Knowledge Repositories for Multiple Uses

Keith Williamson, Patricia Riddle
Advanced Technology Center
Boeing Computer Services
P.O. Box 24346, MS 7L-64
Seattle, WA 98124-0346
(206) 865-3281

January 22, 1991

Abstract

In the life cycle of a complex physical device or part, for example, the docking bay door of the space station, there are many uses for knowledge about the device or part. The same piece of knowledge might serve several uses. Given the quantity and complexity of the knowledge that must be stored, it is critical to maintain the knowledge in one repository, in one form. At the same time, because of the quantity and complexity of knowledge that must be used in life cycle applications such as cost estimation, re-design and diagnosis, it is critical to automate such knowledge uses. For each specific use, a knowledge base must be available and must be in a form that promotes the efficient performance of that knowledge base. However, without a single source knowledge repository, the cost of maintaining consistent knowledge between multiple knowledge bases increases dramatically; as facts and descriptions change, they must be updated in each individual knowledge base.

We have developed a use-neutral representation of an hydraulic system for the F-111 airplane. We demonstrated the ability to derive portions of four different knowledge bases from this use-neutral representation: one knowledge base is for re-design of the device using a model-based reasoning problem solver; two knowledge bases, at different levels of abstraction, are for diagnosis using a model-based reasoning problem solver; and one knowledge base is for diagnosis using an associational reasoning problem solver. We have shown how updates issued against the single source use-neutral knowledge repository can be propagated to the underlying knowledge bases.

1 Problem

In the life cycle of a complex physical device or part (e.g., the docking bay of the space station) there are many uses for knowledge about the device or part, about its structure, function, materials, component parts and so on. Such uses might include product definition (i.e., design and manufacturing process planning), testability analysis and test construction, cost estimation, producibility studies, diagnosis of breakdowns, and re-design of features. In an effort to become more competitive, corporations are automating such knowledge uses. For each particular use, a knowledge base must be available and must be in a form that promotes efficient performance of that system. This has resulted in duplicate knowledge in multiple knowledge bases, where a particular piece of knowledge serves multiple uses.

Given the quantity and complexity of the knowledge that must be stored, it is critical to maintain the knowledge in one repository. Without a single source repository, the cost of maintaining the knowledge required by multiple knowledge bases is increased dramatically; as facts and descriptions change, they must be updated in each individual system. It is virtually impossible to maintain synchronicity among the knowledge bases that require the same knowledge. Without a single source repository, the costs of other maintenance tasks (e.g., knowledge verification and validation) are not easily shared across knowledge bases. Finally if there is no common knowledge repository, the development of new knowledge bases

must essentially be done from scratch each time.

To establish a single source knowledge repository, we cannot simply borrow representations from today's knowledge bases. In current knowledge base technology, the use of the knowledge determines its representation. The knowledge encoding is tailored for efficiency in performing the particular task at hand. However, once the knowledge representation is customized for a single use, it is difficult or impossible to apply it to some other use.

In order to maintain knowledge about a complex device in one knowledge repository for multiple uses, a use-neutral representation is required. In order to automate the use of that knowledge, it must be made available to many different problem-solvers and must be put in a form that promotes their efficiency. The need to manage and access single source information for multiple uses places critical requirements on the kind of knowledge that must be captured and the way the knowledge is represented in the repository.

Section 2 gives an overview of the objectives of our project. In section 3 the more detailed issues which arise from this research are discussed. Section 4 gives a survey of related research. In section 5 the initial focus of our project is given. Section 6 gives the progress made to date. In section 7 the future research directions are given. Section 8 concludes.

2 Project Objectives

The overall objective of this project is to develop a methodology for sharing knowledge between several knowledge bases which have different uses (e.g., design and process planning). This methodology consists of a single source use-neutral knowledge repository, and the ability to down-load this knowledge into different knowledge bases tailored for specific tasks. Specific objectives of the project include:

1. Demonstrate tools for transforming and downloading repository knowledge into knowledge bases that can be used by multiple problem solvers, that support the pre-defined range of life-cycle uses.

2. Allow the knowledge repository to be updated, and demonstrate the capability of automatically propagating these updates to the pertinent knowledge bases so as to preserve correctness.

3. Develop a methodology for creating a single source, use-neutral knowledge repository from a set of related use-specific knowledge bases.

4. Demonstrate the ability to automatically generate a first cut at a brand-new knowledge base from the knowledge repository.

5. Evaluate and establish knowledge representation principles for modeling physical devices/parts that support a pre-defined, but extensible, set of uses throughout the device/part's life cycle.

6. Establish requirements on the completeness and consistency of the single source, use-neutral knowledge repository imposed by the pre-defined range of life-cycle uses and develop tools to ensure those requirements are met by the single source model.

3 Issues Raised

The major benefits of such an approach are derived from three sources: from physically sharing knowledge, from updating the repository, and from adding new knowledge bases. From physically sharing knowledge, the knowledge bases have increased consistency with each other and a lower combined management cost and verification and validation (V&V) cost than if supported separately. From updating the repository and propagating the updates to the individual knowledge bases, the knowledge bases have a lower combined update maintenance cost and it is easier to maintain correctness and configuration control of the system. From adding new knowledge bases, this approach enables the identification of inconsistencies in pre-specified KBs and the partial automation of the generation of new KBs.

In this section, the issues raised in achieving these benefits are explored. In later sections, the

focus of our project within these issues and our initial progress will be described. The global structure of the ultimate system is shown in Figure 1. One important aspect of this structure is that it allows the transformations (i.e., the gateway) to access knowledge from several different knowledge stores. This is very important with regards to the Boeing Company. There are knowledge stores for Boeing Corporate Standards which are accessible by vendors and proprietary knowledge stores which are not. These knowledge stores might be stored and accessed separately for security reasons, but their knowledge might need to be combined to create useful knowledge base rules to be used within Boeing. To a system accessing the knowledge repository, the knowledge should appear seamless. Even with non-proprietary knowledge stores, the information might be stored separately (e.g., some in Auburn, Washington and some in Wichita, Kansas). This information must be accessed directly from these different sources so as to maintain the single source model. The ultimate goal is a distributed single source use-neutral knowledge repository.

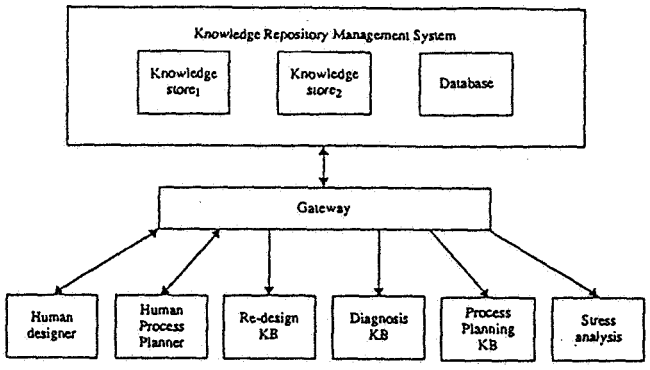

Figure 1: Single Source Repository

Another aspect of this model is that the language used in the use-neutral knowledge repository should have a sufficient expressive adequacy to allow new knowledge bases and their related transformations to be added to the repository later on. The knowledge in the single source repository is not use-neutral in the broadest sense. The knowledge must be represented to serve the "use" of transmitting the pertinent knowledge to multiple knowledge bases. But the knowledge representation used in the single source repository is independent of the specific uses of those knowledge bases, so it will be referred to as use-neutral throughout this paper.

Transformations which down-load knowledge must preserve the aspects of the model which are important for this knowledge base use (i.e., they must be behaviorally equivalent). In certain cases when a knowledge base requires all the details of the model this transformation will be the standard notion of equivalence (i.e., an isomorphic transformation). But in other cases transformations will either abstract away certain knowledge which is not necessary for this knowledge base or approximate knowledge for efficiency reasons. In these cases the transformation will only be behaviorally equivalent. The transformation will be equivalent only with respect to the desired behavior (i.e., a homomorphic transformation). For instance, if stress analysis is done, certain knowledge about the color of the paint or the smoothness of the finishes are not relevant knowledge with respect to the stress analysis.

The ultimate transformational system we envision is a partial-order of transformations (i.e., most likely a forest) from one representation to another. The reformulation from any one representation into another representation (i.e., either the use-neutral representation, a user representation, or the representation associated with a specific problem solver) is a path through this partial-order of transformations. This will allow a maximal amount of reuse of the transformations. Figure 2 illustrates this idea.

The KRMS (Knowledge Repository Management System) must deal with issues of V&V (e.g., consistency and redundancy) and configuration management across multiple knowledge stores. Configuration management includes keeping a history of any updates to the use-neutral knowledge repository so that the version of the knowledge given to a certain knowledge base or person at a certain time can be reconstructed. There is a tradeoff between applying V&V directly to the knowledge repository and applying it to each knowledge base separately. Some tests (e.g., consistency checking) might be best applied to the knowledge repository

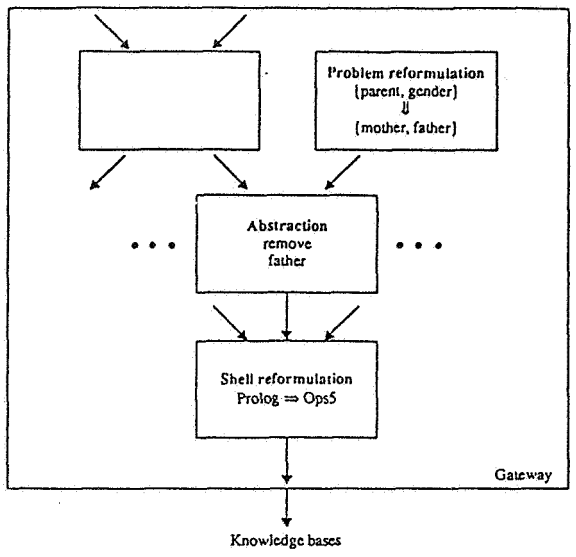

Figure 2: The Gateway Transformations

to assure synchronicity between all the knowledge bases and to avoid the cost of testing each knowledge base separately. Other tests (e.g., functionality testing) might be best applied to the task-specific knowledge base. It is not clear how to test the functionality of the knowledge repository directly.

The knowledge down-loaded to the task-specific knowledge bases must be consistently updated when an update is performed on the knowledge repository. There could be problems with this updating process within either the knowledge repository or the transformations. The representation of the knowledge in the knowledge repository might not be expressive enough to handle the update. The transformations might no longer be applicable to the updated knowledge in the repository or might transform the knowledge such that the knowledge in the knowledge bases is inconsistent with the knowledge in the repository (i.e., the transformation might not be correctness preserving on the updated knowledge). In a similar vein when a new knowledge base is connected to the knowledge repository, two analogous problems can occur. Either the repository does not contain the information necessary for the new knowledge base or the transformations cannot transform the knowledge into the form that the new knowledge base needs.

Updating the knowledge repository could be done using knowledge acquisition techniques. These can be seen as another type of transformation. These types of transformations must be handled differently because there is a person at one end of the transformation instead of a computer, but the basic structure of transforming one representation (a user language) into the use-neutral representation is the same. For instance a dialogue apparatus can be used to elicit knowledge from a person, but a dialog apparatus is difficult to use in a transformation between two computers. Transformations may also connect the use-neutral representation to people so as to give them direct access to the knowledge in the repository. Several different transformations (in each direction) may be necessary since the language used by different people (i.e., designers, manufacturers, maintainers) may not necessarily be the same. Transformations between people and machines might involve a form (possibly simplified) of natural language understanding and text generation.

Gateway security will also be an issue. Vendors must be allowed access to non-proprietary knowledge while users within Boeing must be allowed greater access. Even within Boeing there might be separate grades of accessibility, a factory shop-floor might not be allowed the same access rights as other areas.

4 Survey

In the last few years much work has concerned developing and maintaining large knowledge bases. Very little of this has focused on developing and maintaining large knowledge bases which support multiple uses. There has been some research involved in deriving a "shallow" model from a "deeper" one but not for systems with more than two models. Davis [2] has a system which can derive a sequence of models for use in troubleshooting digital circuits, but it cannot generate different models for different uses.

Rich Keller[1] has demonstrated the multiple use capability of the Stanford modeling project. Keller's [5] system derives two sequences of models, one for diagnosis and one for re-design. This research demonstrated the multiple use capability of the Stanford modeling project. He took approximately

[1] Previously on the Stanford project, now at NASA Ames.

20 diagnosis rules from a 150 rule Lockheed expert system for diagnosis of the Reaction Wheel Assembly (RWA) for the Hubble Space Telescope. Discussions with domain experts derived approximately 5 plausible redesign rules for the RWA. Both of these sets of rules were represented in the expert system shell Strobe. Given these two sets of rules a use-neutral representation was formulated and transformations for connecting these two specific rule sets with the use-neutral representation were derived.

A representation language/s must be chosen for the single source repository. As was stated earlier a single language may not be capable of expressing all the types of knowledge which must be stored in the repository. This language must be capable of representing both deductive and heuristic information and in a form which is independent of its intended use. There are several efforts which deal with creating representation languages which are capable of representing "all" knowledge regardless of use: CYC at MCC [7] and modeling of scientific and engineering device knowledge at Stanford [4]. These representation languages and their associated tools might be suitable for representing the knowledge in our single source repository. The domain representation within these languages must also be determined. It is possible that none of these projects have a representation which will suit our purposes. The Stanford project, since it is also dealing with engineering devices, is more likely to have a predefined problem representation which will suit our purposes. The MCC project is trying to represent "all" knowledge. We are, as is Stanford, trying to solve the easier problem of representing "relevant engineering" knowledge. We have purposefully not spent a lot of time exploring these representation languages. We feel that the possible choice of a representation language should be driven by the needs of the use-neutral representation. We plan to have a first cut of what the requirements of the representation are before we explore these representation languages in depth. This will help to avoid a preconceived bias on our parts.

Research in reformulating knowledge has also been pursued in the last few years. There has been work on abstraction transformations and to a lesser extent on approximation abstractions [6, 3, 14, 15]. Within correctness preserving transformations there has been work on both shifting between expert system shells (i.e., analogous to compilers) [10, 13, 11] and shifting between problem representations (i.e., traditional reformulation) [1, 8, 12, 9]. None of this work has been based on large knowledge bases and therefore not explored certain relevant issues (e.g., knowledge base management, knowledge base maintenance, and version control).

5 Focus

Previously we gave an overview of all the issues related to a single source knowledge repository. It is important at this point to emphasis which of these issues this project is emphasizing. These aspects of the system were chosen to allow the system to have a functional value to Boeing within 5 years while research on the other issues continues. The following four issues are the focus of this project.

A representation language is needed which is capable of expressing the information necessary to handle all the different desired uses of the knowledge. This is a tall order. The approach we take is to try and circumscribe the uses to which we will expect this knowledge to be put. The goal is to represent sufficient knowledge to achieve this set of uses as opposed to any use defined later on . This representation language should be flexible enough to allow extensions to the set of pre-defined uses. It is also apparent that one pre-existing representation language may not be suitable for handling all the different type of knowledge necessary (i.e, deductive knowledge versus heuristic knowledge). Portions of multiple pre-existing representation languages may have to be used within the knowledge repository. Bear in mind that this does not necessarily dictate that redundant knowledge will be represented.

Transformations are necessary to down-load the knowledge in the repository to multiple knowledge bases. We plan to explore three basic types of transformations: abstraction, approximation, and correctness preserving. Frequently the full model of a system is too complex to analyze, due to computational restrictions. In these circumstances a simpler

model of the system must be used for the analysis. The transformation from the full model to a simpler one is an abstraction. These are used frequently in engineering domains where the complexity of the full system is often overwhelming. Approximations are a similar type of transformation. An abstracted model is correct and must just be refined to arrive at the original full model. An approximate model is actually incorrect to some allowed degree of error (i.e., the approximate model cannot be refined into the original full model). These types of transformations are also used frequently throughout engineering domains (i.e., the simplex method). The above two types of transformations are homomorphisms. Correctness preserving transformations change one model into an equivalent model (i.e., it is an isomorphism). Two examples of when this type of transformation is important is a shift between the problem solvers' associated language (i.e., OPS5 to Prolog) or a problem reformulation. A problem reformulation is a shift from one representation of a problem to another within the same representation language. For instance the shift between a model containing **gender** and **parent** to a model containing **mother** and **father**. It turns out that this last type of transformation is very important in achieving a model of the problem which is computationally efficient for a particular problem solver.

The ability to update the single source knowledge repository such that the transformations to the knowledge bases are still applicable and correctness preserving is very important. Note that a transformation between two representations could trivially be $A \Rightarrow B$. But this type of representation is not very general and therefore will not be applicable if A is updated to A'. It is very difficult to create transformations which allow all possible types of updates. We plan to circumscribe the set of updates for which the transformations are guaranteed to still be applicable. This allows the system to actually be used with production level knowledge bases and thus simplify a class of their update problems and lower their maintenance costs while research on broadening the class of guaranteed updates is continued.

The ability to decide later on to connect a new knowledge base to the knowledge repository is also important. Analogous to the problem with updating, it is very difficult to allow the easy connection of any new knowledge base. We plan to circumscribe the set of knowledge bases for which the transformations are guaranteed to connect. This allows the system to actually be used with production level knowledge bases and thus simplify the creation of a class of new knowledge bases and lower their start-up costs while research on broadening the class of connectable knowledge bases is continued.

6 Progress

The progress made by this project in slightly under 3 person-months was a feasibility study using a knowledge base for the F-111 hydraulics system. We established a use-neutral knowledge repository for the F-111 hydraulics system and a set of transformations from this knowledge repository to multiple knowledge bases each for a specific task. In this feasibility study there were four knowledge bases derived: one knowledge base for design using a model-based reasoning problem solver[2]; two knowledge bases (at different levels of abstraction) for diagnosis using a model-based reasoning problem solver; and one knowledge base for diagnosis using an associational reasoning problem solver. The derived associational diagnosis rules were more precise than person derived rules. We demonstrated the connection of a pre-specified knowledge base to the use-neutral knowledge repository; in doing this we discovered inconsistencies in the pre-specified knowledge base. We demonstrated the propagation of updates, which were made to the use-neutral knowledge repository, down to the pertinent knowledge bases. We also demonstrated a partially automated methodology for the derivation of a use-neutral knowledge repository and its associated transformations from a set of related use-specific knowledge bases.

6.1 F-111 Hydraulic Models

The hydraulic system of the F-111 aircraft is shown in Figure 4. The hydraulic system is broken into

[2]The use of model-based reasoning problem solvers for design is just beginning to be explored[16].

two sub-systems; a primary and a utility subsystem. Within both of these subsystems, there are redundant modules (i.e., left and right modules) and a pressure indicator. Each of these modules consists of a pump and a pressure indicator. The right modules of each subsystem share an engine, as do the left. The system is assumed to be given the flow-demand and throttle settings for each of the engines.

Abbreviations	
p	primary
pl	primary.left
pr	primary.right
ip	intermediate.primary
ipl	intermediate.primary.left
ipr	intermediate.primary.right
u	utility
ul	utility.left
ur	utility.right
iu	intermediate.utility
iul	intermediate.utility.left
iur	intermediate.utility.right
le	left.engine
re	right.engine

Constants	
Trcf	throttle.rpm.conversion.factor
Rtfcc	rpm.to.flow.conversion.constant
Ec	Engine constant
Ku	utility pipe constant
Kp	primary pipe constant
Kul	utility left pipe constant
Kur	utility right pipe constant
Kpl	primary left pipe constant
Kpr	primary right pipe constant

Status Variables	
p.pressure.status	u.pressure.status
pl.pressure.status	pl.status
pr.pressure.status	pr.status
ul.pressure.status	ul.status
ur.pressure.status	ur.status
le.status	re.status

Figure 3: Abbreviations and Constants

In the models for the knowledge bases and repository, there are some constant values and common abbreviations used in specifying variable names. The models also contain references to boolean status variables that are assumed to indicate the operational status of certain components. These are shown in Figure 3. In each of the task-specific models, we indicate which of the model variables are assumed to be instantiated prior to use of that model. There were four constraints which were common to all the derived models (see Figure 5). The relationships between the models are shown in Figure 6. The use-neutral model has 17 constraints on the components; these are shown in Figure 7. These models are used to illustrate the technology involved with a use-neutral knowledge repository; there is no claim that they precisely reflect the actual physics involved.

In the Diagnosis I model (see Figure 8), the areas of the pipes are fixed and the resistance of the pipe has been abstracted (i.e., the pipe resistance is 0). In the Diagnosis II model (see Figure 9), the areas of the pipes are fixed but the resistance of the pipe is not abstracted. The Design model (see Figure 10) also abstracts the resistance of the pipe but does not fix the areas of the pipes (i.e., while during diagnosis the pipes should not be allowed to change size, this is desirable in a design model).

Chronology These models were derived as follows. Diagnosis I was chosen as our initial model; it was previously developed by another research project in the Advanced Technology Center. Using this model, high-level descriptions of the Diagnosis II (by adding the notion of pipe resistance) and Design (by adding the notion that pipe radii can change) models were hypothesized. Using these three models (two of which were only high-level descriptions) and knowledge of basic physics, the use-neutral model was hypothesized. Using the use-neutral model, the assumptions made by the various models, and a set of general transformations; the exact set of constraints specified in Diagnosis I and a plausible set for Diagnosis II and Design were derived.

6.2 Transformations

The transformations used in deriving the task-specific models will now be discussed. The reformulation of the use-neutral into Diagnosis I is not shown here; it is a combination of the transformations used to reach the other two models.

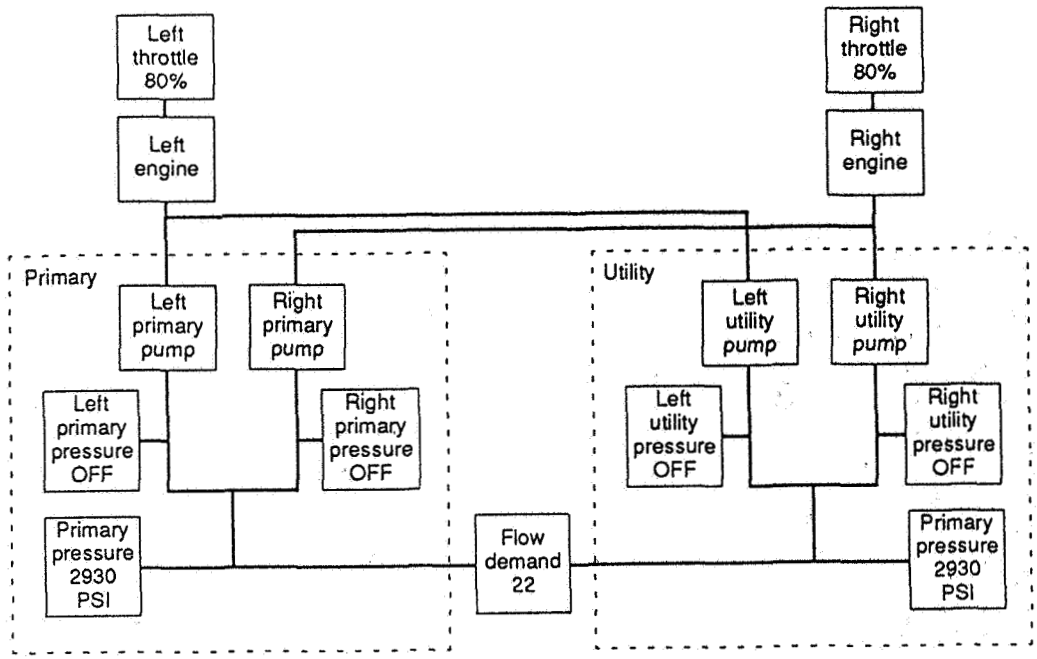

Figure 4: Hydraulic System for F-111

1 flow-demand-rule
if T then u.flow.demand = sys.flow.demand/2
if T then p.flow.demand = sys.flow.demand/2

2 flow-large-rule
if T then u.flow = ul.flow+ur.flow
if T then p.flow = pl.flow+pr.flow

16 reading-rule
if u.pressure.status=0 then
  u.pressure.indicator = u.pressure
if p.pressure.status=0 then
  p.pressure.indicator = p.pressure

17 pressure-indicator-constraint
if pl.pressure.status=0 then
  if pl.pressure>1300 then pl.pressure.indicator=1
    else pl.pressure.indicator=0
if pr.pressure.status=0 then
  if pr.pressure>1300 then pr.pressure.indicator=1
    else pr.pressure.indicator=0
if ul.pressure.status=0 then
  if ul.pressure>1300 then ul.pressure.indicator=1
    else ul.pressure.indicator=0
if ur.pressure.status=0 then
  if ur.pressure>1300 then ur.pressure.indicator=1
    else ur.pressure.indicator=0

Figure 5: Rules common to ALL Models

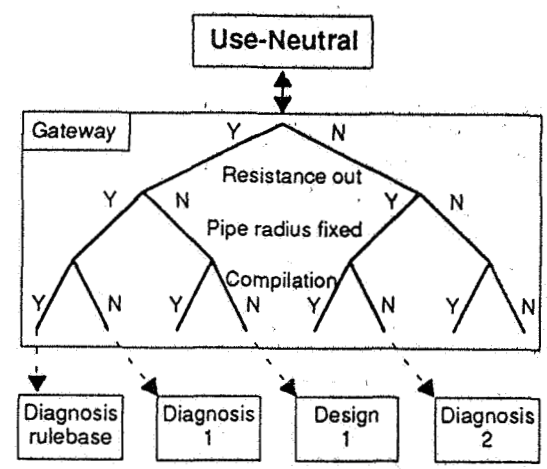

Figure 6: Relationships between Models

Use-Neutral to Diagnosis II This is a fairly straight forward set of transformations. First the assumptions are added (i.e., that the radii are constant). This allows the use-neutral constraints (UNC) 4 and 10 in Figure 7 to become constants (i.e., the pipe areas are now constants). The intermediate results computed in UNC 5 can be folded into UNC 3 since no other constraint refers to them and they are not tested by a sensor. This gives the set of constraints shown in Figure 9. This highlights two general transformation techniques: compiling constraints into constants and folding constraints into each other.

Inputs	
sys.flow.demand	fluid.density
le.throttle	re.throttle
u.radius	p.radius
ul.radius	pl.radius
ur.radius	pr.radius

3 flow-small-rule
if ul.status=0 then
  ul.flow = ul.area×fluid.density×ul.velocity
if ur.status=0 then
  ur.flow = ur.area×fluid.density×ur.velocity
if pl.status=0 then
  pl.flow = pl.area×fluid.density×pl.velocity
if pr.status=0 then
  pr.flow = pr.area×fluid.density×pr.velocity

4 area-small-rule
if T then ul.area = π×ul.radius^2
if T then ur.area = π×ur.radius^2
if T then pl.area = π×pl.radius^2
if T then pr.area = π×pr.radius^2

5 velocity-rule
if ul.status=0 then ul.velocity = Ec×le.rpm
if ur.status=0 then ur.velocity = Ec×re.rpm
if pl.status=0 then pl.velocity = Ec×le.rpm
if pr.status=0 then pr.velocity = Ec×re.rpm

6 rpm-rule
if le.status=0 then le.rpm = Trcf×le.throttle
if re.status=0 then re.rpm = Trcf×re.throttle

7 final-pressure-rule
if u.flow≥u.flow.demand then u.pressure=2930
  else u.pressure=2930×(u.flow/u.flow.demand)
if p.flow≥p.flow.demand then p.pressure=2930
  else p.pressure=2930×(p.flow/p.flow.demand)

8 intermediate-large-pressure-rule
if ul.status=0 & ur.status=0 then
  iu.pressure = u.pressure+(u.resistance×u.flow^2)
if pl.status=0 & pr.status=0 then
  ip.pressure = p.pressure+(p.resistance×p.flow^2)

Figure 7: Use-Neutral Model

9 resistance-large-rule
if T then u.resistance = Ku/(2×fluid.density×u.area^2)
if T then p.resistance = Kp/(2×fluid.density×p.area^2)

10 area-large-rule
if T then u.area = π×u.radius^2
if T then p.area = π×p.radius^2

11 force-large-rule
if ul.status=0 & ur.status=0 then
  u.force = iu.pressure×u.area
if pl.status=0 & pr.status=0 then
  p.force = ip.pressure×p.area

12 force-small-rule
if ul.status=0 & ur.status=0 then
  u.force = ur.force+ul.force
if pl.status=0 & pr.status=0 then
  p.force = pr.force+pl.force

13 intermediate-small-pressure-rule
if ul.status=0 then iul.pressure = ul.force/ul.area
if ur.status=0 then iur.pressure = ur.force/ur.area
if pl.status=0 then ipl.pressure = pl.force/pl.area
if pr.status=0 then ipr.pressure = pr.force/pr.area

14 pressure-small-rule
if ul.status=0 then ul.pressure =
  iul.pressure+(ul.resistance×ul.flow^2)
if ur.status=0 then ur.pressure =
  iur.pressure+(ur.resistance×ur.flow^2)
if pl.status=0 then pl.pressure =
  ipl.pressure+(pl.resistance×pl.flow^2)
if pr.status=0 then pr.pressure =
  ipr.pressure+(pr.resistance×pr.flow^2)

15 resistance-small-rule
if T then ur.resistance=Kur/(2×fluid.density×ur.area^2)
if T then ul.resistance=Kul/(2×fluid.density×ul.area^2)
if T then pr.resistance=Kpr/(2×fluid.density×pr.area^2)
if T then pl.resistance=Kpl/(2×fluid.density×pl.area^2)

Figure 7 Continued: Use-Neutral Model

Inputs	
sys.flow.demand	
re.throttle	le.throttle

3 flow-small-rule
if ul.status=0 then ul.flow = Rtfcc×le.rpm
if ur.status=0 then ur.flow = Rtfcc×re.rpm
if pl.status=0 then pl.flow = Rtfcc×le.rpm
if pr.status=0 then pr.flow = Rtfcc×re.rpm

4 rpm-rule
if le.status=0 then le.rpm = Trcf×le.throttle
if re.status=0 then re.rpm = Trcf×re.throttle

5 final-pressure-rule
if u.flow≥u.flow.demand then u.pressure=2930
  else u.pressure=2930×(u.flow/u.flow.demand)
if p.flow≥p.flow.demand then p.pressure=2930
  else p.pressure=2930×(p.flow/p.flow.demand)

6 pressure-small-rule
if ul.status=0 then
  if ul.flow<u.flow.demand/2 then
    ul.pressure=0 else ul.pressure=u.pressure
if ur.status=0 then
  if ur.flow<u.flow.demand/2 then
    ur.pressure=0 else ur.pressure=u.pressure
if pl.status=0 then
  if pl.flow<p.flow.demand/2 then
    pl.pressure=0 else pl.pressure=p.pressure
if pr.status=0 then
  if pr.flow<p.flow.demand/2 then
    pr.pressure=0 else pr.pressure=p.pressure

Figure 8: Diagnosis I Model

Inputs	
sys.flow.demand	fluid.density
re.throttle	le.throttle
u.area	p.area
ul.area	ur.area
pl.area	pr.area

3 flow-small-rule
if ul.status=0 then ul.flow = Rtfcc×le.rpm
if ur.status=0 then ur.flow = Rtfcc×re.rpm
if pl.status=0 then pl.flow = Rtfcc×le.rpm
if pr.status=0 then pr.flow = Rtfcc×re.rpm

4 rpm-rule
if le.status=0 then le.rpm = Trcf×le.throttle
if re.status=0 then re.rpm = Trcf×re.throttle

5 final-pressure-rule
if u.flow≥u.flow.demand then u.pressure=2930
  else u.pressure=2930×(u.flow/u.flow.demand)
if p.flow≥p.flow.demand then p.pressure=2930
  else p.pressure=2930×(p.flow/p.flow.demand)

6 intermediate-large-pressure-rule
if ul.status=0 & ur.status=0 then
  iu.pressure=u.pressure+(u.resistance×u.flow^2)
if pl.status=0 & pr.status=0 then
  ip.pressure=p.pressure+(p.resistance×p.flow^2)

7 resistance-large-rule
if T then u.resistance=Ku/(2×fluid.density×u.area^2)
if T then p.resistance=Kp/(2×fluid.density×p.area^2)

8 force-large-rule
if ul.status=0 & ur.status=0 then
  iu.pressure = u.force/u.area
if pl.status=0 & pr.status=0 then
  ip.pressure = p.force/p.area

9 force-small-rule
if ul.status=0 & ur.status=0 then
  u.force = ur.force+ul.force
if pl.status=0 & pr.status=0 then
  p.force = pr.force+pl.force

Figure 9: Diagnosis II Model

10 intermediate-small-pressure-rule
if ur.status=0 then iur.pressure = ur.force/ur.area
if ul.status=0 then iul.pressure = ul.force/ul.area
if pr.status=0 then ipr.pressure = pr.force/pr.area
if pl.status=0 then ipl.pressure = pl.force/pl.area

11 pressure-small-rule
if ul.status=0 then ul.pressure =
  iul.pressure+(ul.resistance×ul.flow^2)
if ur.status=0 then ur.pressure =
  iur.pressure+(ur.resistance×ur.flow^2)
if pl.status=0 then pl.pressure =
  ipl.pressure+(pl.resistance×pl.flow^2)
if pr.status=0 then pr.pressure =
  ipr.pressure+(pr.resistance×pr.flow^2)

12 resistance-small-rule
if T then ur.resistance=Kur/(2×fluid.density×ur.area^2)
if T then ul.resistance=Kul/(2×fluid.density×ul.area^2)
if T then pr.resistance=Kpr/(2×fluid.density×pr.area^2)
if T then pl.resistance=Kpl/(2×fluid.density×pl.area^2)

Figure 9 Continued: Diagnosis II Model

Use-Neutral to Design This is a more complex set of transformations; the intermediate constraints are shown in Figure 11. First the assumptions are added (i.e., that the pipe resistances are 0). This allows UNC 9 and 15 to become constants; these constants are replaced in UNC 8 and 14 to produce 8′ and 14′. Since 8′ and 14′ are now equalities, they can be substituted into UNC 11 and 13 and removed. This produces 11′ and 13′. Notice that this can only be done since the antecedents of constraints 8′ and 14′ are implied by the antecedents of constraints UNC 11 and 13 (this type of requirement occurs throughout these transformation sequences).

The assumptions that the left and right pipes for each subsystem have equal sizes and that the pipes out of each subsystem are twice the size of these inflow pipes are added; transforming 13′ into 13″. The constraint 11′ is now substituted into the constraint UNC 12 and removed; producing 12′. The constraint 12′ is substituted into the constraint 13″ and removed; producing 13‴. At this point UNC 10 can also be removed[2]. The assumption that the forces through the left and right modules of each subsystem are equal is added. This allows the in-

Inputs	
sys.flow.demand	fluid.density
re.throttle	le.throttle
ul.radius	ur.radius
pl.radius	pr.radius

3 flow-small-rule
if ul.status=0 then
  ul.flow = ul.area×fluid.density×ul.velocity
if ur.status=0 then
  ur.flow = ur.area×fluid.density×ur.velocity
if pl.status=0 then
  pl.flow = pl.area×fluid.density×pl.velocity
if pr.status=0 then
  pr.flow = pr.area×fluid.density×pr.velocity

4 area-small-rule
if T then ul.area = π×ul.radius^2
if T then ur.area = π×ur.radius^2
if T then pl.area = π×pl.radius^2
if T then pr.area = π×pr.radius^2

5 velocity rule (small pipe)
if T then ul.velocity = Ec×le.rpm
if T then ur.velocity = Ec×re.rpm
if T then pl.velocity = Ec×le.rpm
if T then pr.velocity = Ec×re.rpm

6 rpm-rule
if le.status=0 then le.rpm = Trcf×le.throttle
if re.status=0 then re.rpm = Trcf×re.throttle

7 final-pressure-rule
if u.flow≥u.flow.demand then u.pressure=2930
  else u.pressure=2930×(u.flow/u.flow.demand)
if p.flow≥p.flow.demand then p.pressure=2930
  else p.pressure=2930×(p.flow/p.flow.demand)

8 pressure-small-rule
if ul.status=0 then
  if ul.flow<u.flow.demand/2 then
    ul.pressure=0 else ul.pressure=u.pressure
if ur.status=0 then
  if ur.flow<u.flow.demand/2 then
    ur.pressure=0 else ur.pressure=u.pressure
if pl.status=0 then
  if pl.flow<p.flow.demand/2 then
    pl.pressure=0 else pl.pressure=p.pressure
if pr.status=0 then
  if pr.flow<p.flow.demand/2 then
    pr.pressure=0 else pr.pressure=p.pressure

Figure 10: Design Model

8′ intermediate-large-pressure-rule
if ul.status=0 & ur.status=0 then
  iu.pressure = u.pressure
if pl.status=0 & pr.status=0 then
  ip.pressure = p.pressure

14′ pressure-small-rule
if ul.status=0 then ul.pressure = iul.pressure
if ur.status=0 then ur.pressure = iur.pressure
if pl.status=0 then pl.pressure = ipl.pressure
if pr.status=0 then pr.pressure = ipr.pressure

11′ force-large-rule
if ul.status=0 & ur.status=0 then
  u.force = u.pressure×u.area
if pl.status=0 & pr.status=0 then
  p.force = p.pressure×p.area

13′ intermediate-small-pressure-rule
if ul.status=0 then ul.pressure = ul.force/ul.area
if ur.status=0 then ur.pressure = ur.force/ur.area
if pl.status=0 then pl.pressure = pl.force/pl.area
if pr.status=0 then pr.pressure = pr.force/pr.area

13″ intermediate-small-pressure-rule
if ul.status=0 then ul.pressure = (2*ul.force)/u.area
if ur.status=0 then ur.pressure = (2*ur.force)/u.area
if pl.status=0 then pl.pressure = (2*pl.force)/p.area
if pr.status=0 then pr.pressure = (2*pr.force)/p.area

12′ force-small-rule
if ul.status=0 & ur.status=0 then
  u.pressure×u.area = ur.force+ul.force
if pl.status=0 & pr.status=0 then
  p.pressure×p.area = pr.force+pl.force

13‴ intermediate-small-pressure-rule
if ul.status=0 then ul.pressure =
  (2*ul.force)/((ur.force+ul.force)/u.pressure)
if ur.status=0 then ur.pressure =
  (2*ur.force)/((ur.force+ul.force)/u.pressure)
if pl.status=0 then pl.pressure =
  (2*pl.force)/((pr.force+pl.force)/p.pressure)
if pr.status=0 then pr.pressure =
  (2*pr.force)/((pr.force+pl.force)/p.pressure)

13⁗ intermediate-small-pressure-rule
if ul.status=0 then ul.pressure = u.pressure
if ur.status=0 then ur.pressure = u.pressure
if pl.status=0 then pl.pressure = p.pressure
if pr.status=0 then pr.pressure = p.pressure

Figure 11: Intermediate Constraints

termediate values computed in 12′ to be removed[3] and 13‴ is transformed into 13⁗. The last two assumptions added are that the flows for each module in a subsystem are equal and that the pressure for each module is either 0 or its maximum value (i.e., 2930) which alters 13⁗ into constraint 8 of the design model (see Figure 10). It is important to note that for the task-specific knowledge base to be consistent some of the assumptions have to explicitly remain (i.e., they cannot be totally compiled into the previous constraints). These assumptions are not explicitly shown in our models. For instance for the Design model the assumptions which are not totally compiled are *ul.area=ur.area, pl.area=pr.area, ul.force=ur.force, pl.force=pr.force, ul.pressure = 2930 ∨ ul.pressure = 0, ur.pressure = 2930 ∨ ur.pressure = 0, pl.pressure = 2930 ∨ pl.pressure = 0, pr.pressure = 2930 ∨ pr.pressure = 0.*

Consistency versus Pre-defined KB If a predefined knowledge base is added to the system a choice must frequently be made. Either the transformations are applied consistently given a specific set of assumptions or they are partially applied so as to retain the exact representation of the original knowledge base. When the latter is chosen, the inconsistent application of the transformations can allow the knowledge base to be inconsistent. The inconsistent application of transformations can be a flag highlighting these inconsistencies and bringing them to the attention of the creator of the knowledge base. For instance in the Diagnosis I model shown in Figure 8, constraints 5 and 6 are inconsistent with each other. This occurs because the assumption that the pressure for each module is either 0 or the maximum is compiled into constraint 6 but not into constraint 5. This inconsistency was preserved so as to achieve the exact representation of the original pre-defined set of constraints for Diagnosis I. The use of our technique highlighted this inconsistency which was not noticed before. Even if the transformations are applied consistently, the

[3]Since the force variables are not input or appear in any other formulas, this formula can only be used to determine values for these forces. And since these forces are never sensed, there is no need to retain the constraints.

derived knowledge base is not necessarily consistent with the original one. But it is consistent modulo its initial input (i.e., the use-neutral knowledge repository, assumptions made, and transformations used).

6.3 Associational Rules

An algorithm for generating an associational knowledge base for diagnosis was demonstrated. Figure 12 shows a rule for the Diagnosis I model. The associational rules were derived using a back-propagation algorithm over the results of a model-based reasoner. For instance the previous rule was derived when the symptom *ur.pressure.status = 1 & ur.pressure.status = 0* and fault *ur.pressure.indicator* were discovered by a model-based reasoner. The symptom was back-propagated over the constraints associated with each component. When the result is totally in terms of the primitives of the model-base reasoner, then the back-propagation terminates and the resulting expression is the antecedent of the associational rule. The rule in Figure 12 was more precise than the rule derived for the same scenario by the person who created the model-base reasoning model.

```
if ur.pressure.status.reading=1&ur.pressure.status=0&
   le.status=0&re.status=0&ul.status=0&ur.status=0&
   [(Rtfcc×Trcf×(le.throttle+re.throttle))>
      (0.22×sys.flow.demand)]&
   (Rtfcc×Trcf×re.throttle)>=.25×sys.flow.demand
 then faulty(ur.pressure.indicator)
```

Figure 12: Associational Diagnosis Rule

6.4 Updates

This methodology is most beneficial when updates are performed on the use-neutral knowledge repository and then propagated to the pertinent knowledge bases. We explored the propagation of updates by running the entire updated knowledge repository through the same transformations again to derive the changes to the knowledge bases. Two specific updates were explored in depth. The first adds the notion of *work* to the knowledge repository in terms of its relationship to force. The transformations determine that this concept is irrelevant for these tasks and derive the same set of knowledge bases as before. Then a second update removes force. After this update the transformations produce knowledge bases which use work divided by distance as a replacement for force in all the pertinent constraints.

There are several problems with this method of updating. The transformations should be applied incrementally after an update. Instead of reprocessing the entire knowledge repository, only the pertinent subportions are reprocessed. This is very difficult, because an update may cause another piece of knowledge in the repository to be transformed differently. Also transformations might be applicable where they were not applicable before the update. Even more difficult is the requirement that no other transformations (outside the set which were applied before the update) are applicable now. Further a change to the repository might make the old transformations invalid. When this happens the knowledge repository might no longer be able to reach the existing representation of the knowledge base anymore.

Handling updates is a hard problem, but classes of updates can be defined which are guaranteed to transform correctly. We must explore updates in many domains so as to define this class.

6.5 Creating a Knowledge Repository

The research done in this area was very preliminary in nature. Given several pre-existing knowledge bases which contain overlapping knowledge, the generation of a knowledge repository can be partially automated. Their knowledge is compared and the most primitive components of the knowledge bases are used to create the knowledge repository. This can be done via partial matching as follows. Given two knowledge bases, all those rules which are an exact match are automatically placed in the use-neutral repository. For pairs of rules which score high on the partial match, assumptions are hypothesized which allow one of the rules to be transformed into the other. At this point user intervention could avoid nonplausible assumptions. The use-neutral repository is constructed from the knowl-

edge base rules to which assumptions were added to reach other rules. The complexity of this technique increases with the number of knowledge bases, but the number of high scoring partial matches increase inversely.

Notice that this technique relies on one rule being an abstraction of another. This methodology will not work when none of the knowledge bases have the most primitive knowledge, but instead are all slightly different abstractions of this knowledge. For instance, assume the primitive knowledge is $A=B\times C^2$. If one knowledge base had $A=0$ and the other knowledge base had $A=100\times B$, then the system will not be able to hypothesis the primitive knowledge on its own. It assumes that $A=B\times C$ was the primitive knowledge using the assumptions that $C=0$ or $C=100$. There is no reason for it to choose the more complex (and in this case correct) set of assumptions. Either a person must aid the derivation of the knowledge repository or another knowledge source containing general purpose primitive knowledge (e.g., a fluid dynamics knowledge seed) must be brought to bear. If the knowledge bases representations are too different the system also fails. For instance comparing the notion of *equalitaral-triangle* and *triangle-where-all-the-sides-are-equal-length.*

We must explore more domains, to determine what techniques will be useful in which situations.

7 Future Research

Exploration of this technology's capacity for generating new knowledge-based systems is needed. First cuts at totally new knowledge bases can be generated. For instance assume that a design knowledge base for a device A and a diagnosis knowledge base for a device B are presently connected to the knowledge repository, we need to derive a diagnostic knowledge base for device A. The same knowledge about the device used in deriving the design knowledge base should be used, but it should now be transformed for a diagnostic task. To derive a first cut at this knowledge base the knowledge concerning device A can be run through the set of transformations previously used on device B. Some transformations are likely not to fire and other necessary transformations are likely to be missing, but the hypothesis is that a good first cut at a knowledge base is generated.

The next logical step in this research is to implement these ideas so as to test them in several domains. To explore the update question in depth, a more complex domain is necessary. We are looking for a NASA domain which meets these requirements. More work is needed on the initial derivation of the knowledge repository and on transformations between multiple problem solvers. Transformations based on approximations (as well as abstractions) and the handling of inverse updates (i.e., updating a knowledge base which is then propagated up to the knowledge repository and then back down to the other pertinent knowledge bases) should be explored.

8 Summary

In the life cycle of complex devices (i.e., the docking bay door of the space station) , there are many processes (e.g., design, process planning, and diagnosis) that can be partially automated by knowledge-based systems. As these knowledge-based systems proliferate, the overlap of knowledge amongst these systems leads to increased knowledge management costs (e.g., consistency and configuration management). A knowledge repository capable of feeding multiple knowledge-based systems is one solution. However, it is critical that knowledge in the repository be stored in a use-neutral format and then transformed into representations that are tailored for specific uses (e.g., design). This explicit decomposition of knowledge into a use-neutral corpus, knowledge transformations on that corpus, and use-specific knowledge that is not shared by multiple systems, is useful not only in maintaining existing knowledge-based systems, but also in generating initial versions of entirely new systems.

We have developed a use-neutral representation of an hydraulic system for the F-111 airplane. We demonstrated the ability to derive portions of four different knowledge bases from this use-neutral representation: one knowledge base is for re-design of the device using a model-based reasoning problem

solver; two knowledge bases, at different levels of abstraction, are for diagnosis using a model-based reasoning problem solver; and one knowledge base is for diagnosis using an associational reasoning problem solver. The use of our technique highlighted inconsistencies which were not noticed before. The associational rules derived were more precise than the rules derived for the same scenarios by the person who created the model-base reasoning model. We have shown how updates issued against the single source use-neutral knowledge repository can be propagated to the underlying knowledge bases. We have also shown a plausible methodology for generating the knowledge repository given a set of pre-existing knowledge bases.

References

[1] J. Van Baalen. Automated Design of Specialized Representations. In *Proceedings of the Change of Representation and Problem Reformulation Workshop, Price Waterhouse*, 1990.

[2] R. Davis. Diagnostic Reasoning Based on Structure and Behavior. *Artificial Intelligence*, 24, 1984.

[3] N.S. Flann. Learning Appropriate Abstractions for Planning in Formation Problems. In *Proceedings of the Sixth International Conference on Machine Learning, Ithaca, New York*, pages 235–239. Morgan Kaufmann, 1989.

[4] T. Gruber and U. Iwasaki. How things work: Knowledge-based modeling of physical devices.

[5] R.M. Keller. Model Compilation: An Approach to Automated Model Derivation. In *Proceedings of the Change of Representation and Problem Reformulation Workshop, Price Waterhouse*, 1990.

[6] C.A. Knoblock. Learning Problem-Specific Abstraction Hiearchies. In *Proceedings of the Change of Representation and Problem Reformulation Workshop, Price Waterhouse*, 1990.

[7] D. Lenat and R.V. Guha. The world according to cyc. Technical Report ACA-AI-300-88, MCC, 1988.

[8] M.R. Lowry. Category Theory and Homomorphic Abstraction. In *Proceedings of the Change of Representation and Problem Reformulation Workshop, Price Waterhouse*, 1990.

[9] P.J. Riddle. Automating Shifts of Representation. In P. Benjamin, editor, *Change of Representation and Inductive Bias.* Kluwer, 1989.

[10] P. Rothman. Knowledge Transformation. *AI Expert*, 1988.

[11] R.A. Stachowitz and J.B. Combs. Validation of Expert Systems. In *Proceedings of the Twentieth Hawaii International Conference on System Sciences, Kona, Hawaii*, January 1987.

[12] D. Subramanian. A Theory of Justified Reformulations. In *Proceedings of the Change of Representation and Problem Reformulation Workshop, Price Waterhouse*, 1990.

[13] B. Thuraisingham. From Rules to Frames and Frames to Rules. *AI Expert*, 1989.

[14] A. Unruh and P.S. Rosenbloom. Abstraction in Problem Solving and Learning. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Detroit, Michigan*, pages 681–687. IJCAI, Morgan Kaufmann, August 1989.

[15] D.S. Weld. Discrepancy Driven Selection of Approximation Reformulations. In *Proceedings of the Change of Representation and Problem Reformulation Workshop, Price Waterhouse*, 1990.

[16] B.C. Williams. Interaction-based invention: Designing novel devices from first principles. In *Proceedings of the Eighth National COnfernce on Artificial Intelligence*, 1990.

☆U.S. GOVERNMENT PRINTING OFFICE: 1991-527-06/26017