Checking for Dimensional Correctness in Physics Equations

C.W. Liew
Department of Computer Science
Lafayette College
liew@cs.lafayette.edu

D.E. Smith
Department of Computer Science
Rutgers University
dsmith@cs.rutgers.edu

Abstract

One of the key components of an Intelligent Tutoring System (ITS) is the mechanism for reasoning about the student's input. The impact of this component extends far beyond the presentation of the lesson material to the success of the system itself. It affects how precisely the system can pinpoint student errors and thus the subsequent help that the system provides.

This paper describes an example of a class of physics problems whose answers are most naturally represented as systems of algebraic equations. Analyzing such input requires not only an understanding of algebra but also knowledge of physics concepts.

This paper describes a technique for determining the *dimensional consistency* of algebraic equations in physics using constraint propagation. Unlike other methods, it does not depend on the user defining the dimensions of each variable. Instead, it uses a knowledge base of well known physics variables combined with constraint propagation to determine both the dimensions of values (variables and constants) and also the dimensional consistency of an equation. The technique has been successfully tested on answers obtained from a class of college level introductory physics students.

Introduction

This paper describes ongoing work in addressing the credit-blame assignment problem in the context of a tutoring system for an introductory college level physics course. There are many physics problems at this level whose answers are most naturally represented as systems of algebraic equations. Most physics tutoring systems are unable to reason about systems of algebraic equations because analysis of such input requires not only an understanding of algebra but also knowledge of physics concepts.

One of the main differences between *generic* algebraic equations and algebraic equations describing a relationship in physics is that the latter must be dimensionally consistent. Two examples of such equations are shown below:

A. $T - m_1 * g = m_1 * a$

B. $a_1 = -a_2$

Algebraically speaking, these equations could be added to one another to form a new equation. However in physics, each of the variables, constants, terms, expressions, and even equations must have dimensions. Further they can only be combined using dimensionally consistent operations. Equation A is likely to have the dimensions of Force ($kg\ m\ s^{-2}$) while Equation B would have dimensions of Acceleration ($m\ s^{-2}$). It would thus be incorrect to add these equations since that operation would violate dimensional consistency, i.e., only equations in the same dimensions can be added to or subtracted from one another. Such reasoning requires knowledge of algebraic operations and of the underlying physics concepts.

Before a system can decide if an algebraic equation in physics is correct, it must first see if the equation is *dimensionally consistent*. This is analogous to verifying that the syntax of a program is correct by verifying that the type of each variable is consistent with the operations on that variable. This is a crucial step otherwise any attempt at critiquing the system of equations would result in erroneous and misleading feedback, thus confusing the student even more.

To date, the analysis techniques used by Intelligent Tutoring Systems (ITS) in this domain have required that all variables be defined, i.e., the dimensions are known. This can be a very cumbersome process as a typical set of equations can easily involve ten variables or more.

This paper describes a technique based on constraint propagation for determining the *dimensional consistency* of algebraic equations in physics. Unlike other methods, it does not depend on the user defining the dimensions of each variable. The algorithm uses a knowledge base of well known physics variables and also heuristic inference based on propagation of the values to evaluate the dimensions of an equation. The algorithm is also able to heuristically isolate the error and give the student feedback about the most likely source of a dimension error.

An Example Problem

A simple mechanism that is introduced early in the physics curriculum is that of the pulley. An Atwoods machine is a pulley with two masses, m_1 and m_2 hanging at either end and is shown in Figure 1. A common problem based on the Atwoods machine is to ask the student for the equation(s) that would determine the acceleration of the mass m_1, assuming that m_1 and m_2 are not equal.

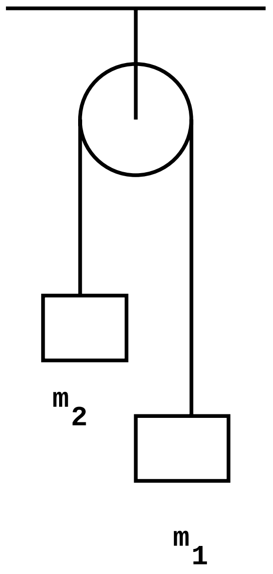

Figure 1: Atwoods Machine

A set of equations that would solve the problem is shown below. Equations 1, 2 and 3 deal with Force while Equation 4 is concerned with Acceleration.

1. $T_1 - m_1 * g = m_1 * a_1$
2. $T_2 - m_2 * g = m_2 * a_2$
3. $T_1 = T_2$
4. $a_1 = -a_2$

The equations represent (1) the net forces acting on the block of mass m_1 (Equation 1 and the block of mass m_2 (Equation 2) through the rope, (2) the connection between the two pieces of rope resulting in the tension in both pieces being the same (Equation 3), and (3) the acceleration of the two blocks in opposite directions with equal magnitude (Equation 4).

An example of an equation that is similar to these equations but is dimensionally inconsistent is:

$T_1 - m_1 = m_1 * a_1$

The equation is inconsistent because in common physics usage, T_1 could represent tension in the rope ($kg\ m\ s^{-2}$) or time (s), while m_1 represents a mass (kg) and a_1 represents an acceleration ($m\ s^{-2}$).

The Issues

An ITS system for physics should be able to generate useful feedback for dimensionally inconsistent equations. The system should do more than just indicate whether an equation is dimensionally consistent or not. It should try and point out where the error occurred. To do this, the system must be able to resolve the following issues:

- determine the dimensions of variables: The system has to reason about the dimensions of the variables T, T_1, T_2 in equations 1-3. In common physics usage, they could represent either tension with the dimensions of ($kg\ m\ s^{-2}$) or time with the dimensions of (s). Defining all variables explicitly before they are used can be a long and tedious process and it is desirable if this step can be minimized.
- determine the dimensional consistency of an equation consisting of multiple terms.
- localize the error(s): Once an error has been detected, i.e., the equation has been determined to be dimensionally inconsistent, the next step is to localize the error. The more accurately the system can focus on the error, the more accurate the user feedback will be. Depending on whether the operators are additive ($+, -$) or multiplicative ($*, /$), the terms may or may not all have the same dimensions.

Related Work

Checking for dimensional consistency is a necessary step before a system can go on to reasoning about the equations. If the dimensions are inconsistent, then the equations will not make sense from a physics viewpoint. Existing systems, e.g. ANDES (Gertner 1998) and PHYSICS-TUTOR (Liew, Shapiro, & Smith 1999), require that the dimensions of each variable and constant be known *a priori* either through a knowledge base of variables and constants or by having the student define them. Once these dimensions are known, it is a fairly simple step to determine if the equation is dimensionally consistent by using some form of "dimension mathematics".

There are many systems that use constraint propagation to ensure consistency of values of variables. Examples of such systems include VEXED (Steinberg 1987), OPIS (Ow & Smith 1986). Their use of constraint propagation is similar except that they are propagating values and not dimensions.

There has also been some work done on adding dimension specifications to programming languages to support compile-time (Novak 1995; Hilfinger 1988) and run-time (Cunis 1992) detection of dimension errors. These systems are more like strongly typed programming languages where every variable has to be defined and has a type. Our system is analogous to a weakly typed language where variables are partially defined on first use and their types are inferred from the context.

Dimension Check Algorithm

This section describes how the algorithm checks for dimensional consistency in equations. The checks are performed in a series of steps as described below:

1. reformulate the equation into a canonical form
2. setup the corresponding constraint tree of dimensions
3. instantiate the dimensions of the variables and thus the dimensions of the leaf nodes in the constraint tree
4. propagate values in the constraint tree and determine the consistency of the overall tree

5. generate feedback to the user

Our algorithm combines the use of a knowledge base of commonly used physics variables and constants with constraint propagation. The knowledge base is used to determine the probable dimensions of each variable. There may be more than one possible combination for a variable. Constraint propagation is used to propagate dimension information to other terms and literals to (1) infer dimension information and (2) determine dimensional consistency. The algorithm can take partial information about the dimensions of a variable and combine that with knowledge of operators and functions (which are just operators) to completely determine dimensions. In essence knowledge, even incomplete knowledge, propagates from one part of the equation to another. This permits the algorithm to reason about dimensional consistency when the variables are not explicitly defined.

Reformulating Equations

In the first step, all equations are reformulated into a canonical form as a sum of products with the right hand side of the equation being set to zero. This involves rewriting the equation to take care of multiplicative operators ($*, /$) as they appear and moving terms to the left hand side. One advantage of this canonical form is that each term must have the same dimensions as every other term and the system can use this requirement to check each term for correctness. Equation 1 would be rewritten as:

$$T_1 - m_1 * g - m_1 * a_1 = 0$$

Setting Up The Tree of Constraints

The system now sets up a binary constraint tree representing the dimensions used within the equation. Each interior node in the graph represents an operator (*e.g.,* $=, +, -, *, /$) or a function (*e.g.,* sin, cos). The leaves of the tree represent each instance of a variable or a constant in the equation. For example, if a variable occurs twice in the equation, there will be two nodes in the tree labeled with that variable. Figure 2 shows the initial tree from Equation 1.

```
      =
    /   \
  0       -
        /   \
      T1      +
            /   \
           *     *
          / \   / \
        m1   g m1  a
```

Figure 2: Constraint Tree

The edges of the constraint tree connect nodes that affect each other directly. The value at each node represents the set of disjoint dimensional values that are consistent with the values of the nodes connected to it. Each member of the set is a three tuple specifying the values in each dimension. The tuple is ordered as $(mass, distance, time)$. For example, a dimensionless variable has a value $< 0, 0, 0 >$ while a variable m for mass will have a value of $< 1, 0, 0 >$.

Initializing Values

Once the constraint tree has been constructed, the system attempts to obtain initial dimension values for the leaf nodes, the variables and constants. The algorithm uses a knowledge base of commonly used variables and “constants” along with their dimensions. It may be that there is either no mapping available or that there is more than one mapping. In these cases, the system will either leave the specific dimension blank or set the node to reflect that more than one dimension is possible. For example, the knowledge base contains the knowledge that T typically means either tension with the dimensions of ($kg\ m\ s^{-2}$) or time with the dimensions of (s). The system will therefore map the variables T, T_1, T_2 from the instructor’s solution to a disjoint set of tension and time. Figure 3 shows the constraint tree for Equation 1 after the initial values have been acquired.

```
             =
          <?,?,?>
        /        \
      0            -
   <?,?,?>    <?,?,?>
            /          \
          T1             +
      <1,1,-2>      <?,?,?>
      <0,0,1>      /       \
                  /         \
                 *           *
            <?,?,?>      <?,?,?>
            /     \      /      \
          m1       g    m1        a1
      <1,0,0> <0,1,-2> <1,0,0> <0,1,-2>
```

Figure 3: Constraint Tree with Initial Values

Note that the node $T1$ has two possible values representing that it could either be tension ($kg\, m\, s^{-2}$) or time (s). In addition note that the nodes with values $<?, ?, ? >$ are dimensionally unconstrained.

Propagating Constraints

Once the initial dimensions for the leaf nodes have been acquired (as much as possible), the next step is to propagate the dimensions to determine if the overall equation is dimensionally correct. The system uses a few simple rules to propagate and infer dimensions. The rules for reasoning about dimensions are listed below:

1. If a node represents an additive operator ($+, -, =$), then the dimension of this node and its children must be the same.

2. If a node represents a trigonometric function (sin, cos), then the node and its child are dimensionless.

3. If a node represents a multiplication operator ($*$), then the dimension of this node is the component-by-component sum of the dimensions of its children.
4. If a node represents a division operator ($/$), then the dimension of this node is the result of subtracting the dimension (component by component) of the right child from the dimension of the left child.
5. Nodes with unknown dimensions acquire them as necessary to maintain dimensional consistency.

Figure 4 shows the tree of Equation 1 after the initial values have been propagated one step.

```
              =
          <?,?,?>
         /       \
       0           -
  <?,?,?>     <1,1,-2>
              <0,0,1>
            /         \
         T1            +
     <1,1,-2>      <?,?,?>
     <0,0,1>      /       \
                 /         \
                *           *
          <1,1,-2>    <1,1,-2>
            /    \      /      \
         m1       g    m1       a1
   <1,0,0> <0,1,-2> <1,0,0> <0,1,-2>
```

Figure 4: Propagating Dimension Values

At each node when new values for dimensions from its neighbors are acquired, the information is used to determine how the values at the current node are affected. This may result in the set contracting or staying the same, it will never expand. For example, Figures 5 and 6 show how the value at the node labeled $(-)$ just below the node labelled $(=)$ changes as information is propagated through two and three steps[1].

If the values at the current node change, then the new values are propagated to all of its nearest neighbors. The algorithm terminates when no contractions can be made to the values of any node in the tree. Figure 7 shows the final tree of dimension values.

Throughout the process, the algorithm checks each node when its set of possible values changes. If the set is reduced to a null set at any time, then the equation is determined to be *dimensionally inconsistent*. In the best case, every node has only one possible value for dimensions and every value is fully determined.

However, it may happen that some nodes may have more than one value that could make the whole equation dimensionally consistent. This can happen when one or more variables are not dimensionally specified or many constants are

[1]The representation $<?,?,?>$ is a shorthand for all possible choices, the universal set of possibilities. The possible values for the node labeled $(-)$ was contracted from $<?,?,?>$ to $(<1,1,-2>, <0,0,1>)$

```
          =
      <1,1,-2>
      <0,0,1>
     /       \
   0           -
<?,?,?>    <1,1,-2>
           <0,0,1>
         /         \
      T1            +
   <1,1,-2>    <1,1,-2>
   <0,0,1>    /       \
             /         \
            *           *
      <1,1,-2>     <1,1,-2>
        /    \      /      \
      m1      g    m1       a1
 <1,0,0> <0,1,-2> <1,0,0> <0,1,-2>
```

Figure 5: Values after Two Propagation Steps

```
          =
      <1,1,-2>
      <0,0,1>
     /       \
   0           -
<1,1,-2>   <1,1,-2>
<0,0,1>     /      \
           /        \
         T1           +
     <1,1,-2>    <1,1,-2>
     <0,0,1>     /       \
                /         \
               *           *
         <1,1,-2>    <1,1,-2>
           /    \     /      \
         m1      g   m1       a1
   <1,0,0> <0,1,-2> <1,0,0> <0,1,-2>
```

Figure 6: Values after Three Propagation Steps

```
          =
      <1,1,-2>
     /        \
   0            -
<1,1,-2>    <1,1,-2>
           /         \
         T1            +
     <1,1,-2>     <1,1,-2>
                 /        \
               *            *
         <1,1,-2      <1,1,-2>
          /    \       /      \
        m1      g     m1       a1
   <1,0,0> <0,1,-2> <1,0,0> <0,1,-2>
```

Figure 7: Final Tree of Dimension Values

used. In such cases, the system will display the list of possible dimensional values for the variables and ask the student to select one.

Generating Feedback

If the equation is determined to be dimensionally correct, then the algorithm terminates. However, if an equation has been found to be dimensionally inconsistent, the next step is to localize the error. The advantage of reformulating the equation as a sum of products is that individual terms can be verified against each other. The algorithm can usually tell the user what the desired dimensions are for the term that is incorrect. However, this does not apply when there are only two terms. In this case, the algorithm cannot identify which term is at fault only that they are not in agreement. For example, if the equation is:

$$T - m_1 = 0$$

then the equation is inconsistent but which term is incorrect cannot be determined. Thus the parent, in this case the whole equation, is marked as the culprit.

In the general case, it is difficult to be accurate when trying to localize errors below the level of a term. There are two general cases:

1. missing a dimension: For example, if the term is $m_1 * v$ $< 1, 1, -1 >$ and the desired dimensions are in Force ($< 1, 1, -2 >$), it could be because (i) the v should be an acceleration or (ii) the term should be divided by a variable with the dimensions of time $< 0, 0, 1 >$.
2. extra dimension: If the term is $m_1 * v^2 < 1, 2, -2 >$ and the desired dimensions are in Force ($< 1, 1, -2 >$), it could be because (i) the term should be divided by a distance or (ii) the use of v^2 is erroneous and should be replaced with an Acceleration.

All dimension errors for each term can be seen as a combination of the above two cases. In most cases, the algorithm cannot accurately localize the error below the level of the term and can only point to the whole term as being in error.

There are a few circumstances when the algorithm can more accurately localize errors below the level of a term. This arises when functions with known dimensions are used. As an example, trigonometric functions return a dimensionless value. If we have a term like ($m_1 * a * sin\theta$) and the desired dimension is Energy $< 1, 2, -2 >$, then the algorithm will point at $m_1 * a$ as the faulty variable. The sin function does not change the dimensions of the term at all and so cannot affect the dimensional consistency.

Experimental Evaluation

In the spring of 2001, we collected roughly 350 answers to four physics problems from 88 different students. They varied in difficulty from the example presented to a problem involving an accelerating pulley. The answers came from an introductory physics course for engineers and science majors at Lafayette College. Analysis showed that dimension errors occurred in roughly fifteen percent of the answers.

Our algorithm was able to correctly analyze all the answers. There were two to three answers for each problem where it was unable to determine the dimensions of some variables and had to prompt the user for further input.

Conclusion

This paper has described a technique for determining the *dimensional consistency* of algebraic equations in physics. Unlike other methods, it does not depend on the user defining the dimensions of each variable. The algorithm uses a knowledge base of well known physics variables and also heuristic inference based on constraint propagation to determine (1) the dimensions of values (variables and constants) and also the dimensional consistency of an equation. The algorithm has been successfully tested on answers obtained from a class of introductory physics students at a small college. Unlike existing systems, variables need not be defined explicitly before they are used.

Acknowledgments

Margie Dunn was instrumental in the implementation of the algorithm. Joel Shapiro was helpful in many discussions of the algorithm. Bradley Antanaitis greatly helped our work by collecting the answers from his physics class at Lafayette College. We are grateful to the students in the Spring 2001 physics 121 course at Lafayette College for participating in our experiment. The comments of several reviewers helped improve the readability of this paper.

References

Cunis, R. 1992. A package for handling units of measure in lisp. *ACM Lisp Pointers* 5(2).

Gertner, A. S. 1998. Providing feedback to equation entries in an intelligent tutoring system for physics. In *Proceedings of the 4th International Conference on Intelligent Tutoring Systems*.

Hilfinger, P. N. 1988. An ada package for dimensional analysis. *ACM Transactions on Programming Languages and Systems* 10(2):189–203.

Liew, C.; Shapiro, J. A.; and Smith, D. 1999. Reasoning about algebraic answers in physics. In *Proceedings of Twelfth International Florida AI Research SocietyConference*, 167–171.

Novak, G. S. 1995. Conversion of units of measurement. *IEEE Transactions on Software Engineering* 21(8):651–661.

Ow, P. S., and Smith, S. F. 1986. Towards an opportunistic scheduling system. In *Proceedings of 19th Hawaii International Conference on System Sciences*.

Steinberg, L. 1987. Design as refinement plus constraint propagation: The VEXED experience. In *Proceedings of AAAI-87*.