

# KB-PHaSE: A Knowledge-Based Training Tool for a Space Experiment\*

Peter Clark, John Thompson                      Mary Lynne Dittmar  
Applied Research and Technology,    Int. Space Station Ops and Utilization, ISDS,  
Boeing, PO Box 3707, Seattle, WA    Boeing, 2100 Space Pk Blvd, Nassau Bay, TX

December 1998

## Abstract

This document summarizes a short (3 month) project exploring the use of knowledge-based training tools for Space Station payload experiments, and develop a demonstration prototype (called KB-PHaSE). The purpose of the knowledge-base is to augment existing training material with a ‘dynamic component’, whereby the user can ask questions during a training session and receive customized, situation-specific answers, generated automatically from the knowledge base. We describe the application domain, the system from the user’s point of view, and the structure of the knowledge-base underlying the application. Finally, we reflect on the benefits and challenges of this approach.

## 1 Introduction

This document summarizes a short (3 month) project exploring the use of knowledge-based training tools for Space Station payload experiments, and develop a demonstration prototype (called KB-PHaSE). Knowledge-base technology involves constructing a structured representation (the *knowledge-base*, or KB) describing the application domain, and using that representation to provide question-answering services to the user. In the role of training, the knowledge-base adds an ‘active’ component to on-line training material: in addition to browsing documentation, users can now also ask questions and have situation-specific answers generated by the KB at run-time. We view the KB’s role as one of augmenting (rather than replacing) existing training tools.

In this project, we explored the use of a knowledge base to both answer users’ questions at run-time, and perform interactive simulation of the experiment for training purposes. The potential benefits the knowledge-base can provide are as follows:

**Robustness:** By inferring answers at run-time, rather than reciting a pre-written paragraph of text, the system can respond to questions unanticipated at the time of system construction.

**Customization:** As answers are synthesized by the knowledge-base, answers can be customized to the user’s particular situation, for example answering questions such

---

\*This work was funded as part of ISS Operations and Utilization IR&D, Training Systems Integration (Boeing, Houston)

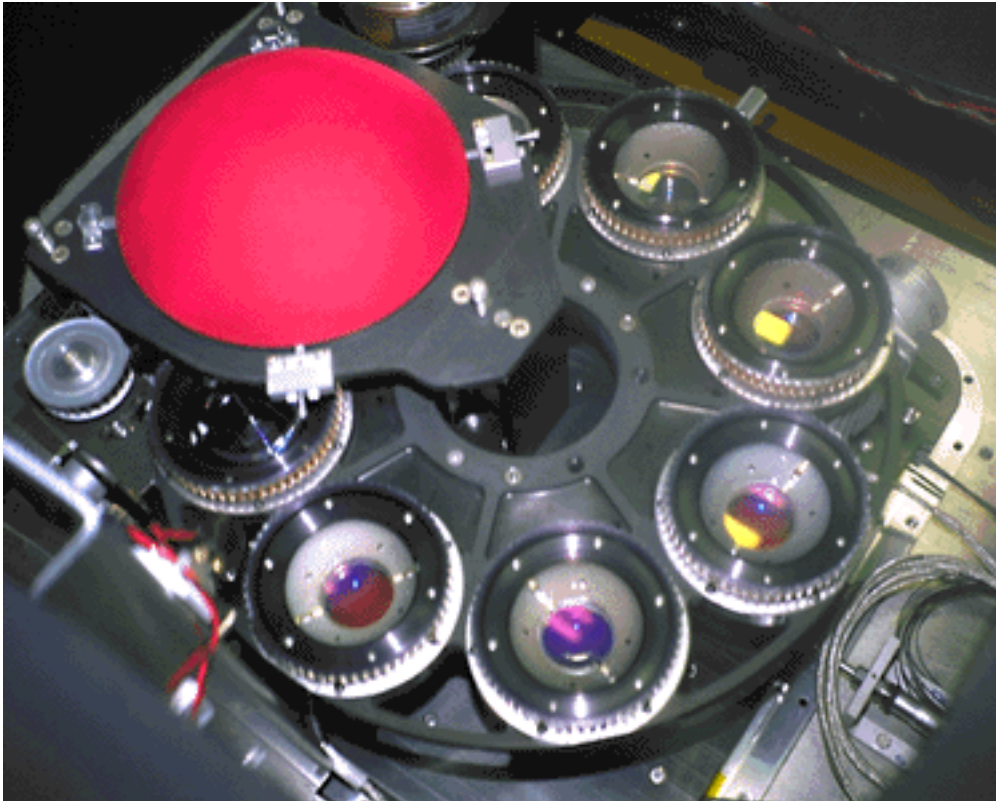


Figure 1: The rotating carousel in PHaSE, containing sample suspensions of tiny spheres for investigation.

as “What is the current state of the motor?”, “Why did light 3 come on?”, “What should I do next?”.

**Controllable level of detail:** Similarly, the level of detail can be dynamically controlled to suit the user’s level of expertise, by controlling how much information from the knowledge-base is included in the answer. Although this was not explored in this project, this has been illustrated elsewhere (eg. [1]).

**Adaptability to users’ preferred learning style:** By integrating instructional material, documentation on the science underlying the experiment, and knowledge-based question-answering, the user has multiple ways of exploring the material to suit his/her own learning style.

During this project, we constructed a knowledge-base about a particular science experiment (the PHaSE experiment, described shortly), and provided a Web-based interactive interface that allowed the user to explore the knowledge, ask questions, and try simulations. Example experimental faults could be introduced during the simulation, so that the user could then learn/practice recovery procedures from such faults.

In the report, we first outline the PHaSE experiment, then describe the system from the users’ point-of-view, illustrating how the knowledge-base and on-line documentation combine for training. We then describe the underlying structure of the knowledge-base, and some examples of what was encoded. Finally we offer some reflections on the merits and limitations of this approach.

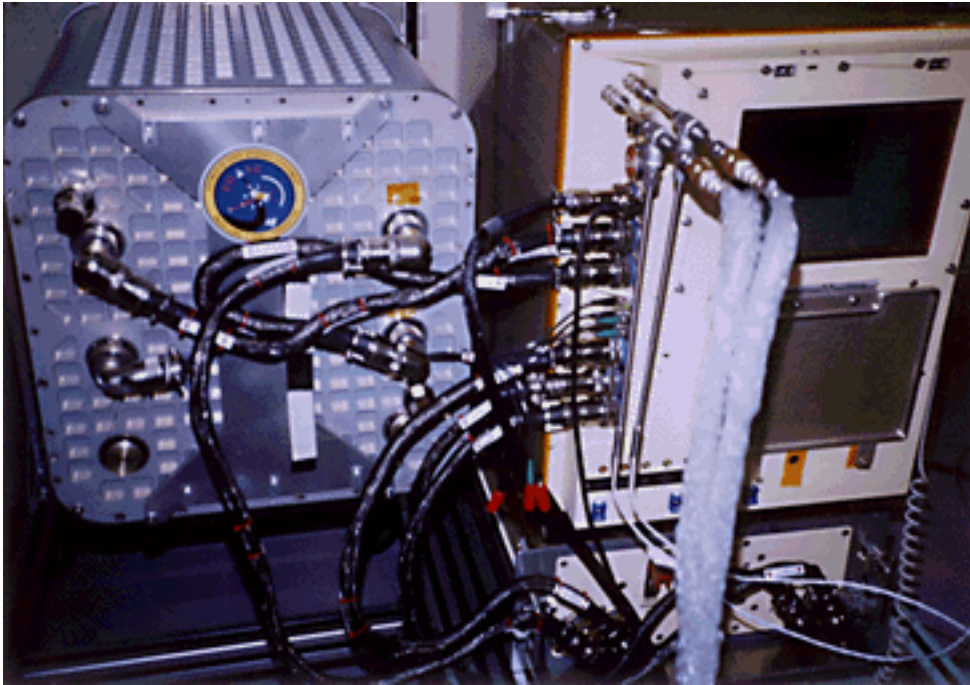


Figure 2: The flight hardware modules holding the PHaSE experiment. On the left is the test section, on the right are the avionics and power sections.

## 2 Description of Domain

The domain of this training system was the Physics of Hard Spheres Experiment (PHaSE) planned for the International Space Station, and already tested on a Space Shuttle mission. The experiment involves projecting a laser beam through various colloidal suspensions of tiny spheres in liquids to study the transitions among solid, liquid, and glass states in micro-gravity. Different suspensions are stored in a carousel, which is rotated to place one such sample under the laser beam in turn (Figure 1). The samples are oscillated and a camera records the laser image, which is then transmitted to the main experiment computer. The apparatus itself consists of three hardware assemblies: the test section (holding the carousel and camera), the avionics section (for data acquisition and control), and the power section (for power control) (Figure 2). An astronaut runs the experiment by powering up the electronic drawers that hold the apparatus and loading various script files as needed for special cases. Further information on this experiment is available on the Web at <http://zeta.lerc.nasa.gov/phase/>.

The astronaut may be satisfied to run the experiment by following the checklists that are provided, or he or she may also want to learn and understand how the experiment works and what its purpose is. The automated training system built for this project allows for both types of training: rote learning by simulating the checklist actions, or exploratory learning by browsing the knowledge about the experiment's hardware design and purpose.

### 3 User Interface Design

We first describe KB-PHaSE from the trainee’s point-of view. The Web-based interface that the user (astronaut) sees is divided into a left-hand window and a right-hand window, as illustrated in Figure 3. The left-hand window is dedicated to showing one page at a time from the pre-existing training manual, for example showing the checklist for activating the PHaSE system, or a description of the science experiment and all its hardware components. The user can navigate to any page by clicking the mouse on the page-turning arrows or can go directly to any page from the table of contents page.

The right-hand window is used for two purposes. One purpose is to show a simplified control panel that the user can operate as if it were the actual PHaSE hardware. The panel shows the current state of each device, switch, and indicator light via text and icons. The panel page also provides a menu of possible actions that the user can select and then watch the state of the panel indicators change. This is illustrated in Figure 3.

When the control panel is not being displayed, the right-hand window is available to show one page at a time for each concept in the knowledge base. The prototype system automatically creates one Web page for each and every concept (frame) in the knowledge base, giving its name, definition, superclasses, and a menu of questions relevant to that type of thing. For example, for physical devices the user can ask for the parts of the device and for the possible malfunctions that could occur with it. If the user clicks on one of these questions, the system infers the answer using the knowledge base and displays the answer in English on a revised version of the same web page. Examples of this are shown in Figures 4 and 5, and will be discussed in more detail shortly.

Each concept in the knowledge base has an associated set of English words or phrases that, if they occur in the displayed text, are hyperlinked to this concept. The prototype system automatically reads and hyperlinks all Web pages in the display, so the user can click on any term of interest and see the knowledge base page that explains that concept. For example, the user could click on any unfamiliar word or phrase in the left-hand page that describes the experiment’s hardware, and immediately see the knowledge base page on the right that defines that concept. From there, the user can ask questions about that concept, or follow hyperlinks to related concepts.

### 4 Example Usage

To demonstrate the features of this system, we will walk through a possible training scenario. Suppose that an astronaut sits down at a computer screen to learn how to perform the PHaSE experiment. The left-hand screen shows the cover page of the existing training manual and he clicks to turn to the table of contents.

He first selects the PHaSE Hardware Description page to learn the purpose of the experiment and the hardware involved. As he reads the description of the apparatus, he becomes curious about what the “rotary carousel” looks like and clicks on that phrase. The right-hand window brings up a page titled “PHaSE Rotary Carousel” which includes a description of the carousel and a menu of questions he can ask. He selects “show me a picture” and a color image of the carousel appears below the question. He also selects “what are its parts” and sees that a group of eight Sample-Cells is listed, and he clicks on those words to learn more about the sample cells on another page in the right-hand window.

After browsing the Hardware Description and becoming familiar with that, he goes

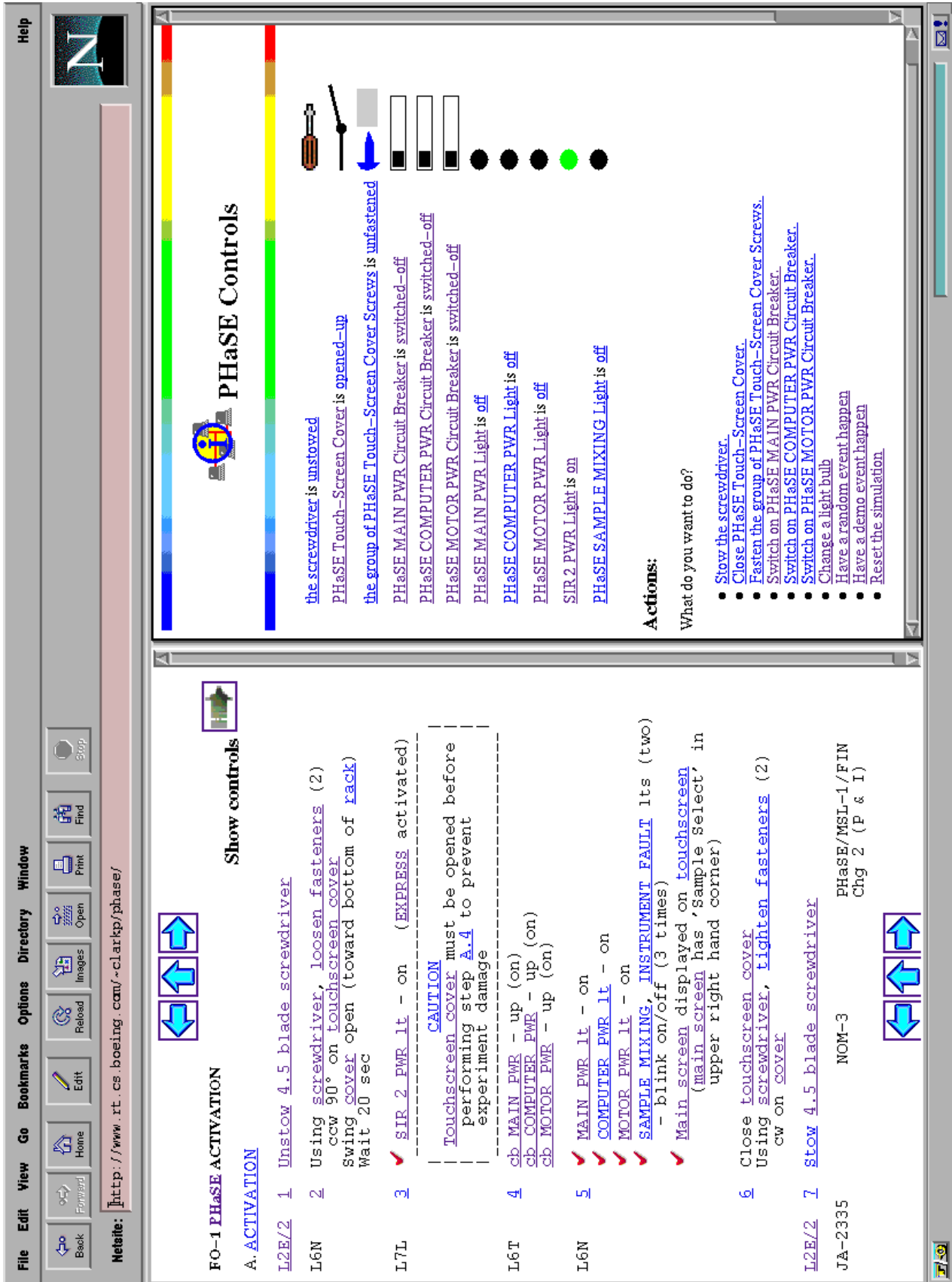


Figure 3: KB-PHaSE user interface. The left-hand window shows the pre-written checklist actions, the right-hand window is a KB-generated description of the current experimental state. Highlighted concepts can be clicked on for information about them.

File Edit View Go Bookmarks Options Directory Window Help

http://www.rt.cs.boeing.com/~clarke/phase/

**FO-1 PHaSE ACTIVATION**

**Show controls**

A. ACTIVATION

L2E/2 1 Unstow 4.5 blade screwdriver

L6N 2 Using screwdriver, loosen fasteners (2)  
ccw 90° on touchscreen cover  
Swing cover open (toward bottom of rack)  
Wait 20 sec

L7L 3 ✓ SIR\_2\_PWR\_lt - on (EXPRESS activated)

-----  
**CAUTION**  
 Touchscreen cover must be opened before performing step A.4 to prevent experiment damage  
 -----

L6T 4 cb MAIN\_PWR - up (on)  
cb COMPUTER\_PWR - up (on)  
cb MOTOR\_PWR - up (on)

L6N 5 ✓ MAIN\_PWR\_lt - on  
✓ COMPUTER\_PWR\_lt - on  
✓ MOTOR\_PWR\_lt - on  
✓ SAMPLE\_MIXING\_INSTRUMENT\_FAULT\_lts (two)  
- blink on/off (3 times)  
✓ Main\_screen displayed on touchscreen  
(main screen has 'sample select' in upper right hand corner)

6 Close touchscreen cover  
Using screwdriver, tighten fasteners (2)  
cw on cover

L2E/2 7 Stow 4.5 blade screwdriver

JA-2335 NOM-3 PHaSE/MSL-1/FIN  
Chg 2 (P & I)

http://thumper.rt.cs.boeing.com/PEC-bin/webblink?frame=Universal-Small-Experiment-Container

---

**BOEING**

**Topic: PHaSE MOTOR PWR Light**

**Definition:**  
A light that shines to show that the USEC (Universal Small Experiment Container) box in the PHaSE equipment in the Space Station's EXPRESS rack is receiving electrical power.

**Generalizations:**

- [Power-On Light](#)

**Specializations:**

**Questions:**

- [Purpose of a PHaSE MOTOR PWR Light](#)
- [Possible malfunctions related to a PHaSE MOTOR PWR Light](#)
  - The bulb may have no power. To confirm: Check to see if other bulbs have power.
  - The bulb may be burned out. To confirm: Change the bulb.
  - The PHaSE motor may be tripped. To confirm: Toggle the PHaSE main power.
- [Show me picture of a PHaSE MOTOR PWR Light](#)
- [Parts of a PHaSE MOTOR PWR Light](#)

**References:**

Figure 4: KB-PHaSE user interface. In response to the MOTOR PWR light failing to go on, the user has asked about this object, and then clicked on the malfunctions question to find possible causes of failure. The knowledge base has responded with a list of possible problems and corrective actions to pursue.

to the page in the left-hand window that shows the checklist for the PHaSE Activation procedure. He is ready to practice this checklist, so he clicks on the “show controls” button and sees a simplified control panel appear in the right-hand window (similar to Figure 3, only with the controls in their initialized state). The various switches and indicator lights are at the top of the right-hand window, and below is a menu of possible actions that can be taken from this initial state of the system, such as unstow the screwdriver, or switch on the MAIN PWR circuit breaker, or change a light bulb.

The astronaut reads step 1 of the checklist and clicks on the action to unstow the screwdriver. The control panel, which formerly said “the screwdriver is stowed” and showed an icon of a toolbox, now changes to say “the screwdriver is unstowed” and shows an icon of a screwdriver. The menu of possible actions also changes to include “unfasten the touch-screen screws” as a new possibility. Behind this interface, the knowledge-base is performing a simple simulation of the user’s chosen action, to determine the new state of the experiment and what new actions are available to the user.

The astronaut continues down the checklist, clicking on the correct actions to unfasten the screws, open the touch-screen cover, and check that the SIR 2 PWR light is on. (At this stage the controls appear exactly as in Figure 3). He then turns on the three circuit-breakers (which turn on indicator lights). In addition, at any point the astronaut can select the action “cause a random event to occur,” which might cause one of the lights to burn out, or cause the experiment’s motor to turn off (for example)<sup>1</sup>. Let us consider the astronaut now clicks on this action in this scenario.

Now the astronaut, continuing with the checklist, turns on the MOTOR PWR circuit breaker. However, its indicator light fails to come on. This is an unexpected condition, from which he now has to decide what to do. He first decides to find more information about this indicator light, so clicks on the highlighted phrase “MOTOR PWR lt” (light) in the checklist; in response, the knowledge base generates a page describing that light in the right-hand window. He then clicks on the “possible malfunctions” question, to find possible reasons the light may not be operating correctly, and gets a list of three malfunctions and their corrective actions, as illustrated in Figure 4.

The possible problems and corrective actions listed are as follows:

- The bulb may have no power. To confirm: Check to see if other bulbs have power.
- The bulb may be burned out. To confirm: Change the bulb.
- The PHaSE motor may be tripped. To confirm: Toggle the PHaSE main power.

The astronaut goes back to the control panel and tries each of these actions in turn. The other bulbs are lit (ruling out the first candidate malfunction); he tries changing the MOTOR PWR light by clicking on the “change light” action, but the light still remains off (ruling out the second candidate malfunction); finally he toggles the MAIN PWR switch off and on, which does correct the problem<sup>2</sup>.

The astronaut has the option of resetting the system at any time to practice the checklists as many times as he likes, with as many random events added as he chooses

---

<sup>1</sup>In this prototype, the timing of an unexpected event (eg. a malfunction) is explicitly controlled by the user; in a more sophisticated version, in which the training system tracked the user’s performance and knowledge, the system itself would control when (as well as which) such events occur.

<sup>2</sup>This scenario and behavior are not actually found in the training materials we have; we created it as illustrative of the behavior we could simulate if we had more knowledge of the system components and their malfunctions

to add. He is also free to browse the knowledge base of concepts for definitions, images, and question-answering anytime he likes.

In addition to following a simulation, the astronaut can also browse the knowledge base to explore various concepts related to the experiment, ask questions about these concepts, and from here browse additional documentation about them. Figure 5 shows the state of the screen after the astronaut has asked about the Test Section, and then clicked on the Reference cited in the right-hand page. As a result, the left-hand page changes to the on-line documentation describing the test section. In this way, the user is able to browse information about the experiment in general, and explore the underlying science if he/she is interested, in addition to (or interleaved with) following a directed checklist of actions. Figure 8 shows the state of the screen after the astronaut has asked for more details about Step 2 in the checklist, to find out the reason why he is being asked to carry out the task, and for the list of equipment he needs. In all cases, answers in the right-hand window have been synthesized at run-time from the knowledge-base, based on the encoded knowledge about the objects/concepts being described.

## 5 Software Architecture

We now describe the underlying software and knowledge-base used in KB-PHaSE. The prototype system consists of three main architectural components:

1. a user interface based on Web pages,
2. a module of question-answering procedures and a page generator, and
3. a knowledge base with an inference engine.

This is illustrated in Figure 6.

The question-answering module handles all mouse clicks on hyperlinked words and phrases. If the selection is a simple hyperlink to another page, that page is generated (if necessary) by a page generation routine, and displayed in the proper window. If the selection is a question or action on a menu, this module processes the question or action and passes the request on to the knowledge base module, gets back an answer or result, and generates a revised page that contains the answer or includes the result of the action. This module is written in Lisp.

The knowledge base module contains all the declarative knowledge describing the PHaSE experiment, and more general concepts (eg. light, carousel, rotation) related to the experiment. The KB is built using a frame-based representation language called KM. The inference engine for the KM language accepts queries from the question-answering module, computes an answer via automated deductions in the knowledge base, and returns the result to the question-answering module. Sometimes the answer is an English sentence that is generated in the knowledge base by filling in simple sentence templates with the proper concepts.

## 6 Knowledge Base Architecture

The knowledge base itself was written using a Lisp-based frame language called KM (the Knowledge Machine) from the University of Texas at Austin [2]. Constructing the knowledge base was the main task required for this prototype (the software for the Web interface was already available).

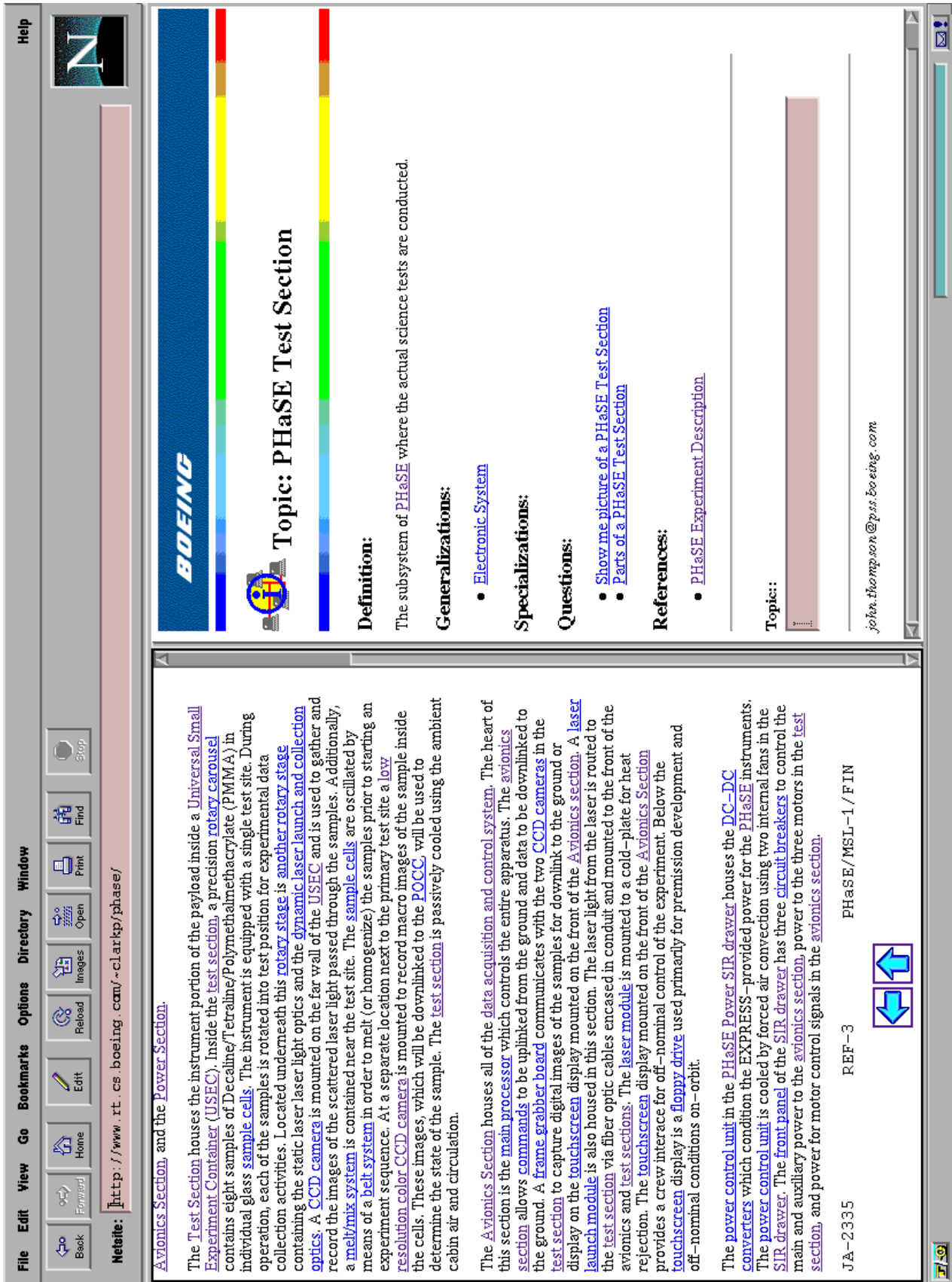


Figure 5: KB-PHaSE user interface. Here the user is browsing the knowledge-base in a more open-ended fashion, and has clicked to find about the Test Section (right-hand window). He/she has then clicked on the References cited, to call up on-line documentation describing the test section (left-hand window).

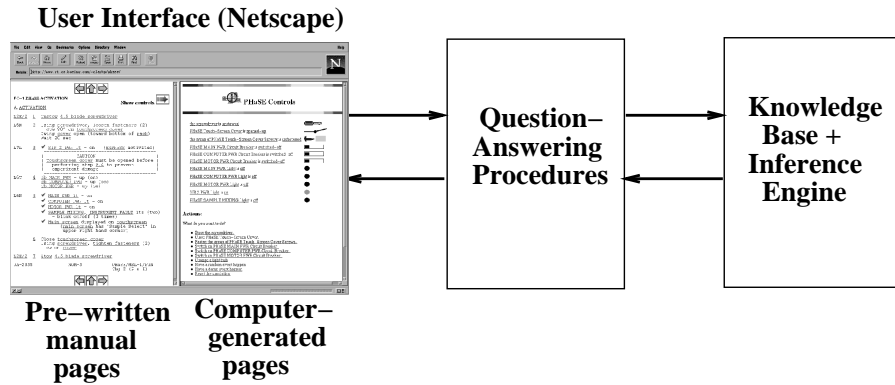


Figure 6: Basic Architecture of the KB-PHaSE trainer.

Although the knowledge-base is a single entity, it is useful to view the knowledge-base as comprising three parts:

- The **ontology** (‘isa hierarchy’), enumerating the concepts in the domain, and their generalization/specialization relationships, and some basic properties of those concepts. This provides a core vocabulary for representing the knowledge about the PHaSE experiment.
- A **domain description**, containing
  - a structured representation of PHaSE experimental setup, and the checklist tasks which the astronaut is to perform.
  - more general, ‘common sense’ knowledge related to that experiment. For example, the KB knows that an activity needs all the tools required for its subactivities.
- A collection of **mini-theories**, providing axioms (rules) for reasoning with the concepts in the ontology. For example, one such theory is about electrical circuits, enabling KM to reason about circuits and draw conclusions (eg. whether a light bulb is on or off).

We summarize each of these three parts in turn.

## 6.1 The Ontology

For this application, the ontology was divided into three layers:

**Upper level** General-purpose concepts that would apply to many application domains, ie. are not domain-specific. Examples of such concepts are **Physical-Object** and **Activity**.

**Mid-level** Concepts which are specific to the domain of space, but not PHaSE-specific. Examples include **Astronaut** and **Electronic-Rack**.

**Application-specific** Concepts that are specifically about the PHaSE experiment, such as the **PHaSE-Avionics-Section** and the **PHaSE-Main-Power-Light**.

The purpose of this separation is to modularize the ontology into reusable parts: The upper level ontology should be portable to other applications (indeed, it was largely

imported from a previous application in Boeing’s Factory Education project). Similarly, the mid-level should be portable to other space applications, while the application-specific layer is targeted just to the particular PHaSE task. The choice of which level to place a concept in is largely pragmatic, based on the expected reuse of the concept.

The complete ontology is shown in Appendix A.

## 6.2 The Domain Description

In addition to providing the basic concepts, it is necessary to provide a description of the PHaSE experiment, and the checklist actions to be modeled, using the vocabulary which the ontology provides. In addition, more general, ‘common sense’ knowledge is required about more general concepts in the ontology. In this prototype, we concentrated on representing the checklist actions and associated concepts, and also provided a simple representation of the PHaSE electrical circuit for simulating the experiments’ behavior.

The checklists themselves are subclasses of the concept **Written-Specification**. One checklist was the focus of this prototype: the seven-step checklist for activating (powering-on) the PHaSE system, shown in the left-hand window of Figure 3. Each step was modeled in terms of the kinds of actions involved, such as **Removing-From-Storage**, **Loosening-A-Screw**, **Opening-A-Hinged-Cover**, **Waiting**, **Checking**, and **Throwing-A-Switch**.

For example, step 6 of the Activation checklist in Figure 3 contains these actions:

1. Close touchscreen cover
2. Using screwdriver, tighten fasteners (2) cw on cover

This is represented in the knowledge base as illustrated in Figure 7, by a ‘frame’ (ie. a knowledge-base unit) for step 6 and two more frames for substeps 6a and 6b. Sub-step 6a includes an **action** slot that contains a **Closing-A-Hinged-Cover** action with an **Astronaut** as the **agent** and the **PHaSE-Touch-Screen-Cover** as the **object**. Sub-step 6b has an **action** slot that contains a **Tightening-A-Screw** action in which the **agent** is an **Astronaut**, the **instrument** is the tool unfastened in step 1, namely a **Screwdriver-With-4.5-Blade**, the **object** is the **PHaSE-Touch-Screen-Cover-Screws** (a group of two screws), and the **direction** is **Clockwise**. The KM representation itself is shown in Figure 7. Note the “path” used in step 6b, stating that the **instrument** for **Tightening-A-Screw** should be the same instrument used in Step 1 – at run-time, KM’s inference engine follows this path, to compute what this is (in this case, is a **Screwdriver**).

Given a structured representation in this form, KM is able to reason with it to answer questions such as “Which tools are used in step X?”, “When is the Screwdriver needed?”, “What is the purpose of step X?”, “What might go wrong in step X?”. The implementation is able to answer a number of questions of this kind, for selected concepts in the knowledge base. Figure 8 illustrates KB-generated answers to several questions, this time focusing on Step 2 of the checklist.

## 6.3 Mini-Theories

In order to reason about, for example, the behavior of the lights in the PHaSE apparatus, it is not sufficient to just know the PHaSE circuit diagram, or that a light is an artifact and an electrical component – some additional knowledge is also required to define what such statements mean. A small collection of mini-theories provide such knowledge, providing axiomatizations (rules) about electrical circuits, machines, two-state devices, etc. and

```

;;; "Step 6 has two substeps, 6a and 6b."
(PHaSE-Activation-Step-6 has
  (instance-of (PHaSE-Checklist-Step))
  (print-name ("PHaSE Activation Step A.6"))
  (location (PHaSE-Avionics-Section))
  (parent-step (PHaSE-Activation-Procedure))
  (subsections-in-order (
    PHaSE-Activation-Substep-6a
    PHaSE-Activation-Substep-6b)))

;;; "Close touchscreen cover."
(PHaSE-Activation-Substep-6a has
  (instance-of (PHaSE-Checklist-Substep))
  (print-name ("PHaSE Activation Step A.6 - 1st substep"))
  (parent-step (PHaSE-Activation-Step-6))
  (action ((a Closing-A-Hinged-Cover with
    (object (PHaSE-Touch-Screen-Cover))
    (agent ((a ISS-Astronaut)))))))

;;; "Using screwdriver, tighten fasteners (2) cw on cover."
(PHaSE-Activation-Substep-6b has
  (instance-of (PHaSE-Checklist-Substep))
  (print-name ("PHaSE Activation Step A.6 - 2nd substep"))
  (parent-step (PHaSE-Activation-Step-6))
  (tools-and-equipment ((the instrument of (the action of Self))))
  (action ((a Tightening-A-Screw with
    (instrument ((the object of          ; the tool used in Step 1 (a screwdriver)
      (the action of PHaSE-Activation-Step-1))))
    (object (PHaSE-Touch-Screen-Cover-Screws))
    (turning-direction (Clockwise))
    (agent ((a ISS-Astronaut)))))))

```

Figure 7: Knowledge-Base Representation of Step 6.

File Edit View Go Bookmarks Options Directory Window Help

Website: <http://www.rt.cs.boeing.com/~clarkep/phase/>

Back Forward Home Edit Open Print Find Stop

Images Reload

FO-1 PHaSE ACTIVATION

A. ACTIVATION

L2E/2 1 Unstow 4.5 blade screwdriver

L6N 2 Using screwdriver, loosen fasteners (2)  
 ccw 90° on touchscreen cover  
 Swing cover open (toward bottom of rack)  
 Wait 20 sec

L7L 3 ✓ SIR\_2\_PWR\_lt - on (EXPRESS activated)

CAUTION  
 Touchscreen cover must be opened before performing step A.4 to prevent experiment damage

L6T 4 cb MAIN\_PWR - up (on)  
cb COMPUTER\_PWR - up (on)  
cb MOTOR\_PWR - up (on)

L6N 5 ✓ MAIN\_PWR\_lt - on  
 ✓ COMPUTER\_PWR\_lt - on  
 ✓ MOTOR\_PWR\_lt - on  
 ✓ SAMPLE MIXING, INSTRUMENT FAULT lts (two)  
 - blink on/off (3 times)  
 ✓ Main screen displayed on touchscreen  
 (main screen has 'sample select' in upper right hand corner)

6 Close touchscreen cover  
 Using screwdriver, tighten fasteners (2)  
 cw on cover

L2E/2 7 Stow 4.5 blade screwdriver

JA-2335 NOM-3 PHaSE/MSL-1 /FIN  
 Chg 2 ( P & I)

Show controls

Topic: PHaSE Activation Step A.2

Definition:  
 Step A.2 in the PHaSE Activation checklist FO-1.

Generalizations:  
 • PHaSE Checklist Step

Specializations:

Questions:  
 • What is the wording of PHaSE Activation Step A.2  
 • Describe the process of doing a PHaSE Activation Step A.2  
 • First, Loosen the group of PHaSE Touch-Screen Cover Screws counter-clockwise using the Screwdriver With 4.5 Blade.  
 • Then, Open PHaSE Touch-Screen Cover by rotating it on its hinges.  
 • Finally, Wait for 20 seconds.  
 • What are the substeps for doing PHaSE Activation Step A.2  
 • What is the goal in doing a PHaSE Activation Step A.2  
 • The Touch-Screen Cover is open and the PHaSE System is no longer vulnerable to a power surge from opening the Touch-Screen Cover.  
 • Tools and equipment used in doing a PHaSE Activation Step A.2  
 • the Screwdriver With 4.5 Blade  
 • Location for doing a PHaSE Activation Step A.2  
 • Some things that might go wrong during a PHaSE Activation Step A.2  
 • Some things that should be avoided while doing a PHaSE Activation Step A.2  
 • Purpose of a PHaSE Activation Step A.2  
 • Possible malfunctions related to a PHaSE Activation Step A.2  
 • Show me picture of a PHaSE Activation Step A.2

Figure 8: KB-PHaSE user interface. Here the user is browsing information about Step 2 in the checklist, and has asked about the process involved, the goals, and the equipment needed. The answers shown are synthesized at run-time by KB-PHaSE from the knowledge-base.

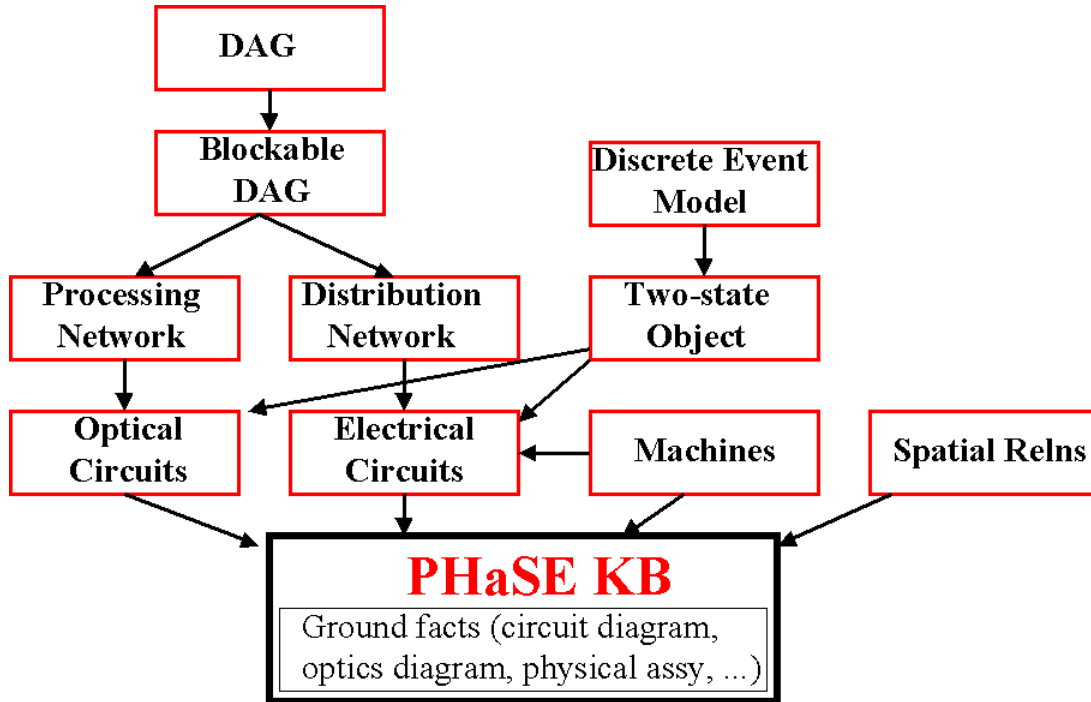


Figure 9: The component “mini-theories” used in KB-PHaSE. Each box denotes a theory (set of rules) describing a phenomenon. Arcs denote inclusion relations. Some of the theories themselves are described in Appendix B.

how they behave. For example, the circuits mini-theory includes rules such as: if an electrical component is connected to a power supply, and there are no breaks in that connection (eg. no switch is open), then that component will receive electrical power. The machines mini-theory includes rules such as: machines can be either working or broken, and if they are broken then they will be unable to carry out their primary function.

These mini-theories are small but intricate collections of axioms, again represented in KM. They are modularized so that new theories can be defined in terms of old theories, and to enable reuse of theories. For example, the “electrical circuit” theory is defined as a specialization of the “distribution network” theory. Examples of some of these mini-theories and their relationship are shown in Appendix B. The theories used in KB-PHaSE are illustrated in Figure 9.

## 7 Overview of The KB Content

Finally, we briefly outline the knowledge itself contained in the KB. The ontology itself is relatively complete for the PHaSE experiment, describing most of the main concepts involved (approximately 300 concepts). This is shown in Appendix A.

The domain description for the PHaSE concepts includes a text definition, information about part-subpart relationships, lexical information (print-names of concepts), and some additional structural information (eg. that the `Touch-Screen-Cover` is secured by two screws). The domain description for the checklist steps is the most completely filled-out part of the knowledge base, including information about a step’s location, title, the action(s) involved (including the agent, object, instrument etc. required), possible failure

modes, and the goal of the step. In addition, connectivity of the electrical components is recorded.

The mini-theories provide knowledge for reasoning about circuits, and for reasoning about dynamic systems based on a discrete event model. Various objects are modeled as two-state objects: A **Tool** can be **Stowed** or **Unstowed**; A folding cover can be **Open** or **Closed**; A **Light** can be **Switched-On** or **Switched-Off**, etc. Associated actions (**Opening**, **Closing**, **Stowing**, etc.) cause the states of these objects to change. A mini-theory about machines states that a **Machine** can be **Broken** or **Not-Broken**. These theories interact: A **Light-Bulb**, which is modeled as a **Machine**, a **Two-State-Device**, and an **Electrical-Component**, acquires properties and behaviors from three mini-theories.

## 8 Related Work

There are two areas of work closely related to this project: work on intelligent tutoring systems (ITS), and work on knowledge-based systems (KBS). We provide some introductory pointers to related work in both the areas below.

### 8.1 Intelligent Tutoring Systems

As an approximation, it is useful to view ITSs in two main categories: those that use a *simulation approach*, and those that use a *network navigation* approach. The simulation approach is characterized by the tutoring component being linked to an interactive simulator of the artifact the student is learning about, with which the student can perform various actions, observe changes over time, experiment, etc. In the simulation approach, the student is presented with training scenarios that require him or her to respond correctly to each step of a scenario. The system may generate random scenarios, or there may be a set of predefined scenarios that are run. The system monitors the actions of the trainee, compares them to an “expert model” (the correct responses, or the rules of correct behavior), and keeps a simple “student model” of the trainee’s score for each task. The simulation approach is typically used when the trainee needs to learn to perform a complex process through repeated practice sessions, as opposed to learning some intellectual material from a textbook. Some example ITSs using the simulation approach are RIDES/VIVIDS [3], LEAP [4], and ICAT *icat*.

In the network navigation approach, the system presents the trainee with a series of pre-built training pages to view. It uses its “teaching strategy” to decide which training unit to present next, based on the user’s responses to the previous unit. Each separate unit may be a conventional CBT (Computer Based Training) page, possibly with multimedia included. The system’s intelligence is in deciding how to navigate the user through the network of readings, examples, exercises, hints, and examinations. Examples include the Eon system [5], COCA/REDEEM [6], and ISIS-Tutor [7]. In some cases the simulation and network navigation approaches are both included in an overall tutoring system, eg. RIDES/VIVIDS [3].

Our PHaSE prototype falls mainly into the simulation approach to ITS, due to its control panel simulation, but it also contains some elements not generally found in either approach, such as unconstrained exploration of a subject area and dynamic question-answering. Conversely, there are some common ITS functionalities which we have not yet included in the PHaSE prototype which would clearly be desirable: in particular, the PHaSE prototype does not have any explicit representation of a syllabus, teaching

strategies, nor a “student model” for recording the student’s progress through that syllabus. Addition of a more active and explicit teaching component, or integration of the PHaSE knowledge-base with an existing ITS shell supporting this functionality, would be an obvious next step in developing the system further.

## 8.2 Knowledge-Based Systems

As described earlier, the domain knowledge needed by the PHaSE prototype is encoded in a modular fashion, using a frame-based knowledge representation language (KM) [2]. KM is similar in style to other frame-based languages such as KRL [8] and LOOM [9], but was extended as part of this project to enable reasoning about *situations*, and the effects of actions on situations, using a situation calculus implementation (loosely similar in style to Reiter’s work [10, Chapter 5]). This mechanism is described in [11].

The use of a component-based architecture (“mini-theories”) aims to organize knowledge into reusable and maintainable modules (rather than create a gigantic, monolithic knowledge base). This approach is described more fully in [12], and can be viewed as applying ideas from compositional modeling (eg. [13]) to knowledge-based construction. Several other groups are also pursuing work in modular knowledge-base construction, including the Ontolingua ontology library project [14], ‘microtheories’ [15], and the TOVE project [16]. Modular, reusable ontologies is currently a high-profile topic in the ontology community, eg. see [17].

## 9 Conclusion and Lessons Learned

This short project provided an opportunity to explore the use of knowledge-based technology for training, and demonstrate some of the potential capabilities which it offers. Although we were not able to formally evaluate the system, it is useful to make some comments based on our experience.

The two most significant potential benefits which the knowledge-base provides are the ability to generate situation-specific answers at run-time, and to answer questions unanticipated at the time of system construction. We were able to demonstrate the former, primarily through the inclusion of a knowledge-based simulation component: the system can model and report the effects of a users’ action, and perform analysis of the current state. It would be straightforward to add additional situation-specific question-answering capabilities based on this infrastructure, for example answering the question “what should I do next?” or “why did that happen?”. Such situation-specific questions are difficult or impossible to answer if only pre-written paragraphs of text are available.

The ability to answer unanticipated questions is based on the idea that general knowledge in the KB (eg. that lights can burn out) can be applied to specific concepts (eg. the PHaSE SIR2 power light) at run-time. We were able to demonstrate this capability in a limited way, for specific questions, but to achieve this more fully requires that the knowledge-base be more complete, with substantially more knowledge about the general concepts in the ontology.

The main bottleneck in the project, as is well-known in the literature, is the task of building the knowledge-base in the first place. During the three-month duration of the project we encoded only part of the knowledge required to cover the application domain completely; more effort would have been needed for complete coverage. This barrier has been a stumbling-block for many knowledge-based projects; the ‘solution’,

which we pursued here as a research theme, is the construction of a knowledge-base from prefabricated, reusable knowledge components. The knowledge-base was deliberately designed to modularize the encoded knowledge, so as to provide potential for reusing the knowledge on future projects. This approach is still a research topic, but we are hopeful that it can significantly ease the knowledge modeling bottleneck in the future. Both the layering of the ontology, and the encapsulation of representational ‘mini theories’, are important steps in this direction.

A second, related issue is that constructing a knowledge-base requires a skilled knowledge engineer; we thus also see the need for powerful but easy-to-use authoring tools that might allow a subject matter expert (who is not also a knowledge engineer) to enter at least some of the required knowledge via a simple interface. In fact, the knowledge-base itself is (to an approximation) a mixture of small, complex theories (eg. for modeling electrical circuits) and simple databases of facts (eg. listing the connectivity of components in the PHaSE electrical circuit). It seems likely that the latter could be easily entered by a subject matter expert, again easing the knowledge modeling bottleneck.

Finally, there are several possible extensions to this framework that would be of value. As discussed earlier, KB-PHaSE does not have any explicit tutoring component for following a curriculum, tracking student performance, and providing testing/evaluation functions; such a component would be valuable for migrating KB-PHaSE from an “active manual” to a full tutoring system. The model of the PHaSE experiment itself is fairly simple at present; a more sophisticated model would be beneficial, and perhaps it might even be possible to link an existing simulator into the knowledge-base. Finally, the prototype uses only a few pages of documentation; proper technology for organizing and indexing into larger volumes of documentation would be desirable for a full-scale system, perhaps using the knowledge-base’s ontology as an indexing mechanism. These are all future developments with high potential value. Integrating knowledge-based question-answering with a tool which already performs these additional tutoring functions (Section 8.1) would be a promising direction to proceed.

## Acknowledgements

We are greatly indebted to Kim Krome, Marshall Space Flight Center, for her invaluable help answering questions about the PHaSE experiment. It should be noted, however, that due to the short nature of this project, our representation of PHaSE experiment is still very approximate – any interpolations, errors, and improvisations we have made are solely our responsibility. Thanks also to the Advanced Computing Group, Boeing Huntsville, AL, for technical support and liasoning, in particular George Williams.

## References

- [1] James C. Lester and Bruce W. Porter. Developing and empirically evaluating robust explanation generators: The knight experiments. *Computational Linguistics*, 23(1):65–101, 1997.
- [2] Peter Clark and Bruce Porter. KM – the knowledge machine: Users manual. Technical report, AI Lab, Univ Texas at Austin, 1998. (<http://www.cs.utexas.edu/users/mfkb/manuals/userman.ps>).

- [3] Allen Munro, D. Durmon, Q. Pizzini, and M. Johnson. Collaborative authored simulation-centered tutor components. In *Intelligent Tutoring System Authoring Tools (AAAI Fall Symposium Technical Report FS-97-01)*. AAAI, CA, 1997. (related papers at <http://btl.usc.edu/rides/>).
- [4] Scott Dooley, L. Meiskey, R. Blumenthal, and R. Sparks. Developing usable ITS shells. In Nigel Major, Tom Murray, and Charles Bloom, editors, *AI-ED-95 Workshop on Authoring Shells for Intelligent Tutoring Systems*. AACE, VA, 1995. (<http://www.pitt.edu/~al/aied/workshop.html>).
- [5] Tom Murray. From story boards to KB's – first paradigm shift in making CAI intelligent. In Nigel Major, Tom Murray, and Charles Bloom, editors, *AI-ED-95 Workshop on Authoring Shells for Intelligent Tutoring Systems*. AACE, VA, 1995. (<http://www.pitt.edu/~al/aied/workshop.html>).
- [6] Nigel Major and S. Ainsworth. Developing intelligent tutoring systems using a psychologically motivated authoring environment. In *Intelligent Tutoring System Authoring Tools (AAAI Fall Symposium Technical Report FS-97-01)*. AAAI, CA, 1997.
- [7] Peter Brusilovsky. Integrating hypermedia and intelligent tutoring technologies: From systems to authoring tools. In Nigel Major, Tom Murray, and Charles Bloom, editors, *AI-ED-95 Workshop on Authoring Shells for Intelligent Tutoring Systems*. AACE, VA, 1995. (<http://www.pitt.edu/~al/aied/workshop.html>).
- [8] D. Bobrow and T. Winograd. An overview of KRL, a knowledge representation language. In R. Brachman and H. Levesque, editors, *Readings in Knowledge Representation*, pages 264–285. Kaufmann, CA, 1985. (originally in *Cognitive Science* 1 (1), 1977, 3–46).
- [9] R. MacGregor and R. Bates. The LOOM knowledge representation language. Tech Report ISI-RS-87-188, ISI, CA, 1987.
- [10] Raymond Reiter. Knowledge in action: Logical foundations for describing and implementing dynamical systems. (<http://www.cs.toronto.edu/~cogrobo/>), 1999.
- [11] Peter Clark and Bruce Porter. KM – situations, simulations, and possible worlds. Technical report, AI Lab, Univ Texas at Austin, 1998. (<http://www.cs.utexas.edu/users/mfkb/manuals/situations.ps>).
- [12] Peter Clark and Bruce Porter. Building concept representations from reusable components. In *AAAI-97*, pages 369–376, CA, 1997. AAAI. (<http://www.cs.utexas.edu/users/pclark/papers/aaai97.ps>).
- [13] B. Falkenhainer and K. Forbus. Compositional modelling: Finding the right model for the job. *Artificial Intelligence*, 51:95–143, 1991.
- [14] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993. (<http://www-ksl.stanford.edu/knowledge-sharing/papers/README.html#ontolingua-intro>. Also see the project page at <http://www-ksl.stanford.edu/knowledge-sharing/ontolingua/index.html>).
- [15] Paul Blair, R. V. Guha, and Wanda Pratt. Microtheories: An ontological engineer's guide. Tech Rept CYC-050-92, MCC, Austin, TX, Mar 1992.

- [16] M. S. Fox and M. Gruninger. Ontologies for enterprise integration. In *Proc 2nd Conference on Cooperative Information Systems*, pages 82–90. Univ Toronto, 1994. (Also see ontologies at <http://www.ie.utoronto.ca/EIL/tove/toveont.html>).
- [17] Adam Farquhar and Mike Gruninger, editors. *Proc AAAI Spring Symposium on Ontological Engineering*. AAAI (in press), Mar 1997.

## Appendix A: The Ontology

The complete ontology used in PHaSE is as shown below, comprising approximately 300 concepts. The ... indicates a repeated occurrence of a concept in the taxonomy, thus omitting its details for second and subsequent times. (A concept will occur multiple times in the print-out if it has multiple parents). The prefix \* denotes an instance, rather than a class.

```
Thing
  Physical-Object
    Physical-System
      Electronic-System
        *PHaSE-Control-Panel
        *PHaSE-Power-Drawer-Front-Panel
        *PHaSE-Stepper-Servo-Motor-Power-Circuits
        *PHaSE-DC-DC-Converters-And-Filters
        *PHaSE-Power-Control-Unit
        *PHaSE-Laser-Light-Module
        *PHaSE-Data-Acquisition-And-Control-System
        *PHaSE-GRIN-Lens-System
        *PHaSE-Dynamic-Laser-System
        *PHaSE-Static-Laser-Launch-Optics
        *PHaSE-Bragg-Laser-System
        *PHaSE-Power-Drawer
        *PHaSE-Avionics-Section
        *PHaSE-Test-Section
        *PHaSE-System
        *EXPRESS-Solid-State-Power-Controller-Module
        *EXPRESS-Rack-Interface-Controller
      Electrical-Circuit
      Electronic-Communications-System
        *Operations-Communications-Adapter
      PHaSE-Avalanche-Photo-Diode
    Physical-Artifact
      Tool
        Screwdriver
          Screwdriver-With-4.5-Blade
            *PHaSE-Screwdriver-With-4.5-Blade
      Conduit
      Display-Device
        Projection-Screen
          *PHaSE-Fluorescent-Bragg-Screen
        Visual-Information-Display
          Computer-Screen-Display
            *PHaSE-Main-Screen
      Recording-Device
        Camera
          *PHaSE-Low-Resolution-Color-CCD-Camera
          *PHaSE-Color-CCD-Camera
      Piece-Of-Equipment
      Fastener
        *PHaSE-Touch-Screen-Cover-Screws
      Temporary-Fastener
      Screw
```

- PHaSE-Touch-Screen-Cover-Screw
- Gate
  - Hinged-Cover
    - \*PHaSE-Touch-Screen-Cover
  - Fastenable-Cover
    - \*PHaSE-Touch-Screen-Cover
- Machine
  - Vehicle
    - Spacecraft
      - Space-Station
        - \*International-Space-Station
  - Cooling-System
    - \*EXPRESS-Avionics-Air-Assembly
  - Electrical-Device
    - Electronic-Device
      - \*PHaSE-Mix-Melt-System
      - \*PHaSE-Precision-Rotary-Stage-Motor-And-Sensors
      - \*PHaSE-Carousel-Rotary-Stage-Motor
    - Input-And-Output-Device
      - Touch-Screen
        - \*PHaSE-Touch-Screen
    - Electronics-Board
      - \*PHaSE-Motion-Control-Board
      - \*PHaSE-Frame-Grabber-Board
      - PHaSE-Correlator-Card
      - PHaSE-Interface-Card
    - Computer
      - \*Payload-General-Support-Computer
      - \*PHaSE-DACS-Main-Processor
  - Computer-Diskette-Drive
    - \*OCA-PGSC-Floppy-Drive
    - \*PHaSE-Floppy-Drive
  - Electrical-Switch
    - \*PHaSE-Touch-Screen-Cover
  - Circuit-Breaker-Switch
    - \*PHaSE-MOTOR-PWR-Circuit-Breaker
    - \*PHaSE-COMPUTER-PWR-Circuit-Breaker
    - \*PHaSE-MAIN-PWR-Circuit-Breaker
  - Circuit-Breaker
    - Circuit-Breaker-Switch
      - \*PHaSE-MOTOR-PWR-Circuit-Breaker
      - \*PHaSE-COMPUTER-PWR-Circuit-Breaker
      - \*PHaSE-MAIN-PWR-Circuit-Breaker
  - Electrical-Consumer
    - Motor
    - Light-Bulb
      - Power-On-Light
        - \*PHaSE-INSTRUMENT-FAULT-Light
        - \*PHaSE-SAMPLE-MIXING-Light
        - \*PHaSE-MOTOR-PWR-Light
        - \*PHaSE-COMPUTER-PWR-Light
        - \*PHaSE-MAIN-PWR-Light
        - \*SIR-2-PWR-Light
- Power-Supply
  - \*Shuttle-Power-Supply

- Hardcopy-Information-Source
  - Hardcopy-Document
    - Specification-Document
  - Written-Section
    - Written-Process-Section
      - \*PHaSE-Hardware-Description
      - \*PHaSE-Alternate-Procedures-List
      - \*PHaSE-Nominal-Procedures-List
      - PHaSE-Checklist
        - \*PHaSE-Deactivation-Procedure
        - \*PHaSE-Activation-Procedure
      - PHaSE-Checklist-Step
        - \*PHaSE-Activation-Step-7
        - \*PHaSE-Activation-Step-6
        - \*PHaSE-Activation-Step-5
        - \*PHaSE-Activation-Step-4
        - \*PHaSE-Activation-Step-3
        - \*PHaSE-Activation-Step-2
        - \*PHaSE-Activation-Step-1
      - PHaSE-Checklist-Substep
        - \*PHaSE-Activation-Substep-6b
        - \*PHaSE-Activation-Substep-6a
        - \*PHaSE-Activation-Substep-5e
        - \*PHaSE-Activation-Substep-5d
        - \*PHaSE-Activation-Substep-5c
        - \*PHaSE-Activation-Substep-5b
        - \*PHaSE-Activation-Substep-5a
        - \*PHaSE-Activation-Substep-4c
        - \*PHaSE-Activation-Substep-4b
        - \*PHaSE-Activation-Substep-4a
        - \*PHaSE-Activation-Substep-2c
        - \*PHaSE-Activation-Substep-2b
        - \*PHaSE-Activation-Substep-2a
      - PHaSE-Checklist-Indented-Substep
      - PHaSE-Checklist-Caution-Box
        - \*PHaSE-Activation-Caution-Box-4
  - Written-Table
- Physical-Container
  - Tool-Box
    - \*EXPRESS-L2E/2-Toolkit
  - Electronics-Rack
    - EXPRESS-Electronics-Rack
      - EXPRESS-Rack-For-MSL
        - \*EXPRESS-Rack-For-MSL-And-PHaSE
  - Electronics-Drawer
    - \*EXPRESS-SIR-2-Drawer
  - EXPRESS-Middeck-Locker
    - Group-Of-EXPRESS-Middeck-Lockers
      - \*EXPRESS-Lower-Right-Pair-Of-Middeck-Lockers
      - \*EXPRESS-Lower-Left-Pair-Of-Middeck-Lockers
  - Universal-Small-Experiment-Container
    - \*PHaSE-Test-Section-USEC
  - PHaSE-Sample-Cell
- Storage-Device
  - Circular-Storage-Device

- \*PHaSE-Carousel-Of-Sample-Cells
- PHaSE-Payload-Flight-Data-File
- Agent
  - Organization
    - \*POCC
    - \*NASA
  - Person
    - ISS-Astronaut
- Information-Bearing-Object
  - Electronic-Information-Source
    - Electronic-Command
    - Computer-Diskette
      - PHaSE-Floppy-Disk
  - Display-Device
    - ...
  - Hardcopy-Information-Source
    - ...
- Activity
  - Inserting
  - Holding
  - Checking
  - Placing
  - Waiting
  - Uplinking
  - Preparing
  - Activating
  - Deactivating
  - Contacting
  - Organizational-Project
    - \*The-PHaSE-Project
  - Opening
    - Opening-A-Hinged-Cover
  - Closing
    - Closing-A-Hinged-Cover
  - Fastening
    - Tightening-A-Screw
  - Unfastening
    - Loosening-A-Screw
  - Storing
  - Removing
    - Removing-From-Storage
    - Removing-Diskette-From-Drive
  - Throwing-A-Switch
    - Switching-Off
    - Switching-On
      - Switching-On-The-Main-Power
  - Happening
    - Breaking
  - Repairing
  - Stowing
  - Unstowing
  - Changing-Light
- Group
  - \*PHaSE-Group-Of-4-APDs
  - \*PHaSE-Group-Of-Interface-Cards

- \*PHaSE-Group-Of-2-Correlator-Cards
- \*PHaSE-Touch-Screen-Cover-Screws
- \*PHaSE-Group-Of-8-Sample-Cells
- Intangible-Thing
  - Mental-Object
    - Information
      - Question-Type
        - \*image-qn
        - \*used-by-qn
        - \*physical-properties-qn
        - \*results-qn
        - \*possible-mistakes-qn
        - \*possible-failures-qn
        - \*specifications-qn
        - \*location-qn
        - \*safety-wear-qn
        - \*tools-and-equipment-qn
        - \*preconditions-qn
        - \*reason-for-doing-qn
        - \*malfunctions-qn
        - \*describe-process-qn
        - \*purpose-qn
        - \*parts-qn
        - \*science-action-qn
        - \*normal-response-qn
        - \*extra-info-qn
        - \*substeps-qn
        - \*wording-qn
        - \*definition-qn
- Specification
  - \*PHaSE-Payload-Operating-Procedure
  - PHaSE-Payload-Flight-Data-File
- Color
  - \*Blinking-Red
  - \*Blue
  - \*Green
- Information-Bearing-Thing
  - Reference
- State
  - Failure-State
    - \*Dummy-Test-Failure
- Property
  - \*Broken
  - \*On
  - \*Switched-On
  - \*Opened-Up
  - \*Unfastened
  - \*Unstowed
  - \*Off
  - \*Not-Broken
  - \*Switched-Off
  - \*Fastened
  - \*Closed-Up
  - \*Stowed
  - \*Closed

- \*Open
- Direction
  - \*Counter-Clockwise
  - \*Clockwise
- Physical\_Artifact
  - Breakable-Device
    - \*PHaSE-Carousel-Rotary-Stage-Motor
  - Light-Bulb
    - ...

## Appendix B: Some Component Mini-Theories

### DAG (Directed Acyclic Graph)

#### Synopsis

**Name:** dag

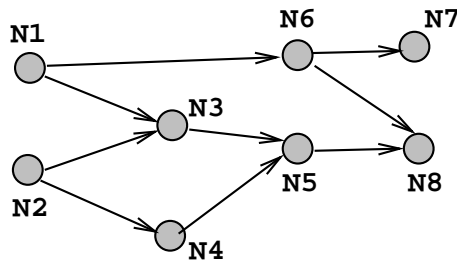
**Summary:** Core theory of directed acyclic graphs.

**Uses:** (none)

**Used by:** blockable-dag

#### Description

This component provides a basic axiomatization of DAGs, a fundamental structure for modeling many real-world phenomena.



A NODE is directly linked TO and FROM zero or more other nodes. In this representation, connecting arcs are represented using the binary relations TO/FROM, rather than reifying them into explicit objects. A node REACHES all its downstream nodes, and is REACHABLE-FROM all its upstream nodes.

#### Example

The above example DAG (here called `Dag1`) is specified by:

```

to(N1,N3).      to(N1,N6).      to(N2,N3).      to(N2,N4).      to(N3,N5).      to(N4,N5).
to(N5,N8).      to(N6,N8).      to(N6,N7).

nodes(Dag1,N1). nodes(Dag1,N2). ...      nodes(Dag1,N8).

reaches(N1,N3). reaches(N1,N5). reaches(N1,N8). reaches(N1,N6). reaches(N1,N7).
..
reachable-from(N6,N1).
reachable-from(N7,N1).
...
  
```

#### Ontology (objects and relations)

Concept Taxonomy

-----

DAG  
Node

Relations

-----

<code>nodes(DAG, Node)</code>	<code>[ inverse: nodes-of(Node, DAG) ]</code>
<code>to(Node, Node)</code>	<code>[ inverse: from(Node, Node) ]</code>
<code>reaches(Node, Node)</code>	<code>[ inverse: reachable-from(Node, Node) ]</code>

### **Axioms (English)**

- A node REACHES all its downstream nodes, and is REACHABLE-FROM all its upstream nodes.
- TO and FROM are inverse relations.
- REACHES and REACHABLE-FROM are inverse relations.

# Blockable-DAG

## Synopsis

**Name:** blockable-dag

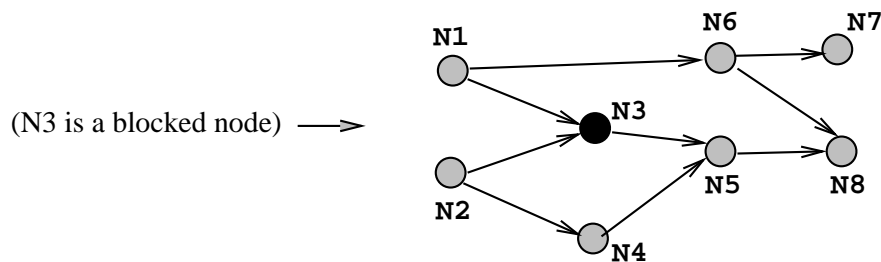
**Summary:** Extension to DAG theory, in which nodes can be blocked (thus preventing reachability).

**Uses:** dag

**Used by:** distribution-network

## Description

This specialization of a DAG adds the notion of blockability, preventing reachability.



A NODE may be BLOCKED or UNBLOCKED. A node UNBLOCKED-REACHES a downstream node only if there is a path of UNBLOCKED nodes connecting the two. In the above example, N1 unblocked-reaches N6, N7, and N8, but not N5 as N3 is blocked (and there are no alternative paths to N5).

Note that N3 still UNBLOCKED-REACHES N5, even though N3 itself is marked as blocked.

## Example

Blockability is specified using the `blocked?` relation, whose values are `*Is-Blocked` and `*Isnt-blocked`.

```
blocked?(N1, *Isnt-blocked). blocked?(N3, *Is-blocked).  
blocked?(N2, *Isnt-blocked). blocked?(N4, *Isnt-blocked).
```

```
unblocked-reaches(N1, N6).  
unblocked-reaches(N1, N7).  
unblocked-reaches(N1, N8). ; but not N5  
...
```

## Applications

- A power circuit can be modeled as a DAG, where a valve/switch is modeled as a blockable node.

## Ontology extensions (objects and relations)

Concept Taxonomy

-----

Blocked-State

\*Is-Blocked

\*Isnt-Blocked

#### Relations

-----

blocked?(Node, Blocked-State)

unblocked-reaches(Node, Node)

unblocked-reachable-from(Node, Node) ; inverse of unblocked-reaches

#### Axioms (English)

- If a node REACHES a downstream node via a path of UNBLOCKED nodes, then it UNBLOCKED-REACHES that node also.
- UNBLOCKED-REACHABLE-FROM is the inverse of UNBLOCKED-REACHES.

# Distribution-Network

## Synopsis

**Name:** blockable-dag

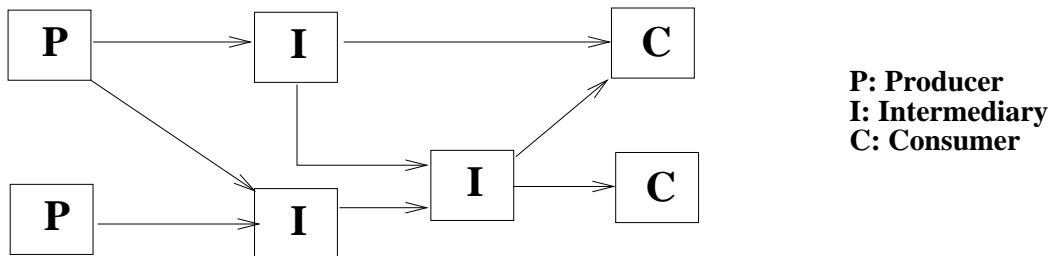
**Summary:** Simple theory of producers, intermediaries and consumers.

**Uses:** blockable-dag

**Used by:** elec-circuit

## Description

We use the clockwork of reasoning about a DAG onto concepts in a distribution network. A distribution network consists of four key classes of elements: PRODUCER, CONSUMER, INTERMEDIARY, and TRANSPORT-MATERIAL (eg. *Water*). Examples include: electrical circuits, hydraulic circuits, commuter traffic.



In this model, there is a flow of TRANSPORT-MATERIAL from PRODUCERS to CONSUMERS via INTERMEDIARIES, providing the intermediary is not BLOCKED. We thus model flow as passing the transport-material along a line (actually a DAG), rather than around a closed loop.

A CONSUMER/INTERMEDIARY is SUPPLIED if there is at least one UNBLOCKED path to it from a SUPPLIER. All elements in the network transport that network's TRANSPORT-MATERIAL.

This model does not include quantitative analysis of flow-rates etc., just that the PRODUCER CREATES TRANSPORT-MATERIAL, and the CONSUMER CONSUMES (eg. destroys/absorbs/uses) TRANSPORT-MATERIAL.

This model is a simple specialization of the `blockable-dag`, with signature morphism (symbol renaming) of `DAG`  $\rightarrow$  `Network`.

## Applications

- An electrical circuit, where we wish to reason whether a power-supply can deliver power to an electrical-device.

## Ontology (main objects and relations)

Concept Taxonomy

-----  
Node

Node-In-Network

Producer

Intermediary

Consumer

## Network

### Relations

-----

supplied?(Consumer)  
produces-material(Producer, Material-Type)  
conveys-material(Intermediary, Material-Type)  
consumes-material(Consumer, Material-Type)  
transport-material(Network, Material-Type)  
unblocked-reaches(Node, Node)  
unblocked-reachable-from(Node, Node)

### Axioms (English)

- If a CONSUMER is UNBLOCKED-REACHABLE-FROM a PRODUCER, then it is SUPPLIED.
- A NODE which is in a NETWORK is a NODE-IN-NETWORK.
- NODE-IN-NETWORKs (producers/intermediaries/consumers) all produce/convey/consume that network's TRANSPORT-MATERIAL.

# Elec-Circuit

## Synopsis

**Name:** elec-circuit

**Summary:** Top level concepts for reasoning about electrical circuits.

**Uses:** distribution-network

**Used by:** (none)

## Description

We use concepts from the `distribution-network` theory to model an electrical circuit as follows:

Electrical-Device	←	Node
Electrical-Power-Supply	←	Producer
Electrical-Appliance	←	Consumer
Electrical-Connector	←	Intermediary
powered?	←	supplied?
circuit-condition	←	blocked?
from	←	from
to	←	to
Open	←	*Is-blocked
Closed	←	*Isnt-blocked
Electrical-Circuit	←	Network
Electrical-Device-In-Circuit	←	Node-In-Network
components	←	nodes
components-of	←	nodes-of

Thus, in this model, an ELECTRICAL-POWER-SUPPLY provides ELECTRICITY to ELECTRICAL-APPLIANCES via ELECTRICAL-CONNECTORS. ELECTRICAL-CONNECTORS (eg. a switch) may be OPEN (off) or CLOSED (on). An appliance is POWERED if there is an open connection from at least one power supply.

This theory exports a condition to the machine theory, namely an ELECTRICAL-DEVICE is a MACHINE, and it must (at least) be POWERED to be WORKING.

## Ontology (main objects and relations)

### Concept Taxonomy

-----

```
Electrical-Device
  Electrical-Device-In-Circuit
  Electrical-Power-Supply
  Electrical-Appliance
    Light-Bulb
    Motor
  Electrical-Connector
    Switch
Circuit-Condition
  *Open
  *Closed
Electrical-Circuit
```

## Relations

-----

powered?(Electrical-Device)

broken?(Electrical-Device)

working?(Electrical-Device)

circuit-condition(Electrical-Device, Circuit-Condition)

from(Electrical-Device, Electrical-Device) ; inverse to

components(Electrical-Device, Electrical-Circuit) ; inverse components-of

transport-material(Electrical-Device, \*Electricity)

## Axioms (English)

- The TRANSPORT-MATERIAL of all ELECTRICAL-DEVICES is ELECTRICITY.
- An electrical-device must (at least) be POWERED to be WORKING.

# Machine

## Synopsis

**Name:** machine

**Summary:** Rather general description of a machine.

**Uses:** (none)

**Used by:** Many.

## Description

A MACHINE is a thing which can be WORKING. To be working, it must (at least) not be BROKEN. Other theories contribute other conditions for working (eg. that it is POWERED, has all necessary INPUTS, is designed correctly). This theory doesn't commit to what those extra conditions are, but will just gather whatever is available.

## Ontology (main objects and relations)

Concept Taxonomy

-----

Machine

Relations

-----

fluents(Machine, Formula)

conditions-for-working(Machine, Boolean)

working-state(Machine)

broken?(Machine)

## Axioms (English)

- a MACHINE is WORKING if all its CONDITION-FOR-WORKINGS are satisfied.
- a CONDITION-FOR-WORKING is that the machine is not BROKEN.