generated, identifying the state where '$time = 2$ seconds' requires some more computation. The state with $time = 2$ seconds comes somewhere between the start and final states. Similarly in problem 1, the state 'minimum velocity' also has to be located in the qualitative model. System identifies states like 'minimum/maximum *quantity*' in the model by traversing the states and analysing the way in which the *quantity* changes as it goes from one state to the next. To identify states with a '$quantity = value$' assignment (as in $time = 2$ seconds), the algorithm first calculates the value of the corresponding quantity at every instantaneous state

starting from the initial state. Depending on how the quantity varies between these instantaneous states, the new state is positioned between them. For example, to locate '$time$ = 2 seconds' in the model shown in Figure 2, system first calculates time at Initial state ($t = 0$ is already present), Instant state (after calculation $t = 1.4$) and Final state (after calculation $t = 2.8$). Since value 2 is between 1.4 and 2.8 and, time is increasing from one state to the next, the state with $time = 2$ seconds should come between Instant and Final state. A new instantaneous state is created in between which is an instance of the already present interval state (hence new state inherits the qualitative values for the remaining quantities from this interval state) to represent '$time$ = 2 seconds' point. The value of a quantity at a state is found using *find(quantity, state)* method which is explained in detail in Section 3.4.

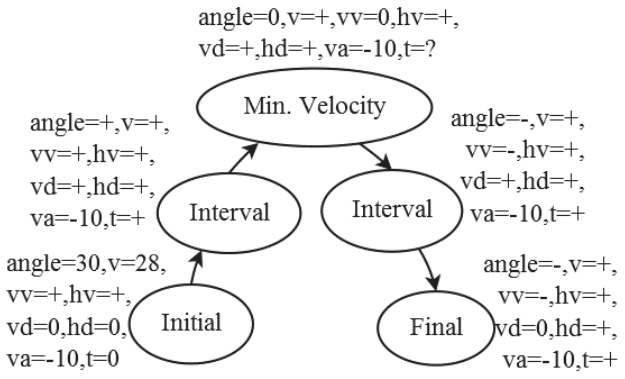

Figure 4: Hybrid representation for Problem 1

For Problem 1, after identifying the minimum velocity point in the model, all the details in the question pertaining to minimum velocity is added to the corresponding state in model. Thus every fact given in the question is identified in the qualitative model. The hybrid representation for Problem 1 is shown in Figure 4.

3.3. Generate Quantitative Representation of the Problem

Given hybrid representation, the system first checks whether it can add more details to a state using the details currently in the state. This is possible if the concept has equations relating quantities pertaining to a single state. For example, the equations for projectile motion are represented in domain theory as:

Physics Concept: Projectile Motion
Equations-Single State: $vv = v * sin(aop)$, $hv = v * cos(aop)$
Applied to: State(S_i) [Any instantaneous state S_i]
Variables:
v: velocity(S_i)
aop: angle of projection(S_i)
vv: vertical velocity(S_i)
hv: horizontal velocity(S_i)

For Problem 1, the system determines the values for vertical velocity and horizontal velocity at the initial state using above equations.

S_i = Initial
v = velocity(Initial) = 28
aop = angle of projection(Initial) = 30
vv = vertical velocity(Initial) = 28*sin(30) = 14
hv = horizontal velocity(Initial) = 28*cos(30) = 24.2

These values are also added to the initial state of the hybrid model of the problem in Figure 4. There are also equations relating quantities from different states. It is important to understand the meaning of an equation before assigning values to its variables. For example, in projectile motion, we have the equation $v = u + at$, where v and u are the values of vertical velocity from two states and t is the difference between the time values at these two states. This equation describes motion in a straight line segment of the trajectory of the projectile. These equations are represented as follows:

Physics Concept: Projectile Motion
Equations: $S = ut + \frac{1}{2}at^2, v = u + at, v^2 = u^2 + 2aS$
Meaning: Motion(S_i, S_j) and Adjacent(S_i, S_j) [captures that the equation is describing motion from instantaneous states S_i to S_j, and that they are adjacent instantaneous states with no other instantaneous states between them (though there can be interval states).This assures the segment of the trajectory considered is a straight line]
Variables:
v : vertical velocity(S_j) [captures that v is final velocity as motion is towards S_j]
u : vertical velocity(S_i) [captures that v is initial velocity as motion is from S_i]
t : time(S_j) - time(S_i) [captures that t is the time taken for the motion from S_i to S_j]
a : vertical acceleration(S_i) or vertical acceleration(S_j) [captures that acceleration is same at S_i and S_j]
S: vertical distance(S_j)-vertical distance(S_i) [distance from S_i to S_j]

In order to use the above equations in projectile motion, the system has to first consider motion between two instantaneous adjacent states S_i, S_j and assign values to the variables v, u, a, t and S accordingly. Out of the two states S_i and S_j, one will be the state in hybrid representation having the unknown quantity and the other will be the instantaneous state adjacent to it. If there is more than one adjacent state available, then the system chooses the one mentioned in the question given. If both the adjacent states are mentioned in the question, then the system chooses one randomly. For example, in Problem 1, the state with unknown quantity is 'Minimum Velocity' and there are two states adjacent to it - Initial and Final. Among Initial and Final states, since Initial state is mentioned in the question, system takes it as the other state. The variables are assigned values to get the quantitative representation for Problem 1:

S_i: Initial
S_j: Min. Velocity
v: vertical velocity(Min. Velocity) = 0 [it is not stated in the

problem but the system inferred it]
u: vertical velocity(Initial) = 14
a: vertical acceleration(Min. Velocity) = -10
t: time(Min Velocity)-time(Initial) $\rightarrow$ Time(Min Velocity)-0
S: vertical distance(Min. Velocity)-vertical distance(Initial)
$Unknown$: t [t at Min. Velocity should be found, hence t is unknown]

3.4. Solve for the Unknown Quantity

Solving for the unknown requires finding the appropriate equation. The hybrid representation should have only one unknown at any time. In case of Problem 1, there is an equation that can directly solve for the unknown which is '$v = u + a * t$' as v, u, a and time at 'Initial' are known to the system and the only unknown is the time at 'Minimum Velocity' state [Substituting, $0 = 14 + -10 * (t(Min.Velocity) - 0)$]. In some cases, in order to find an unknown, the system has to find other quantities. The system finds every other quantity possible with the available information and then tries to solve for the unknown with the newly added numerical values. This is not the ideal way of solving, but since the domain chosen had less number of variables and states, we used this approach. We intend to use a better approach in future that will find only the quantities actually required to solve the unknown. The algorithm is:

Algorithm 2: find(Quantity *Q*, State *S*)

1 Get 'quantity-value' mappings as described in section 3.3, if it has not been generated yet.
2 If any equation can solve for *Q* directly, update the hybrid model by setting value of *Q* at *S* to the value obtained after solving and return the value.
3 Else, try to find value of every other quantity *Q1*, at every instantaneous state *S1*, directly with the available information in hybrid model by calling find_Direct(*Q1*, *S1*) [find_Direct is just Steps 1 and 2 of find()]. This step is repeated until no more new values can be added.
4 If new values are not added after Step 3, it returns that the value of *Q* cannot be found as the system needs more information. Else, try to find the value of *Q* with the updated hybrid model using find_Direct(*Q*, *S*).

3.5. Resolving Ambiguities in Attainable Envisionment

This section describes how quantitative information in the problem can be used to select the actual behaviour for the problem from the many possible behaviors in attainable envisionment. Consider the attainable envisionment in Figure 3. To solve this problem the system has to first choose the actual behavior for the problem from among the multiple behaviours in the attainable envisionment generated. The system requires more knowledge to select the right behaviour. This knowledge is provided as part of the domain theory as procedures. The procedure that helps in resolving '*ball hitting a wall*' ambiguity is encoded as follows:

1. Get the qualitative model for the situation where there is no wall i.e. the behaviour of the ball when there is no wall (dotted line in Figure 3). Let's call this model M. This is encoded as $M = getModel(Wall(False))$. It gets the model for the current problem based on the condition given in its arguments (i.e. $Wall = false$).
2. Locate the horizontal position of wall with respect to the initial state - The wall is at a horizontal distance of 80 m from start point. If the wall is beyond the range of the projectile, it wouldn't hit the wall. Hence the state with $hd = 80$ is identified (let it be S). The above is encoded as $S = findPosition(hd = 80)$. The method calls $find(hd, S)$ for every instantaneous state S to locate the state with $hd = 80$. If ($hd = 80$) is beyond the final state, the ambiguity has been resolved and the behaviour for the problem chosen is M.
3. Find the height reached by the ball at S - This is done only in cases where the wall is within the range of the projectile. If height reached at S is lesser than the wall height (here 50 m) , it will hit the wall, else, it will pass over the wall. This is done by:
   (a) $V = find(vd, S)$
   (b) if($V > WallHeight$) $hit = false$ else $hit = true$ and $hits\_at\_State = S$

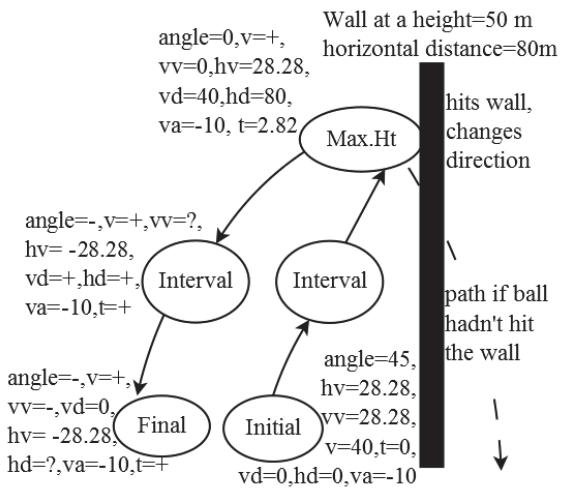

Figure 5: Hybrid representation for Problem 2

Once the system knows whether the object hits the wall or not, it selects the corresponding behaviour from the attainable envisionment. Figure 5 shows that the ball hits the wall at maximum height as the height of wall (50 m) is greater than the height reached by the ball (40 m) at horizontal distance 80 m. There can also be cases where ambiguity cannot be resolved until the problem is solved.

Case 1-More than one behaviour, but only one returns a solution: Consider the following problem:

John kicks a ball at an angle of 53 degrees with a velocity of 10 m/s. What is the maximum height reached?

The height from which the ball is kicked is not mentioned in the question. Hence the algorithm returns two states as possible initial states, one where the initial vertical distance is 'Z' (i.e., ball is thrown from ground-Behaviour 1) and the other where initial vertical distance is '+' (i.e., ball is thrown

from a height-Behaviour 2). If initial vertical distance is unknown, it is not possible to solve the problem. When a student solves this problem, she assumes that the ball is thrown from ground and proceeds. A similar approach is used by the system here. It tries to solve the problem with both the hybrid representations but finds a solution only with one of them (Behaviour 1) and hence chooses that representation.

Case 2-More than one behaviour, and each returns different solutions: For example, consider the problem similar to the problem in Case 1 but with the additional information that total time taken is 20 seconds. This again generates two hybrid representations: one where the initial vertical distance is Zero (H1) and the other where initial vertical distance is '+' (H2). Both the representations can solve the problem, but the maximum height found using H1 is not found to be consistent with the value 20 seconds given in the question. Hence H2 is chosen, i.e. the correct assumption to be made here is that the object is thrown from a height. Consistency is tested by calculating the value of the given quantities one at a time using the value obtained for unknown quantity.

Case 3-More than one behaviour, and each returns same solution: If all the hybrid representations returns the same solution, the solution does not depend on these ambiguities and any one of the representations are chosen randomly.

If the ambiguities cannot be resolved, then the system returns that the problem given is under specified for it to solve. This is a limitation of the system.

4. Discussion

The proposed system is implemented in Java. It is used to solve a set of around 30 problems in projectile motion taken from different sources on the Internet. In general comparing with (Clark et al. 2007), it is seen that the use of hybrid representation allowed solving problems that required some reasoning to infer the implicit details in the question, as illustrated in the examples above. Unlike (Pisan and Bachmann 1998), our system identifies certain partial state specifications in the problem (as in the case of '$time = 2$ seconds' in the problem below) accurately in the qualitative model.

The angle of projection is 53 degrees at start. The velocity is 10 m/s at start. The vertical distance is 0 m at start. What is the velocity at time=2 seconds?

The state with $time = 2$ seconds is identified first in the qualitative model. (Pisan 1998) can identify the initial state correctly but it fails to identify the state with $time = 2$ seconds. It can be mapped to multiple states in the model (any state with $time = +$) and it chooses the first state with $time = +$.

There are also many issues to be resolved. For example, consider the problem below:

A ball is kicked at an angle of 45 degrees to horizontal. The ball should land in the back of a truck which has a trunk of length 2.5 m. If the initial horizontal distance from the back of the truck to the ball is 5 m, what is the max and min velocity so that the ball lands in the truck? Initial height of the ball is equal to the height of the ball at the instant it begins to enter the truck.

In many cases problems describe situations that cannot be represented in CNL. For example, the above problem refers to new entities (like truck trunk length) and there is no quantity in the domain theory to which the value 2.5 m can be assigned. Also, the system needs to know how to solve this problem - the conditions for min and max initial velocity. Such knowledge becomes very problem specific. Also, the system has to generate two representations - one to solve for minimum initial velocity and the other for maximum initial velocity.

5. Conclusion and Future Work

A problem solver was designed that used an intermediate hybrid representation to solve problems that were stated in CNL. This paper illustrated the advantages of having such a representation by solving a set of problems in projectile motion. In future, we would like to extend it to other domains and build a system that will process the original question and give the solution along with an explanation. There are many challenges in developing such a system. Knowledge acquisition is one of them. The domain theory has two types of knowledge: there are rules that help in envisioning and then there are procedures that describe how a problem should be solved (they help in resolving ambiguities in attainable envisionment). Currently, our input is in CNL, but it would be more efficient if the system can convert the original question into CNL or at least design a CNL that is a bit more expressive and can represent questions from other domains as well. When we extend the system to other domains, there should also be a mechanism by which the system can identify the concept to be applied to solve a given problem. We would also like to work on problems that requires application of more than one physical concept to solve a problem.

References

Bredeweg, B.; Bouwer, A.; Jellema, J.; Bertels, D.; Linnebank, F. F.; and Liem, J. 2007. Garp3: A new workbench for qualitative reasoning and modelling. In *Proceedings of the 4th international conference on Knowledge capture*, 183–184. ACM.

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.

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.

Pisan, Y. 1998. An integrated architecture for engineering problem solving. Technical report, DTIC Document.