Using qualitative physics to build articulate software for thermodynamics education

Kenneth D. Forbus
Qualitative Reasoning Group
The Institute for the Learning Sciences
Northwestern University
1890 Maple Avenue, Evanston, IL, 60201 USA
forbus@ils.nwu.edu

Peter B. Whalley
Department of Engineering Science
Oxford University
Parks Road, Oxford, OX13PJ, UK
whalley@vax.ox.ac.uk

Abstract

One of the original motivations for research in qualitative physics was the development of intelligent tutoring systems and learning environments for physical domains and complex systems. This paper demonstrates how a synergistic combination of qualitative physics and other AI techniques can be used to create an intelligent learning environment for students learning to analyze and design *thermodynamic cycles*. Pedagogically this problem is important because thermodynamic cycles express the key properties of systems which interconvert work and heat, such as power plants, propulsion systems, refrigerators, and heat pumps, and the study of thermodynamic cycles occupies a major portion of an engineering student's training in thermodynamics. This paper describes CyclePad, a fully implemented learning environment which captures a substantial fraction of a thermodynamics textbook's knowledge and is designed to scaffold students who are learning the principles of such cycles. We analyze the combination of ideas that made CyclePad possible, comment on some lessons learned about the utility of various techniques, and describe our plans for classroom experimentation.

1. Introduction

One of the central motivations for research into qualitative physics has been its potential for the construction of intelligent tutoring systems and learning environments. By providing computational accounts of human reasoning about the physical world, ranging from what the person on the street knows to the extensive expertise of scientists and engineers, qualitative physics should provide representation languages and reasoning techniques that can be applied to helping people make the transition from novice to expert reasoning about physical systems. Indeed, some of the earliest work in the field was directly aimed at instructional problems (e.g., [1 ,2]). Over the last decade there have been several important efforts aimed at using qualitative physics to help teach diagnosis, troubleshooting, and operation of complex physical systems (e.g., [3 ,4 ,5 ,6]), but little effort has been focused on using qualitative physics in classroom settings, to help undergraduates learn principles of a domain (a rare exception is [7]).

In this paper we describe a system, called CyclePad, that has been built to help engineering undergraduates appreciate and therefore learn important principles of thermodynamics. CyclePad provides a conceptual CAD environment where students can design and analyze power plants, refrigerators, and other thermodynamic cycles. It relies on a synergistic combination of existing AI techniques: compositional modeling to represent and reason with modeling assumptions, qualitative representations to express the intuitive knowledge of physics needed to detect impossible designs, truth-maintenance to provide the basis for explanations, and constraint reasoning and propagation to provide efficient mathematical reasoning. It incorporates a substantial fraction of the knowledge in a typical engineering thermodynamics textbook [8], and has been tested on over two dozen examples of problems involving steady-state, steady flow systems where numerical answers or single-parameter sensitivity analyses are required.

Section 2 describes the pedagogical problems that motivated the design of CyclePad, including a brief overview of what thermodynamic cycles are and how they work. Section 3 demonstrates CyclePad's operation from a user's perspective. How CyclePad works is the subject of Section 4, with Section 5 summarizing the lessons we have learned so far in building the system. Section 6 outlines our plans for future work.

2. The task: Teaching the design of thermodynamic cycles

A thermodynamic cycle is a system within which a working fluid (or fluids) undergoes a series of transformations in order to process energy. Every power plant and every engine is a thermodynamic cycle. Refrigerators and heat pumps are also examples of thermodynamic cycles. Thermodynamic cycles play much the same role for engineering thermodynamics as electronic circuits do for electrical engineering: A small library of parts (in this case, compressors, turbines, pumps, heat exchangers, and so forth) are combined into networks, thus potentially generating an unlimited set of designs for any given problem. (Practically, cycles range

Proceedings of AAAI-94, pp 1175-1182, August, 1994.

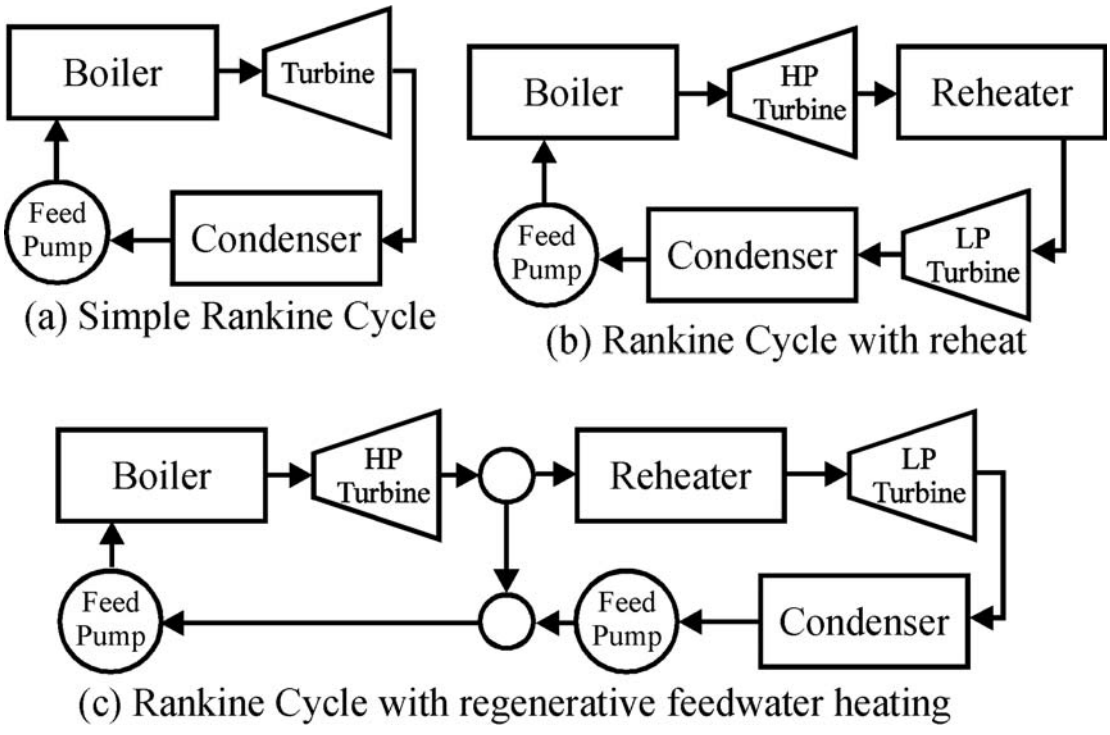

Figure 1: Sequence of conceptual designs for a power plant

from four components, in the simplest cases, to networks consisting of dozens of components.) One source of the complexity of cycle analysis stems from the complex nature of liquids and gases: Subtle interactions between their properties must be harnessed in order to improve designs. Cycle analysis answers questions such as the overall efficiency of a system, how much heat or work is consumed or produced, and what operating parameters (e.g., temperatures and pressures) are required of its components. An important activity in designing cycles (or indeed in many engineering design problems) is performing sensitivity analyses, to understand how choices for properties of the components and operating points of a cycle affect its global properties.

To illustrate, consider the sequence of power plant designs in Figure 1. Figure 1(a) shows a simple Rankine cycle, which pumps a working fluid (as liquid water) into a boiler to produce steam. In the turbine the high-pressure steam expands, thus performing work. Heat is extracted from the steam in the condenser so that the working fluid is again water. Finally, this water is pumped into the boiler (which requires the pump to absorb work) thus beginning the whole cycle again. These processes happen continuously in steady flow, in every part of the system. As it happens, this cycle is not very efficient. Raising the temperature of the boiler increases efficiency, but due to material properties of the components, there are upper bounds on operating pressures and temperatures. Figure 1(b) shows a more efficient design, which uses a second turbine to extract more energy from the steam. The purpose of the reheater is to ensure that the steam does not become "wet", i.e., to begin to condense, because water droplets moving at high speed may damage a turbine. The extra energy required to reheat the steam is more than balanced by the additional work gained from the second turbine. One can do even better, however. Figure 1(c) shows a *regenerative feedwater cycle* where some of the steam from the outlet of the high-pressure turbine is routed back to the water feeding the boiler. The boiler then is adding heat to water that is starting at a higher temperature, which increases efficiency.

The analysis and design of thermodynamic cycles is the major task which drives engineering thermodynamics, aside from applications to chemistry. In thermodynamics education for engineers, cycle analysis and design generally appears towards the end of their first semester, or is even delayed to a second course, since understanding cycles requires a broad and deep understanding of the fundamentals of thermodynamics. Even the most introductory engineering thermodynamics textbooks tend to devote several chapters to cycle analysis, and in more advanced books the fraction devoted to cycles rises sharply. Indeed, some textbooks focus exclusively on cycle analysis (e.g., [9]). Aside from their intrinsic interest, the conceptual design of

Proceedings of AAAI-94, pp 1175-1182, August, 1994.

thermodynamic cycles provides a highly motivating context for students to learn fundamental principles more deeply than they would otherwise.

```
Q: Why is Nu_thermal(CYCLE) = 46%?
A: Nu_thermal(CYCLE) = 46%
   was found by using
   Nu_thermal(CYCLE) =
    net_work(CYCLE)/Q_in(CYCLE)
   on
    Q_in(CYCLE) = 4069.555 kW
    net_work(CYCLE) = 1872.219 kW

Q: Why is net_work(CYCLE) = 1872.219 kW?
A: net_work(CYCLE) = 1872.219 kW
   was found by using
   net_work(CYCLE) =
    work_in(CYCLE)+work_out(CYCLE)
   on
    work_in(CYCLE) = -15.68446 kW
    work_out(CYCLE) = 1887.903 kW

Q: Why is work_out(CYCLE) = 1887.903 kW?
A: work_out(CYCLE) = 1887.903 kW
   was found by using
   work_out(CYCLE) =
    work(TURBINE2)+work(TURBINE1)
   on
    work(TURBINE2) = 1372.131 kW
    work(TURBINE1) = 515.7723 kW

Q: Why does work_out(CYCLE) =
    work(TURBINE2)+work(TURBINE1) hold?
A: work_out(CYCLE) =
    work(TURBINE2)+work(TURBINE1)
   because we assumed:
    CYCLE is a thermodynamic cycle
    the work-flows-out of CYCLE consist
     of {work(TURBINE2),work(TURBINE1)}

Q: How has TURBINE1 works isentropicaly
     been used?
A: TURBINE1 works isentropically
   was used to derive
    TURBINE1 works adiabaticaly
    T(S3) = Tout_i(TURBINE1)
    s(S2) = s(S3)
```

Figure 2: A CyclePad hypertext dialog

A variety of problems arise when teaching students how to design and analyze thermodynamic cycles:[1] (1) Students tend to get bogged down in the mechanics of solving equations and carrying out routine calculations. This leads them to avoid exploring multiple design alternatives and to avoid carrying out trade-off studies (e.g., seeing how efficiency varies as a function of turbine efficiency versus how it varies as a function of boiler outlet temperature). Yet without making such comparative studies, many opportunities for learning are lost. (2) Students often have trouble thinking about what modeling assumptions they need to make, such as assuming that a heater operates isobarically, leading them to get stuck when analyzing a design. (3) Students typically don't challenge their choices of parameters to see if their design is physically possible (e.g., that their design does not require a pump that produces rather than consumes work).

CyclePad was designed specifically to help students learn engineering thermodynamics by providing an intelligent learning environment that handles routine calculations, facilitates sensitivity analyses, helps students keep track of modeling assumptions, and detects physically impossible designs.

3. Overview of CyclePad

CyclePad can be viewed as a CAD system for the conceptual design of thermodynamic cycles, although it provides substantially more explanation capabilities than existing CAD software. CyclePad performs steady state analyses of steady-flow thermodynamic cycles. The restriction to steady-state is standard for this kind of analysis, since issues of how to start up and shut down the plant, or how easy it will be to monitor, maintain, or troubleshoot are issues of concern only after the basic design has been shown to be sound with respect to the goals for it (e.g., amount of work produced, efficiency, etc.). The restriction to steady-flow systems means that CyclePad cannot currently be used to analyze internal combustion engines, such as Otto or Diesel cycles. Although we plan to extend CyclePad to analyze such systems, steady flow cycles constitute the majority of the cycle-related material taught to engineering students. (For example, in [8] four out of five chapters on cycles concern steady flow cycles, in [9] it is 9 out of 10 chapters, and [10] focuses only on steady-flow systems.)

When a user starts up CyclePad, they find a menu of component types (e.g., turbine, compressor, pump, heater, cooler, heat exchanger, throttle, splitter, mixer) that can be used in their design. Components are connected together by *stuffs*, which represent the properties of the working fluid at that point in the system. (Stuffs serve the same role as nodes in electronic circuits.) The interface helps the user put together a design by highlighting what parts remain unconnected and providing simple critiques of the structure. Once the structural description of the cycle is finished (e.g., there are no dangling connections or stuffs), CyclePad allows the user to enter an analysis mode, where the particular properties of the system, such as the choice of working

[1] These observations are based on the experience of the second author, who teaches engineering thermodynamics to undergraduates.

fluid, the values of specific numerical parameters, and modeling assumptions can be entered.

CyclePad accepts information incrementally, deriving from each user assumption as many consequences as it can. At any point questions can be asked, by clicking on a displayed item to obtain the set of questions (or commands) that make sense for it. In addition to numerical parameters and structural information, all modeling assumptions made about a component are displayed with it, and clicking on a component shows the modeling assumptions that can legitimately be made about that component, given what is known about the system so far. The questions and answers are displayed in English, and include links back into the explanation system, thus providing an incrementally generated hypertext. Figure 2 illustrates.

In addition to numerical assumptions, selecting a component provides commands for making or retracting modeling assumptions concerning that component. For example, clicking on a new turbine yields a menu of commands which offers the options of assuming the turbine is adiabatic or isentropic. Such modeling assumptions can introduce new constraints which may help carry an analysis further and new parameters (e.g., the efficiency of the turbine) that must be set.

When CyclePad uncovers a contradiction, it changes the interface to provide tools to resolve the problem by presenting the source of the contradiction (e.g., an impossible fact becoming believed, or conflicting values for a numerical parameter) and the set of assumptions underlying that contradiction. The hypertext dialog facilities can be used with this display to figure out which assumption(s) are dubious and change them accordingly.

We have tested CyclePad on over two dozen examples to date, ranging from simple ideal gas problems to the analysis of a combined gas turbine/steam Rankine cycle system. We believe that the current version of CyclePad can solve all of the problems in [8] concerning steady-state analyses of steady-flow cycles that require numerical answers or sensitivity analyses involving a single parameter. (We are continuing to test it on new examples, drawn from other textbooks as well.) CyclePad is very efficient. The combined gas turbine/steam Rankine cycle is the most complex system in [8], consisting of ten components. Good students take between 20 minutes and one hour to solve this problem. CyclePad does somewhat more work in analyzing this problem than a good student would, instantiating 219 equations involving 362 parameters, whereas a solution can be found using only 52 equations. However, CyclePad is still faster, taking just over two minutes on a workstation, versus just over ten minutes on a PowerBook 165c. We believe that the combination of the speed at which CyclePad carries out the routine calculations, its explanation facilities, and its consistency-checking facilities, will make it a valuable tool for students learning thermodynamics.

4. How CyclePad works

The overall structure of CyclePad was inspired in part by EL [11], an experimental system for DC and AC analysis of analog electronic circuits. EL was one of the first systems to use constraint propagation and dependency networks to organize its reasoning, and introduced the idea of dependency-directed backtracking. In this section we see how CyclePad exploits the advances made by the field since EL, by examining each of the AI ideas that contributes to CyclePad's operation, and the reasons for these particular design choices.

4.1 The role of compositional modeling

Compositional modeling [12 , 13 , 14] provides formal representation and reasoning techniques for formulating and reasoning about models. Knowledge about a domain is organized as collections of *model fragments*, organized by modeling assumptions and the ontology of the domain. Formulating a model for a specific problem consists of instantiating fragments from the domain theory, taking into account the kinds of tasks the model is to be used for.

As noted previously, steady-state analyses are required for the conceptual design of thermodynamic cycles. By restricting ourselves to steady-flow systems, it is also the case that the process structure (i.e., the collection of physical processes acting in each component) is fixed for all time. These restrictions allow us to organize the domain theory around the components which comprise a cycle and the properties of the working fluid at particular locations (i.e., the connections between components).

The modeling language used in CyclePad is similar to other implementations of compositional modeling. For example, Figure 3 shows part of CyclePad's model of a heater. CyclePad's knowledge base consists of 29 conceptual entities, 5 physical processes, 9 assumption classes, 98 equations, 40 pattern-directed rules and 41 background facts about thermodynamics.

Modeling assumptions are organized into *assumption classes* [15 , 13]. Assumption classes are always associated with particular classes of components. The relevance of one assumption class can depend on the particular choices made for another assumption class. For example, it only makes sense to consider whether a compressor is isentropic if it is already known (or assumed) to be adiabatic.

```
  (defEntity (Abstract-hx ?self ?in
                                  ?out)
    (thermodynamic-stuff ?in)
    (thermodynamic-stuff ?out)
    (total-fluid-flow ?in ?out)
    (== (mass-flow ?in)
        (mass-flow ?out))
    (parameter (mass-flow ?self))
    (parameter (Q ?self))
    (parameter (spec-Q ?self))
    (heat-source (heat-source ?self))
    ((parts :cycle) has-member ?self)
   (?self part-names (in out))
   (?self IN ?in)(?in IN-OF ?self)
    ?self out ?out)(?out out-of
?self))

  (defAssumptionClass
    ((abstract-Hx ?hx ?in ?out))
      (isobaric ?hx)
      (:not (isobaric ?hx)))

  (defEntity (Heater ?self ?in ?out)
    (abstract-Hx ?self ?in ?out)
    (?self instance-of heater)
    (heat-flow (heat-source ?self)
               (heat-source ?self)
            ?in ?out)
    ((heat-flows-in :cycle)
       has-member (Q ?self))
    (> (Q ?self) 0.0))

  (defEquation Hx-law
    ((Abstract-Hx ?hx ?in ?out))
    (:= (spec-h ?out)
             (+  (spec-h  ?in)  (spec-Q
?hx))))

  (defEquation spec-Q-definition
    ((Abstract-Hx ?hx ?in ?out))
    (:= (spec-Q ?hx)
        (/ (Q ?hx) (mass-flow ?hx))))
```

Figure 3: A sample of CyclePad's knowledge base

4.2 The role of constraint reasoning and propagation

A design is not finished until numerical values have been chosen for its parameters. This is one reason why the overwhelming majority of thermodynamics textbook problems require numerical answers.[2] This fact, plus the relative simplicity of the equations involved, has meant that constraint propagation has sufficed for CyclePad.

In compiling CyclePad's knowledge base, equations are automatically converted into antecedent constraint rules that propose values for the *n*th variable in an equation whenever the other *n-1* variables are known. Redundant equations are introduced when needed to overcome simultaneities. This automatic translation simplifies development. Equations in their original form are still represented in the knowledge base, however, and are used in two ways. First, they are part of the dependency structure for any results calculated via constraint propagation, for explanatory accuracy. Second, they can be inspected via the query system, so that students can find out what equations mention a specific parameter, and what equations might be used to calculate a desired value.

Property tables comprise a critical source of information for CyclePad. Property tables are woven into the constraint propagator via pattern-directed rules, operating under the same protocol as the rules compiled for equations. Due to the inherent loss of accuracy in interpolation, it is important, unlike equations, to *avoid* using tables in every logically possible fashion. Given a superheated vapour, for instance, knowing the pressure and temperature suffice to determine everything else (e.g., the specific enthalpy, specific entropy, etc.). If one redundantly computes from, say the specific enthalpy and specific entropy what the pressure and temperature will be, it is very likely that the newly estimated values will trigger a contradiction, given the accumulated inaccuracies in the interpolation process. Consequently, an important design choice in implementing tables is selecting which directions of access are likely to prove most productive for the kinds of analyses being made.

4.3 The role of qualitative physics

In CyclePad qualitative physics provides the medium for representing constraints on what is physically possible. Occurrences of physical processes inside components are explicitly represented. Each process occurrence includes ordinal constraints that are tested against numerical values by CyclePad's constraint propagation mechanism. Figure 4 shows a sample of what CyclePad knows about physical processes.

[2] In a typical textbook we surveyed, 90% of the exercises required numerical answers.

```
(defProcessEpisode (fluid-flow ?in ?out)
  (same-substance ?in ?out))

(defProcessEpisode (total-fluid-flow
                     ?in ?out)
  (fluid-flow ?in ?out)
  (== (mass-flow ?in) (mass-flow ?out)))

(defProcessEpisode
 (heat-flow ?src-start ?src-end
            ?dst-start ?dst-end)
  (> (T ?src-start) (T ?dst-start))
  (:not (< (T ?src-start) (T ?dst-end)))
  (:not (> (T ?dst-end) (T ?src-end))))

(defProcessEpisode (compression
                     ?in ?out ?worker)
  (> (P ?out) (P ?in))
  (< (spec-shaft-work ?worker) 0))

(defProcessEpisode (expansion
                    ?in ?out ?receiver)
  (< (P ?out) (P ?in))
  (> (spec-shaft-work ?receiver) 0))
```

Figure 4: Physical process knowledge in CyclePad

4.4 The role of truth maintenance

We used an LTMS [16] in CyclePad because it offered the best tradeoff between inferential power and economy. (In fact, CyclePad's inference engine is the LTRE system from [17] We ruled out a JTMS because Horn clauses are too clumsy for many of CyclePad's inferential needs, including biconditionals (used in definitional consequences of modeling assumptions, e.g., a compressor is operating isentropically exactly when its isentropic efficiency is 1.0) and TAXONOMY constraints [18] (used in implementing assumption classes). The ability of an ATMS to provide rapid switching between very different contexts was not required: While frequent additions and retractions of assumptions are made in carrying out an analysis, typically these changes are a small fraction of the working set of assumptions in force.

A critical role for the LTMS dependency network is as an input for explanation generation. Explanations in CyclePad are in terms of *structured explanations*, an abstract layer between the reasoning system and the interface that casts the consequences of the inference system in terms relevant to the user. This includes summarization (e.g., [19]), as in removing any reference to implementation-dependent information such as the constraint propagation mechanism from an argument. It also includes making explicit implicit dependencies, such as the variables whose values must be known before the constraint rule implementing a particular equation will fire when explaining what assumptions might lead to more progress.

5. Lessons learned from developing CyclePad

CyclePad represents one of the first attempts to apply ideas developed by the qualitative physics community to a real application. While CyclePad has not yet been fielded, we believe that we have already learned several generally useful lessons in building it.

5.1 Compositional modeling scales up

Previous uses of compositional modeling have either focused on large but purely qualitative domain theories, or small quantitative theories. CyclePad demonstrates that the ideas of compositional modeling can be used to organize a substantial body of quantitative and qualitative knowledge so that it can be used effectively.

Automatic model formulation, which typically has been the focus of previous compositional modeling work, is less relevant for this application. Nevertheless, the mechanisms of assumption classes and logical constraints between modeling assumptions provide a valuable service in helping the user organize an analysis. In fact, one of the skills being taught in using CyclePad is model formulation. A boiler, for instance, is typically approximated as a heater for the purposes of cycle analysis. A flash chamber is modeled as a splitter whose working fluid is saturated and with particular assumptions about the dryness of the outlets. A multi-stage turbine is modeled as a sequence of turbines and splitters. CyclePad helps users analyze models, so they can figure out if their choice of idealization makes sense, but currently CyclePad does not provide direct assistance with formulating an idealized model from an informal specification.

5.2 Regarding constraint reasoning

In this task numerical constraint propagation suffices. There are however natural extensions of CyclePad's analytic abilities for which algebraic manipulation would be useful. For instance, some insights about how a cycle works are best captured via equations.[3] We plan to extend CyclePad to derive such equations on demand. Our experience with the constraint rules compiler in CyclePad, and other work on thermodynamics problem solving [20], suggests that relatively simple algebraic capabilities will suffice for this extension.

We draw two additional conclusions regarding constraint manipulation. First, the commercial world has developed many powerful symbolic algebra packages, such as Mathematica, Maple, and Macsyma, which in some cases are excellent off-the-shelf solutions to

[3] For example, figuring out that for a gas turbine cycle the maximum specific work output is achieved when the pressure ratio is the square root of its maximum possible value [8].

particular problems. However, we suspect that many educational applications will be like CyclePad: Simple algebraic facilities are all that is required, and thus the complexity (and expense) of integrating commercial symbolic algebra packages can be avoided. Second, we found that special-purpose constraint languages (e.g., [21]) were too restrictive for our purposes. Given the need to reason about modeling assumptions and the need to integrate information from property tables, it was much easier to implement a simple constraint propagator inside a pattern-directed inference system than it was to interface a special-purpose constraint manipulator. Aside from applications where scaling up to extremely large system descriptions (e.g., VLSI CAD) is a key requirement, it is hard to see any situation where using such languages makes sense.

5.3 Regarding qualitative physics

The combination of steady-state analysis, the restriction to steady-flow systems, and the use of idealized components dramatically simplified the representation of physical processes, since the occurrence of particular physical processes could simply be stipulated inside a component.

CyclePad's focus on quantitative analysis also means that the major inferential role for qualitative physics is ruling out physically impossible designs. We believe similar simplifications will hold in many other applications, since well-designed artifacts explicitly represent the important physical changes in terms of the kinds of components and connections that comprise a schematic, and many science and engineering educational applications involve quantitative knowledge heavily.

On the other hand, certain extensions to CyclePad's capabilities will require substantially more qualitative representations and reasoning. For instance, CyclePad currently does not try to explain how components work, nor does it provide assistance for understanding the physical rationale underlying design changes. To formalize such arguments will take richer qualitative representations, as well as the ability to reason with property diagrams (e.g. [22]). Fortunately, the automatic instantiation of physical process descriptions from a domain theory is an inexpensive and well-understood operation.

5.4 Regarding explanation generation and TMSs

The use of a structured explanation system as an abstraction layer between interface and reasoning system was extremely helpful in developing CyclePad, since it allowed us to optimize each independently. We also found, as suggested by [23], that sophisticated natural language generation techniques were inappropriate for this task. The ability to automatically generate hypertext in response to a user's questions obviates the need for discourse planning, and the fixed nature of the task means that issues such as selecting the appropriate level of detail in an explanation can be postponed. Hypertext allows users to select how much they want to know about a topic, and since the hypertext is only generated on demand, many navigation problems common in fixed hypertexts are avoided.

ATMS technology [24] has been widely used in qualitative reasoning systems because of its ability to rapidly switch between alternate interpretations. As noted previously, this ability is unnecessary in CyclePad, and we suspect that this will be true for most educational applications.

6. Discussion

CyclePad demonstrates that qualitative physics has advanced enough to support new applications of AI to educational problems. Compositional modeling provides representational tools and techniques that can be used to encode a substantial body of knowledge about engineering thermodynamics, with constraint propagation providing analytic capabilities and qualitative representations providing the intuition needed to detect student blunders. Automatically generated hypertext explanations enable the user to explore the consequences of his or her assumptions, and figure out what modeling assumptions are needed to make further progress.

To date, CyclePad has only been tested with graduate student volunteers. We will be testing it with undergraduate engineering students both at Oxford and at Northwestern this academic year. Feedback will be gathered via a combination of electronic mail and interviews, which we will use to further improve the system. Our goal is to have CyclePad continuously available to undergraduates, so that their needs will help guide subsequent development.

Several extensions to CyclePad are in progress. First, we will extend it to handle non-steady flow cycles, such as Otto and Diesel cycles. Second, we will add some algebraic capabilities, so that CyclePad can help students derive algebraic expressions that capture important tradeoffs in specific systems.

We view CyclePad as part of a *virtual laboratory* for exploring thermodynamic cycles. A virtual laboratory is a software environment consisting of a set of parts, corresponding to physical parts or important abstractions in the domains of interest, tools for assembling collections of these parts into designs, and facilities for analyzing and testing designs. By working in this

software environment, students can "build" their designs and try them out without expense or danger. In simpler domains some commercial software exists that can be viewed as virtual laboratories (e.g., Interactive Physics for simple dynamics and Electronics Workbench). A novel contribution of qualitative physics is the ability to generate explanations. For educational applications, explanation generation is vital, to help students see what aspects of a situation are important and to tie what they are observing back to fundamental principles. One of our next steps is to extend CyclePad's explanation facilities, by adding *coaches* [25 , 26 , 27] to help students, both to guide them through the analysis process (including the representation of real devices in terms of ideal components) and to suggest improvements to a student's design.

7. Acknowledgments

This work was supported by a grant from the Science and Engineering Research Council in the UK and by grants from the Office of Naval Research and NASA Langley Research Center in the U.S.. We thank Yusuf Pisan and John Everett for many enlightening bug reports and suggestions.

8. References

1 Forbus, K. & Stevens, A. Using Qualitative Simulation to Generate Explanations. *Proceedings of the Cognitive Science Society*, August 1981.

2 Brown, J.S., Burton, R. & de Kleer, J. Pedagogical, natural language, and knowledge engineering techniques in SOPHIE I, II, and III. In Sleeman, D. and Brown, J.S. (Eds.), *Intelligent Tutoring Systems*, Academic Press, 1982.

3 White, B. & Frederiksen, J. Causal model progressions as a foundation for intelligent learning environments. *Artificial Intelligence*, **42**, 99-157.

4 Massey, L., de Bruin, J. and Roberts, B. A Training System for System Maintenance. In Psotka, J. Massey, L., and Mutter, S. *Intelligent Tutoring Systems: Lessons Learned*. Erlbaum, 1988.

5 Govindaraj, T. Qualitative approximation methodology for modeling and simulation of large dynamic systems: Applications to a marine steam power plant. *IEEE transactions on systems, man, and cybernetics*, vol SMC-17, no. 6, November/December 1987.

6 Masahiro Inui, et al. Development of a model-based intelligent training system for plant operations. *Proceedings of International Conference on ARCE*, Tokyo, pp 89-94, 1990.

7 Roschelle, J. Collaborative Conceptual Change: Jointly acting social and cognitive processes. *Proceedings of CogSci-93*.

8 Whalley, P. *Basic Engineering Thermodynamics*, Oxford University Press, 1992.

9 Haywood, R. W. *Analysis of Engineering Cycles: Power, Refrigerating and Gas liquefaction Plant*, Pergamon Press, 1985.

10 El-Wakil, M. *Powerplant Technology*, McGraw-Hill, 1984.

11 Stallman, R.M. and Sussman, G.J. Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis, *Artificial Intelligence* **9** (1977), 135--196.

12 Falkenhainer, B., and Forbus, K. Setting up large-scale qualitative models. Proceedings of AAAI-88, August, 1988.

13 Falkenhainer, B. and Forbus, K. Compositional Modeling: Finding the Right Model for the Job. *Artificial Intelligence*, **51**, (1-3), October, 1991.

14 Nayak, P. Automated modeling of physical systems. Ph.D. dissertation, Computer Science Department, Stanford University, 1992.

15 Addanki, S., Cremonini, R., & Penberthy, J.S., Reasoning about assumptions in graphs of models. *Proceedings of IJCAI-89*, 1989.

16 McAllester, D. An outlook on truth maintenance. MIT AI Lab memo AIM-551, 1980.

17 Forbus, K. and de Kleer, J. *Building Problem Solvers*, MIT Press, 1993.

18 Hayes, P. Naive Physics 1: Ontology for Liquids. In Hobbs, J. and Moore, R. (Eds.) *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, 1985.

19 Gruber, T. & Gautier, P. Machine-generated explanations of engineering models: A compositional modeling approach. *Proceedings of IJCAI-93*.

20 Skorstad, G. and Forbus, K. Qualitative and quantitative reasoning about thermodynamics, *Proceedings of the Cognitive Science Society*, August, 1989.

21 Steele, G. and Sussman, G.J. CONSTRAINTS: A language for expressing almost-hierarchical descriptions, *Artificial Intelligence*, **14** (1980):1--39.

22 Pisan, Y. Visual reasoning about physical properties via graphs, *submitted for publication*, 1994.

23 Reiter, E. & Mellish, C. Optimizing the costs and benefits of natural language generation, *Proceedings of IJCAI-93*, 1993.

24 de Kleer, J. An assumption-based truth maintenance system. *Artificial Intelligence*, **28**(1986): 127--162.

25 Burton, R. & Brown, J.S. An investigation of computer coaching for informal learning activities. In Sleeman, D. and Brown, J.S. (Eds.), *Intelligent Tutoring Systems*, Academic Press, 1982

26 Lesgold, A., Eggan, G., Katz, S. and Rao, G. Possibilities for Assessment Using Computer-Based Apprenticeship Environments. In Regian, J. & Shute, V. *Cognitive Approaches to Automated Instruction*. Erlbaum, 1992.

27 Schank, R.C., & Nohan, M.Y. Empowering the student: New Perspectives on the Design of Teaching Systems. *The Journal of the Learning Sciences*, **1**(7-35), 1991.