Using Qualitative Representations in Controlling Engineering Problem Solving[1]

Yusuf Pisan
y-pisan@nwu.edu
http://www.eecs.nwu.edu/~yusuf/

Northwestern University
Institute for The Learning Sciences
1890 Maple Ave. STE 300
Evanston, IL 60201-3150
(847) 491-7698

Abstract
Engineering problem solving requires both domain knowledge and an understanding of how to apply that knowledge. While much of the recent work in qualitative physics has focused on building reusable domain theories, there has been little attention paid to representing the control knowledge necessary for applying these models. This paper shows how qualitative representations and compositional modeling can be used to create control knowledge for solving engineering problems. This control knowledge includes modeling assumptions, plans and preferences. We describe an implemented system, called TPS (Thermodynamics Problem Solver) that illustrates the utility of these ideas in the domain of engineering thermodynamics. TPS to date has solved over 30 problems, and its solutions are similar to those of experts. We argue that our control vocabulary can be extended to most engineering problem solving domains, and employed in a wide variety of problem solving architectures.

1. Introduction

Engineering problem solving requires both domain knowledge and an understanding of how to apply that knowledge. Recent work in qualitative physics has focused on building reusable domain theories, concentrating on the fundamental physical laws of various domains. However, there has been little research on representing the control knowledge necessary for applying these models. Qualitative representations have been successfully used for designing binary distillation plants (Sgouros, 1993) and for designing controllers (Kuipers & Shults, 1994).

However, since de Kleer's (1975) original work highlighting how qualitative reasoning was needed to solve physics problems, the task of engineering analysis, as exemplified by textbook engineering problems, has been mostly ignored. An exception was a foray into problem solving in thermodynamics (Skorstad & Forbus, 1989), in a system called

[1] Presented at Qualitative Reasoning Workshop, 1996

SCHISM. While SCHISM pioneered some useful ideas (e.g., heuristics for choosing good control volumes), both its domain knowledge and control knowledge were too limited to solve more than a handful of examples. Other research has created formalizations of engineering thermodynamics that are much broader, but still lack the control knowledge needed for capturing the efficiency and flexibility of human experts. For instance, CyclePad (Forbus & Whalley, 1994) contains enough qualitative and quantitative domain knowledge to carry out numerical steady-state analyses of continuous-flow systems. However, it also is excessive in its computations, sometimes solving hundreds of equations that it doesn't need to. A computational account of engineering problem solving must explain the efficiency of expert solutions by showing how to achieve similar efficiencies in software.

We claim that qualitative representations and compositional modeling can be used to create control knowledge for efficiently solving engineering problems. In this paper we describe an implemented system, called TPS (Thermodynamics Problem Solver) that has been developed to explore the use of qualitative representations in controlling engineering problem solving. TPS represents domain knowledge via compositional modeling techniques. TPS' control knowledge includes modeling assumptions, plans and preferences that are used in constraining search. TPS is currently able to solve 30 representative thermodynamics problems from various chapters of an introductory thermodynamics book. The solution plans used by TPS are similar to experts' plans. We expect problem solving control knowledge based on qualitative representations can be adapted for other domains and integrated into other problem solving architectures.

Section 2 describes expert problem. Section 3 analyzes the global structure of the thermodynamics domain. Section 4 describes how plans and preferences are represented in TPS. Section 5 presents the primitives of our modeling language. Section 6 describes TPS' architecture. In Section 7, we present an example of TPS solving a problem. Section 8 discusses related work in developing control knowledge and Section 9 proposes possible extensions to TPS.

2. Expert Problem Solving

How does an expert solve a problem? Looking at expert and novice differences in problem solving is one way of understanding the control knowledge of the expert. Larkin, McDermott, Simon and Simon (1980) have argued that in the domain of physics, novice problem solvers use backward inference while experts use forward inference. More recent findings by Priest and Lindsay (1992) show that experts and novices use similar amounts of forward and backward inference. The contradictory findings suggest that a finer distinction is needed to differentiate between expert and novice behavior.

We conjecture that the expert-novice difference is due to the difference in control knowledge. Control knowledge, which determines whether and when an equation should be used, is expressed to a large degree using qualitative knowledge. TPS' thermodynamics knowledge is represented using qualitative representations and compositional modeling, making fundamental concepts of the domain explicit. TPS uses

explicit modeling assumptions, domain specific plans and preferences to express expert control knowledge. TPS' problem solving involves abstract plans, minimal search of the problem space and a combination of backward and forward inferences to produce expert-like efficient problem solving behavior.

Control knowledge can be encoded by a domain expert as a part of domain knowledge or the control knowledge can be learned through solving problems by identifying paths that lead to failure or success. Chunking (Laird, Rosenbloom and Newell, 1986), EBL (Dejong, 1986) and generating abstract plans (Knoblock, 1994) are strategies for automatically generating control knowledge through solving problems. Since these systems do not re-represent their domain knowledge, fundamental concepts of the domain that are not initially identified cannot be learned. For complex domains, such as thermodynamics, where the problem space is large and not uniform, control knowledge needs to be identified and encoded by a domain expert. The local improvements provided by today's speedup learning algorithms are insufficient to automatically construct such knowledge.

Experts clearly use weak methods (such as backward chaining, forward chaining, and loop detection) in addition to task-specific and domain-specific control knowledge. Although TPS does include weak methods as part of its control knowledge, its expert-like behavior is the result of domain and task specific knowledge. Equation-solving is an intermediate level method, general within the class of domains that involves certain kinds of mathematical models. TPS includes a simple equation solver, of course, but the real art (as de Kleer (1975) and de Kleer & Sussman (1980) pointed out) is to find the right set of equations to work with. This requires knowledge that exploits the global structure of the domain, which can then be encoded as plans and preferences so that a problem solver reasoning locally can achieve expert performance. Consequently, we discuss the global structure of the domain next.

3. Analysis of the Thermodynamics Domain

Thermodynamics is the study of how energy can be transferred from one form to another. Engineering thermodynamics provides the theoretical underpinnings for power plants, engines of all types, refrigerators, and heat pumps. What makes thermodynamics difficult for students is not the complexity of equations, which are no more complex than those found in, say, dynamics. The challenge lies in the fact that the properties being modeled are subtle, meaning that there are both more equations and the range of their applicability is more limited. Engineering thermodynamics requires more explicit reasoning about modeling assumptions in order to derive a consistent set of equations than many domains.

The kinds of performance bounds thermodynamics problems provide are critical in evaluating designs and suggesting improvements to existing systems. The domain knowledge and the control knowledge necessary for engineering problem solving is an important part of thermodynamics experts' professional knowledge. Discussions with domain experts (P. B. Whalley, personal communication) suggests that textbook problems are reasonable approximations for engineering analyses performed in industry.

Most engineering thermodynamics problems require interleaving qualitative and quantitative analysis. Like other engineering domains, analysis of a thermodynamics problem starts with making some modeling assumptions. In many domains these are givens, i.e., ignoring air friction in most elementary dynamics, or ignoring tolerances of components in DC circuit analysis. In thermodynamics, assumptions are resolved by deriving or comparing values for particular parameters, which in turn opens up the possibility of making further modeling assumptions.

We have used compositional modeling (Falkenhainer & Forbus, 1991) for representing the thermodynamics knowledge in TPS. Our thermodynamics domain model is based on the model used in CyclePad (Forbus & Whalley, 1994). It is organized around processes and pure substances which are transformed by processes. A *process* is an instantiation of a physical process, such as heating or cooling, or a combination of basic processes, where energy is transformed form one form to another. A pure *substance* is a homogeneous collection and has associated parameters such as mass, volume, temperature and specific volume that are constant throughout the substance.

Modeling assumptions, equations and background facts tie the different parameters of substances to each other and to the process acting on the substance. Different modeling assumptions can be made about processes and devices. For example, a turbine can be assumed to be isentropic (entropies at inlet and exit being equal) or adiabatic (ignoring any heat loss from the turbine). By making applicable modeling assumptions about a device explicit, TPS makes it possible to reason with modeling assumptions.

3.1 Finding the Phase of a Substance

The phase of substance is the primary modeling assumption about a substance. Fixing the phase of a substance is necessary because it provides the most information for finding the set of applicable equations and tables. A substance can be in one of three states: gas, saturated or liquid (solid states are not normally considered in thermodynamics). The three phases a thermodynamic stuff can be in is represented as an assumption class (Figure 1). When there is a *thermodynamic-stuff* the possible phases of the stuff are instantiated. When a control decision is made to decide the phase of the stuff, the choices for phase are examined further.

```
(defAssumptionClass (decide (phase-of ?stuff))
    :triggers ((thermodynamic-stuff ?stuff))
    :choices ((gas ?stuff) (saturated ?stuff) (liquid ?stuff)))
```

Figure 1: An assumption class, specifying the three phases of a substance

To determine the phase of the stuff, TPS examines the logical relations that can lead to fixing the phase of a substance. Figure 2 shows one of the relations for showing that the substance is in gas form. If TPS decides to use this relation to find the phase, the temperature and the saturated temperature of the substance are calculated and compared.

```
(define-relation gas-T>Tsat
    :triggers ((thermodynamic-stuff ?stuff))
    :relation (:IFF (gas ?stuff) (> (T ?stuff) (Tsat ?stuff))))
```

Figure 2: A logical relationship for showing the phase of the substance

4. Representing Plans and Preferences

Control knowledge for a domain needs to be part of how the domain is modeled. In TPS, we use plans, preferences and common assumptions to guide the problem solving. Plans provide a framework for the problem, preferences are used to decide among choices and common assumptions are used for making assumptions when there is not contradictory information.

Modeling assumptions determine the level of granularity of the model we instantiate for a specific problem. For example, the potential and kinetic energy difference between the inlet and the outlet of the turbine is usually considered insignificant. As a result, when the modeling assumption `(not (consider-ke ?process))` is true, the first law of thermodynamics is simplified to exclude information about kinetic energy. When kinetic energy and potential energy are specified in the problem a finer analysis is possible using the first law in its original form.

Common modeling assumptions are assumptions that are made frequently to simplify problems. Knowing what common assumptions are applicable is an important piece of control knowledge since assumptions are necessary for finding applicable equations and simplifying them. For example, change in kinetic energy is usually ignored for turbines, so when applying the first law to a turbine, TPS makes an attempt to see if a kinetic energy difference can be derived. Determining kinetic energy requires knowing the velocity of the substance, which cannot be derived from any other knowledge in the domain. Lack of knowledge concerning velocities enables TPS to ignore kinetic energy and use a simplified version of the first law to solve the problem. In another problem where velocity is specified the first law can be used in its original form. For a different device, such as a nozzle, changes in kinetic energy cannot be ignored. The common-assumption about turbines is given as a background fact (Figure 3) and is instantiated when TPS is working with a turbine.

```
(defbackground-fact ((turbine ?name ?from ?to))
   (common-assumption (NOT (consider-ke (turbine ?name)))))
```

Figure 3: A background fact

4.1 Representing Control Knowledge for Equations

An equation relates a set of numerical parameters to each other. In TPS, equations contain several parts. The *form* is the traditional mathematical expression of the equation. Equations also have *trigger conditions*, *assumptions*, and *preferences*. Before an equation can be considered its triggers must be known to be true. Trigger conditions are for facts that cannot be derived using the inferences in the domain theory. For

example, the existence of a container can never be derived, so an equation that applies to containers is instantiated only when TPS is already working with a container.

The *assumptions* of an equation are facts that must be true before we can use the equation. An equation that is applicable to gases can not be used for other substances. The reason for separating trigger conditions and assumptions is to guide our inference mechanism. Suppose for example we are asked to calculate the temperature of a piece of fluid, consisting of a known substance. We would have a number of equations we can use to derive temperature, some of which would be applicable if the substance is a gas and others if it is liquid or gas. By separating trigger conditions from assumptions, we can decide that we must find the phase of the fluid before choosing an equation. The structure of the domain and how it is modeled determines which statements should be triggers and which ones should be assumptions.

Preferences reflect how an equation is typically used in problem solving. Although an equation can be solved for any one of its parameters in theory, in practice equations tend to be solved for one variable more than others. The equation given in Figure 4 is the definition for constant-pressure specific heat (cp). This equation is often used for finding specific heat and never used for finding the gamma value of the substance. (Gamma is either known from tables or derived from other equations that contain only a single occurrence of the term.) Control conditions for equations, in terms of preference statements, provide directionality to equations. An equation can still be used to solve for any of its variables, but marking an equation's directionality of typical use explicitly helps guide problem solving.

```
(defequation cp=R/gamma*gamma-1
    :Triggers ((substance ?sub))
    :Assumptions ((ideal-gas ?sub))
    :not-use-for ((gamma ?sub))
    :use-for ((cp ?sub))
    :Form (:= (cp ?sub) (/ (* (R ?sub) (gamma ?sub)) (- (gamma ?sub) 1))))
```

Figure 4: An equation with preferences for direction of solving

It is not possible to create the perfect set of preferences that will prevent the problem solver from ever making the wrong choices. Even experts do not solve every problem with a perfect plan, without backtracking, every time. Preferences provide guidance, but do not eliminate decisions, so TPS can still solve a problem that does not fit its plans through backtracking.

5. Primitives of our Modeling Language

TPS integrates the "physics" of the domain with control knowledge for problem solving. This makes it possible to represent and reason about control knowledge explicitly. Control decisions, suggestions and plans for guiding problem solving are part of the model for the domain, just like equations of the domain. This has required creating a compositional modeling language that includes control knowledge in an integrated fashion. We believe the set of modeling primitives we use are applicable across a wide variety of engineering domains. They are:

defEntity, define-Process: These statements are used to define conceptual entities and physical processes. Each entity and each process has a set of parameters which are instantiated when it exists. Constraints and relations of entities and processes are expressed using define-relation and equations.

defAssumptionClass: An assumption class provides a taxonomy of modeling assumptions. During problem solving, finding the right modeling assumption is necessary before equations and tables can be used.

defBackground-fact: Background facts are instantiated when their triggers are satisfied and instantiate common assumptions to be used in problem solving. Common assumptions can be modeling assumptions that are specific to a process or a numerical value such as molar-mass of water. (See Figure 3 for an example)

define-Relation: Define-relation is used to define logical relations among modeling assumptions and numerical values. Modeling assumptions can be shown by proving that the relation holds between the numerical values (See Figure 2 for an example)

defEquation: There are over 100 equations in TPS. Equations are instantiated and become *available* when the trigger conditions are satisfied. When the assumptions of an equation are satisfied the equation becomes *applicable* and can be used for finding the values of parameters. As described above, each equation can also have control knowledge that guide TPS in using it appropriately.

Control knowledge that is automatically learned usually includes information about what equations should be used in what circumstances. TPS does not have any global knowledge for choosing equations based on the circumstances. TPS uses preferences about directionality of the equation, common assumptions and plans for choosing between equations. The equation with the least unknowns is chosen when multiple equations can be used.

define-Plan: A plan is instantiated when its triggers are satisfied and provides a list of steps to achieve its goals. The triggers of the plan determine when the plan becomes applicable. A simple plan is shown in Figure 6. This plan is applicable when there is a thermodynamic-stuff and a numerical parameter of the stuff is being searched for. The plan tries to achieve the statements in `:goals`, which are intermediary goals for solving the current problem. The steps of the plan are an ordered set of statements that needs to be achieved. If the goals of the plan are found while executing the plan or while solving for another goal, TPS would declare the plan dead and not use it. One of the functions of plans is to interleave forward and backward chaining by directing the problem solver towards resolving modeling assumptions. The plan given in Figure 6 directs the problem solver to fix the phase of the substance, which is needed in order to figure out which equations may be appropriate.

6. The Architecture of TPS

TPS uses the *suggestion architecture* (Forbus & de Kleer, 1993) to solve problems. A problem is provided in the form of some initial assertions and a goal.
A central controller selects what goal to work on next. Working on a goal consists of (1) asking for *suggestions* as to how to achieve that goal and (2) selecting one of the suggestions to act upon. In TPS, the modeling language primitives are automatically compiled into pattern-directed rules that make suggestions and carry them out when the appropriate control assertions are made by the central controller. Carrying out a suggestion can either directly solve the problem, or introduce new problems that must be solved in order to use the proposed method. This process continues until either the original problem is solved or the system runs out of things to try or until resource bounds are exceeded.

A logic-based truth maintenance system (McAllester, 1978; Forbus & de Kleer, 1993) is used to keep track of all data and control dependencies, such as the relationships between goals and the relevance of plans and equations. Thus the status of a goal, whether it is solved, failed or open (Nilsson, 1980), as well as suggestions for achieving the goal are stored in the LTMS. This simplifies control reasoning in several ways. For instance, an open problem about which no suggestions have been made is unsolvable. Such failures are marked as assumptions, which can be reexamined as needed if no other solution is found.

Epstein (1994) argues that multiple heuristic agents can be used to portray human expertise. We have found this model useful in TPS. The agents in TPS are the different plans that are proposed for the current goal. We use a simple heuristic for evaluating between plans: Plans that aim to achieve the current goal are given precedence over plans that suggest subgoals not directly related to the current goal. The shortest plan among relevant plans is chosen when there are multiple plans. When choosing among equations, preferences about whether the equation should be used for that variable is considered. When preferences are not sufficient, an equation with the least number of unknowns is chosen.

TPS' control knowledge is not dependent on the architecture of the problem solver. The problem solver needs to be able to execute plans suggested by plans, subgoal on resolving modeling assumptions when necessary and use equation preferences for choosing between equations. Although the control knowledge can be used with other problem solvers, the representation of the domain model needs to make fundamental concepts of the domain explicit to allow integration and use of control knowledge.

7. An example of TPS solving a problem

A problem description is given in Figure 5 and the plan chosen for this problem is given in Figure 6. Temperatures are given in Kelvins and the pressures are given in Pascals.

```
(add-problem
 :name 'Wylen&Sonntag-Ex5.10
 :givens '((turbine TUR s1 s2)
           (thermodynamic-stuff s1)  (thermodynamic-stuff s2)
           (substance-of s1 water)
           ((mass-flow (turbine TUR)) NVALUE 1.5)
           ((Q (turbine TUR)) NVALUE 8500)
           ((P s1) NVALUE 2000000)  ((T s1) NVALUE 623.15)
           ((velocity s1) NVALUE 50)  ((height s1) NVALUE 6)
           ((P s2) NVALUE 100000)
           ((dryness s2) NVALUE 1)
           ((velocity s2) NVALUE 200)  ((height s2) NVALUE 3)
           ((gravity-on s1) NVALUE 9.8066))
 :goal '(find-numerical-value (work-done (turbine TUR))))
```

Figure 5: TPS' representation of problem 5.10 from Van Wylen & Sonntag (1985)

```
(define-plan analyze-process-with-first-law
    :triggers ((process ?process ?from ?to)
               (thermodynamic-stuff ?from) (thermodynamic-stuff ?to))
    :goals ((or (find-numerical-value (work-done ?process))
                (find-numerical-value (Q ?process))))
    :steps ((show (fixed-phase ?from)) (show (fixed-phase ?to))
            (show (resolve-modeling-assumptions ?process))
            (show (apply-first-law ?process))))
```

Figure 6: Plan chosen by TPS to solve the problem given in Figure 5

The plan given in Figure 6 is one of the most general plans TPS uses. This plan applies to all processes and tries to apply the first law. The first two steps of the plan are fixing the initial and the final state of the substance, then the modeling assumptions about the process are determined and finally the first law is applied.

When there are multiple suggestions and there is no explicit control knowledge to choose among suggestions, suggestions are evaluated and scores are attached to each suggestion. TPS' scoring mechanism is domain independent, giving priority to shorter plans and showing assumptions. Suggestions with low scores are not eliminated, which ensures that even when a problem has to be solved in an obscure manner (i.e., using equations differently than normal), finding a solution is still possible.

Plans are used to interleave backward and forward chaining. An equation, which is a simple plan, requires finding the values of variables and showing necessary assumptions to solve for the current goal. Since each step required by the equation is necessary, equation plans cause the problem solver to perform backward chaining through subgoaling. Other plans, such as the one given in Figure 6 propose subgoals which may not be essential to the current goal. Subgoals that are not essential to the current goal cause the problem solver to forward chain. Although forward chaining could potentially cause exploring part of the problem space that was not needed for the current problem, it is necessary for solving complex problems and reflects experts' pattern of problem solving. In TPS, plans force the assumptions of equations to be resolved before a commitment is made to using an equation. Once the relevant assumptions are resolved, the number of choices is reduced and heuristics are used.

Control Volume: Turbine
Inlet State: Fixed
Exit State: Fixed
Process: SSSF
Model: Steam tables
Analysis: First Law:
$Q_{cv} + m(H_i + V_i^2/2 + gZ_i) = m(H_e + V_e^2/2 + gZ_e) + W_{cv}$
$Q_{cv} = -8.5kW$
Solution:
$h_i = 3137.0$ (from the steam tables)
$V_i^2/2 = (50 \times 50) / (2 \times 1000) = 1.25$ kJ/kg
$gZ_i = (6 \times 9.8066) / 1000 = 0.059$ kJ/kg
Similarly, h_e, $V_i^2/2$ and gZ_e are calculated
Substituting all into first law gives: $W_{cv} = 655.7$ kW

Figure 7: The expert solution to problem given in Figure 5 as given in Van Wylen & Sonntag (1985).

The expert solution to the problem given in Figure 5 is shown in Figure 7. An expert solution uses the minimal number of equations with no backtracking. The solution in Figure 7 uses five equations (kinetic energy and potential energy equations are used twice) and two table-lookups to solve the problem. We compare these results to CyclePad and TPS' results.

CyclePad (Forbus & Whalley, 1994) solves thermodynamics problems using unconstrained forward chaining. TPS' domain knowledge is based on CyclePad's domain knowledge which has been provided by Forbus and Whalley. Although CyclePad can solve problems very fast, it solves a lot more equations then needed. For example, when CyclePad solves a simpler version of the problem given in Figure 5, it instantiates 42 equations and calculates the values for 67 parameters.

```
TPS starts the problem by looking for an appropriate plan
The plan given in Figure 6 is chosen
Executing the plan:
Step 1: Fixing the state at the inlet.
Assume (gas s1) and verify it using saturation temperature.
Step 2: Fixing the state at the exit
Infer that exit is saturated since dryness is given.
Step 3: Resolving modeling assumptions about the Turbine
       Reject no-KE assumption because velocity is given
       Reject no-PE assumption because height is given(
       Reject no-Heat assumption, the value for heat is given
       Not consider Isentropic assumption since it is not adiabatic
Step 4: Applying first law:
       Q = ΔH + ΔKE + ΔPE + W
       Q = 8.5Kw     Given in problem statement
       (spec-h s1) is marked known using tables
Propagating known information:
       (H s1) is marked known using equation
            (H s1) = (mass-flow s1)*(spec-h s1)
       Information about known parameters get propagated. Equation solving
begins after work is marked as known.
Calculating the answer:
       Calculating (change-in-enthalpy (turbine TUR))
       (H S1) = (mass-flow S1)*(spec-h s1) = 1.5 * 3137 = 4705.5
            (2 equations, 1 table lookup)
       (H S2) = (mass-flow S2)*(spec-h s2) = 1.5 * 2675.5 = 4012.5
            (2 equations solved, 1 table lookup)
       (change-in-enthalpy (turbine TUR)) = (H S2) - (H S1)
            (1 equation solved)
       change-in-ke is calculated by finding KE of S1 and S2
            (3 equations solved)
       change-in-pe is calculated by finding PE of S1 and S2
            (3 equations solved)
       W = 655.7kW is found by solving the equation for first law.
            (1 equation solved)
```

Figure 8: TPS' solution to problem given in Figure 5

TPS' solution for the problem involves 3 table lookups, and solving 12 equations. TPS instantiates and solves a many fewer equations than CyclePad. TPS has limited knowledge of symbolic algebra, so instead of combining equations like an expert, it solves equations creating some intermediary values that the expert is able to skip.

TPS is not able to solve all textbook thermodynamics problems for several reasons. TPS' domain model covers steady-state steady flow problems and processes acting on contained substances. Uniform state uniform flow problems, such as a filling a container, are not covered in TPS' domain model. We are currently extending TPS' domain model to cover this class of problems. TPS uses saturated and superheated tables for water and freon, but does not have the necessary tables for other substances yet. The equation solver used in TPS performs limited symbolic algebra, so TPS cannot currently solve problems requiring an equation as an answer. This can be remedied by embedding another program inside TPS for performing symbolic algebra or by building one ourselves. We believe that TPS will be able to solve all of the problems found in an introductory thermodynamics textbook when its domain model and control knowledge have been extended to cover the variety of devices and processes encountered.

8. Related Work

In SCHISM Skorstad and Forbus (1989) use molecular collection ontology (Collins & Forbus, 1987) to determine the control volume for analysis. SCHISM constructs a control volume by following a *piece-of-stuff* around the cycle. TPS uses initial state, the process and the final state of the substance to form control volumes. A separate control volume is created for each device examined. When finding parameters of the whole system, the control volume is expanded to include the relevant parts of the cycle.

In OUZO (Sgouros 1993) heuristic design strategies and knowledge about physical principles is combined for designing separation systems in chemical engineering. OUZO goes through a propose-modify-redesign cycle to find a reasonable design for separation. Unlike problem solving, design problems usually have multiple acceptable solutions and the method for finding the design is not considered important. Textbook problems are more constrained compared to design problems. TPS' control knowledge is used to find the right path to the answer as well as the right solution.

Kuipers (1994) uses qualitative simulation to predict the set of possible behaviors for the system which can then be used by a controller. The behavioral trees include all possible behaviors. When solving textbook problems, we are interested in finding what the current behavior is rather than controlling the behavior. TPS' control knowledge is useful for finding the behavior of the system without considering all the possible alternative behaviors.

9. Discussion

Engineering problem solving requires having domain knowledge and control knowledge for constraining the search space. Previous work on problem solving has concentrated on building problem solvers with general strategies and automatically generated control knowledge from examples. Since our theories of learning and re-representation are not adequate, these programs have had little success. Research in qualitative physics has focused on constructing domain theories, but have not focused sufficiently on how these domain theories can be used for problem solving.

Plans, preferences and logical relations in TPS make it possible to reason about the modeling assumptions explicitly which is necessary for all engineering problem solving. TPS demonstrates how domain specific control knowledge can be created using qualitative representations and compositional modeling principles. TPS has solved over 30 problems and its solutions are similar to those of experts.

We expect that TPS' coverage can be extended to the majority of textbook thermodynamics problems by remedying the limitations pointed out in Section 7. To completely cover such problems two additional extensions are needed. TPS currently cannot read or produce graphs when solving problems. We are planning on integrating TPS with SKETCHY (Pisan, 1995) to enable TPS to solve problems that require

interpreting graphs and diagrams. Another avenue of research is incorporating qualitative simulation as one of TPS' plans. Qualitative simulation can be used to explore possible scenarios for problems where device behavior is ambiguous. We also plan to test our ideas in other engineering domains to determine their generality.

10. References

Chi, M., Fletovich, P. & Glaser, R. (1981) Categorization and representation of physics problems by experts and novices. Cognitive Science, 5, 121-153.

Collins, J. and Forbus, K. (1987). Reasoning about fluids via molecular collections. Proceedings of AAAI-87.

de Kleer, J. (1975). Qualitative and quantitative knowledge in classical mechanics. Technical Report 352. MIT AI Lab, Cambridge, MA.

de Kleer, J. and Sussman, G. J. (1980). Propagation of constraints applied to circuit synthesis. *Circuit Theory and Applications*, 8, 127-144.

Dejong, G. F. (1986). Explanation based Learning. In *Machine Learning: An Artificial Intelligence Approach, Vol. II*. Morgan Kaufman, Los Altos, CA.

Epstein, S. L. (1994). For the right reasons: the FORR architecture for learning a skill domain.. Cognitive Science, 18, 479-511.

Falkenhainer, B. and Forbus, K. (1988). Setting up large-scale qualitative models. In Proceedings of *AAAI-88.*

Forbus, K. and de Kleer, J. (1993). Building Problem Solvers, MIT Press.

Forbus, K and Whalley, P. B. (1994). Using qualitative physics to build articulate software for thermodynamics education. In *Proceedings of AAAI-94*, pp. 1175-1182.

Knoblock, C.A. (1994). Automatically generating abstractions for planning. *Artificial intelligence*, 68, 243-302.

Kuipers, B. J. and Shults, B. (1994). Reasoning in logic about continuous systems. In The Eight International Workshop on Qualitative Reasoning about Physical Systems, Nara, Japan, 164-175.

Laird, J., Rosenbloom, P. and Newell, A. (1986). Universal Subgoaling and Chunking. Kluwer Academic Publishers.

Larkin, J., McDermott, J., Simon, H. and Simon, D. (1980). Models of competence in solving physics problems. Cognitive Science, 4, 317-345.

McAllester, D. (1978). A three-valued truth maintenance system. S.B. thesis, Department of Electrical Engineering, Massachusetts Institute of Technology.

Nilsson, N. J. (1980). *Principles of Artificial Intelligence*. Morgan Kaufman.

Pisan, Y. (1995). A visual routines based model of graph understanding. In Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society, pp. 692-697.

Priest, A. (1986) Inference strategies in physics problem solving. In A.G. Cohn and J.R. Thomas (Eds.), Artificial Intelligence and its Applications. New York: John Wiley & Sons Inc.

Priest, A. & Lindsay R. (1992). New light on novice-expert differences in physics problem solving. British journal of *P*sychology, 83, 389-405.

Sgouros, N. (1993). Representing physical and design knowledge in innovative engineering design. Ph.D. Thesis. Evanston, IL. Northwestern University.

Skorstad, G. and Forbus, K. (1989). Qualitative and quantitative reasoning about thermodynamics. In Proceedings of the Tenth Annual Conference of the Cognitive Science Society.

Van Wylen, G. J. & Sonntag, R. E. (1985) Fundamentals of Classical Thermodynamics. New York: John Wiley & Sons Inc.

Whalley, P. B. (1992). Basic Engineering Thermodynamics. Oxford, NY: Oxford University Press.