Projection and Reaction for Decision Support in Refineries: Combining Multiple Theories

Kurt D. Krebsbach
Department of Computer Science
Lawrence University
115 South Drew Street
Appleton, WI 54911
kurt.krebsbach@lawrence.edu

David J. Musliner
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418
david.musliner@honeywell.com

Introduction

Modern decision support systems are forced to deal with perpetual change in their environment: change that includes a high degree of uncertainty over the current state of the plant, effects of actions, accuracy of sensor readings, and consistency of execution of approved procedures by plant operators.

In this paper we report on AEGIS (Abnormal Event Guidance and Information System), a large-scale automated system to provide decision support for refinery operations personnel (Krebsbach & Musliner 1997; Musliner & Krebsbach 1998). In particular, we will concentrate on the components concerned with setting goals, planning actions, executing those actions (or suggesting actions to be manually executed), observing action effects, and recovering from unintended plant states. We will describe what worked well, what did not, and why future such attempts must go outside single theories of planning and acting to provide sufficiently flexible decision support in complex environments.

Background: Refinery Control

Petrochemical refining is one of the largest and most complex industrial endeavors worldwide. The functional heart of a refinery is the Fluidized Catalytic Cracking Unit (FCCU). As illustrated in Figure 1, the FCCU is primarily responsible for converting crude oil (feed) into more useful products such as gasoline, kerosene, and butane (Leffler 1985). The FCCU *cracks* the crude's long hydrocarbon molecular chains into shorter chains by combining the feed with a catalyst at carefully controlled temperatures and pressures in the riser and reactor vessels. The resulting shorter chains are then sent downstream for separation into products in the fractionator (not shown). The catalyst is sent through the stripper and regenerator to burn off excess coke, and is then reused.

Figure 2 illustrates how a typical state-of-the-art refinery is controlled. The Distributed Control System (DCS) is a large-scale programmable controller tied to plant sensors (e.g., flow sensors, temperature sensors), plant actuators (e.g., valves), and a graphical user interface. The DCS implements thousands of simple Proportional Integral Derivative (PID) control loops to make control moves based on discrepancies between setpoints (SPs) and present values (PVs). For example, as depicted in Figure 1, the dotted line connecting the temperature sensor and the riser slide valve denotes that the position of the slide valve is dependent on the temperature being sensed in the riser. As the temperature drops, the slide valve will be opened to increase the flow of hot catalyst. A typical FCCU will have on the order of a thousand readable "points" (sensors), and a few hundred writable "points" (actuators). In addition to PID control loops, the DCS can be programmed with numerous "alarms" that alert the human operator when certain constraints are violated (e.g., min/max values, rate limits). "Advanced control" is the industry term given to more powerful mathematical control techniques (e.g., multivariate linear models) used to optimize control parameters during normal operations.

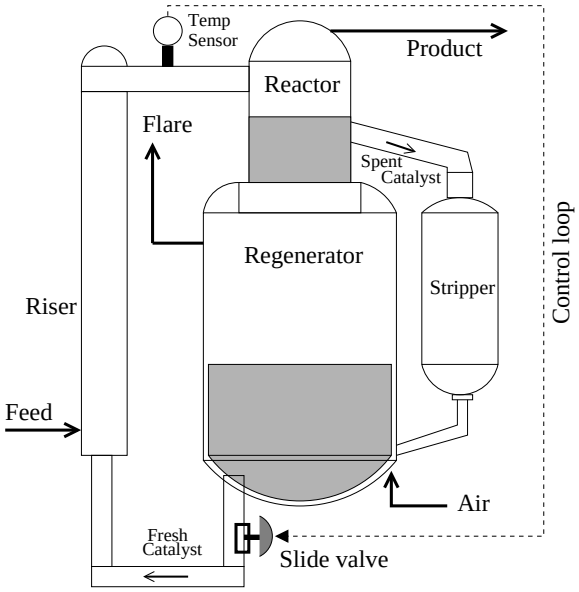

Figure 1: A Fluidized Catalytic Cracking Unit.

Current Refinery Operations

Human operators supervise the operation of the highly-automated plant. This supervisory activity includes monitoring plant status, adjusting control parameters, executing pre-planned operational activities (e.g., shutting down

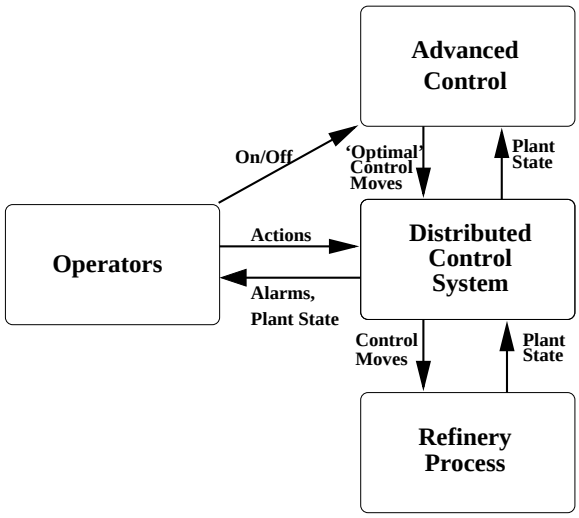

Figure 2: State-of-the-Art Refinery Control.

a compressor for maintenance), and detecting, diagnosing, compensating for, and correcting abnormal situations. The operator has a view of the values of all control points, plus any alarms that have been generated. The actions the operator is allowed to take include changing SPs, manually asserting output values for control points, and toggling advanced control modules.

Abnormal Situations

Abnormal situations occur frequently in refineries, and violate many of the assumptions upon which the DCS control systems are designed. Minor incidents often cause dozens of alarms to trigger, requiring the operator to perform anywhere from a single action to dozens, or even hundreds, of compensatory actions over several hours. Major incidents can precipitate an *alarm flood*, in which hundreds of alarms trigger in a few seconds, leading to scrolling lists of alarm messages, panels full of red lights, and insistent klaxons. In these situations, the operator is faced with severe information overload, which often leads to incorrect diagnoses, inappropriate actions, and major disruptions to plant operations. Abnormal situations can be extremely costly, resulting in poor product quality, schedule delays, equipment damage, reduced occupational safety, and environmental hazards.

Current Practice: What doesn't work

Operators undergo extensive training to apply **Standard Operation Procedures** (SOPs) in a wide variety of situations. Many of these procedures are periodically performed during normal, steady-state operation (e.g., unit shutdown to replace catalyst every three months), while other procedures are specifically designed as a response to an abnormal situation (e.g., loss of combustion air to the FCCU).

Because abnormal situations are so serious, many regulatory and administrative structures are already in place to help manage them. Primarily, operators are trained to respond to abnormal situations based on extensive written SOPs. The procedures can be quite long (dozens of pages), with a high degree of logical control structure and contingency to offset the fact that the exact state of the plant is almost never known with certainty. Many procedures involve sampling data, confirming other readings, performing diagnostic tests, conferring with other plant personnel, and adjusting DCS control parameters. Some procedures apply to extremely general contexts (e.g., we're losing air pressure from somewhere), while some are less general (air compressor AC-3 has shut down), and some are very specific (the lube oil pump for AC-3 has a broken driveshaft).

Refineries are required to develop, distribute, train for, and submit written SOPs for every common or dangerous potential situation. In several such plants, however, we observed that many three-ring binders of SOPs were stored away and had not been touched or updated for years. Of course, other, more regularly-performed procedures were more accessible.

The biggest problem, however, was the fact that these written procedures were interpreted quite differently from plant to plant, and from operator to operator, which was hardly OSHA's (the Occupational Safety and Health Administration's) original intent. For example, some trainers regarded such OSHA-mandated procedures to be "rough guidelines," while others insisted that their procedures were intended to be followed to the letter and were under constant review for possible improvement. Likewise, we found 12-hour procedures consisting of a few general steps (e.g., incrementally reduce the temperature in a vessel over the course of 4 hours), which, when explained to us, was a highly-complex procedure of its own (e.g., requiring constant monitoring of dozens of variables, and execution of dozens of well-timed, situation-specific setpoint updates). When asked why some procedures were very detailed while others contained no detail at all, we were told that there was wide disagreement between operators on the "best exact way" to perform certain steps, and that each operator learns, through experience, which way works best for him. Such an attitude did not, of course, result in repeatable or predictable execution of these procedures, and often, even as we watched, resulted in costly or dangerous miscommunication between operations personnel.

Theory 1: Procedural Reasoning

We implemented the core reasoning engine of AEGIS in C-PRS, the C-based version of the Procedural Reasoning System (Ingrand 1994; Ingrand, Georgeff, & Rao 1992; Georgeff & Lansky 1986). As shown in Figure 3, knowledge in PRS is represented as a declarative set of facts about the world, together with a library of user-defined *knowledge areas* (KAs) that represent procedural knowledge about how to accomplish goals in various situations. Goals represent persistent desires that trigger KAs until they are satisfied or removed. The *intention structure* represents currently-selected KAs that are in the process of executing or awaiting execution, in pursuit of current goals. The PRS *interpreter* chooses KAs appropriate for current goals, selects one or more to put onto the intention structure, and executes one step from the current intention.

We chose an integrated approach to Goal setting, Planning, and Execution (GPE) based on the AI community's

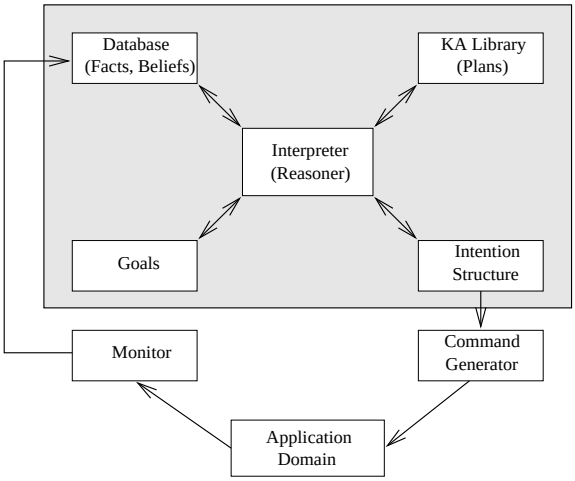

Figure 3: The Procedural Reasoning System.

past experiences with autonomous systems applied to real-world domains (e.g., robotics); experience that has shown that choosing a goal to pursue, planning a course of action, and executing the steps of the plan are inevitably intertwined by the unpredictable and dynamic nature of real-world domains. Execution failures, changing goals, difficult planning problems, and environmental changes all disrupt the ideal of simple forward information flow. If the GPE functions were separated into distinct programs, the amount of information constantly passing back and forth due to the changing domain, plans, and goals would be overwhelming. In our integrated GPE approach, in contrast, those changes are kept largely local to GPE, so the C-PRS interpreter can be efficient about managing that information.

Other features of PRS which have proven to be extremely useful for this domain include the following:

- The **hierarchical**, subgoaling nature of the procedural representation that allows PRS to combine pieces of plans in novel ways, which is important for flexible plan execution and goal refinement.
- Its ability to pursue multiple, **goal-directed** tasks while at the same time being **responsive** to changing patterns of events in bounded time.
- The ability to construct and act on **partial** (rather than complete) **plans**.
- **Meta-level** (or reflective) reasoning capabilities, an important feature for controlling the allocation of processing resources, planning attention, and alternative goal achievement strategies.
- Knowledge representation assumptions, which encourage **incremental refinement** of the plan (procedure) library, an enormous advantage for large-scale applications. For example, a knowledge engineer could simply add a single, temporary KA to the plan library to reflect that an alternative spill route will be used until a normally-available tank is freed up, rather than redefine the plant equipment model to allow for this temporary situation.

PRS: What doesn't work

There were several major obstacles to making this first approach sufficient in a real refinery setting:

- **Plan Artifacts:** While dynamically generating a plan customized to the specific situation at hand seemed appropriate and desirable from an AI standpoint, it was clearly inconsistent with current practices at refineries. What plan would we show to OSHA? Which of the combinatorial number of potential plans should be printed out? In short, OSHA could not approve plans that didn't already exist, and trainers could not print out these plans to use for training sessions in the conventional way. The lack of a pre-existing plan artifact seemed to simultaneously be both the power of our approach, and the death-blow to the existing refinery culture!
- **Operator Check-Offs:** Keeping everyone on the team (including the system) in sync required operators to "check off" tasks when they knew they were actually completed. In general, operators strongly resisted this added responsibility and could not be counted on to do it. Also, in many refinery situations, it is impossible to automatically infer from sensor readings when an action had been taken, either because of an under-sensored plant (sensors were often eliminated because sensor replacement was the lion's share of plant maintenance costs), or because the needed data would not become available early enough to be useful (e.g., the valve has been manually closed, but the temperature difference won't be noticeable for 20 minutes, which is too long to delay the next step in the plan).
- **Procedure Maintenance:** PRS procedures were implemented for field tests by AI specialists. Plant personnel were therefore understandably concerned about implementing, updating, and maintaining the large set of plant procedures that were initially and painstakingly transformed from text files printed on paper into an executable PRS plan library.

Projection vs. Reaction

Classical AI planners *project* forward the actions they are considering, deriving the expected results of the actions and using that projected information to make planning decisions (Fikes & Nilsson 1971; Wilkins 1988). Projection allows a planner to think about the future and build plans in intuitively powerful ways; however, performing this type of projection requires a model of the environment that supports simulation. You must be able to "run time forward" on the model to find out what the state resulting from a particular action will look like. In many real world domains, including oil refineries during abnormal situations, these sorts of models are simply not commercially available, and those that are lack the fidelity required to support projection. Even worse, during an abnormal situation, such models will no longer be accurate, almost by definition: if something is broken, then some component is not functioning according to predictions, or else the system would be in normal operations. In fact, it is this very observation–that plant models are *not* generally

available (and seem impractical with current modeling and diagnosis technology)–that led us to pursue a more *reactive* approach using PRS in the first place.

The reactive nature of PRS has pros and cons relative to projection. On the positive side, the procedures are easier to understand and control because they can look a lot like an SOP. PRS is essentially doing the job of dynamically invoking the procedures and managing their complexity. On the negative side, this means that PRS can only generate new combinations of those procedures: it cannot synthesize new procedures from lower-level primitive descriptions like a synthetic projective planner might. On the positive side, PRS does not require a complex, accurate model of the plant to achieve intelligent behavior. On the negative side, it means PRS can't tell you a lot about what it expects to do beyond the scope of a single procedure (whose size and granularity is really up to the knowledge engineer).

Because PRS is reactive, it does not look ahead to determine which procedure it will select to achieve a given goal until that goal has been reached in the procedure. We believe this is "correct" from an engineering perspective, because the precise method of achieving a goal should not be determined until the full environmental context is available for evaluating the alternatives. This context can only be known when the goal is posted, not before; however, this is insufficient from the operator's perspective, because it provides little insight into what the system is planning globally.

In summary, there are two distinct aspects to this problem within the context of executing a single PRS procedure:

1. **Future goal-achieving procedures are not yet selected.** PRS procedures are, in the simplest serial case, executed like a normal computer program[1]. When PRS selects a procedure, it instantiates it, and sets the "program counter" at the first goal. Applicable procedures are determined to achieve that goal, and one is chosen. While this newly-chosen procedure is being executed, however, selection of procedures for goals beyond the program counter is deferred. Some future goals and actions are known but not available to the interpreter. Although the names of goals and primitive actions beyond the program counter are available in the procedure source code by inspection, they are not available to the C-PRS interpreter until the program counter arrives.

2. **Future goals and actions are not necessarily meaningful to the user anyway.** Even if future goals and actions could be accessed by the interpreter, some are at the wrong level to be relevant to the user (e.g., binding a local variable), while others are not in a form useful to an operator (code), or easily translatable into such a form. In general, it is not reasonable to expect the PRS procedure author to use (implementation-based) names and constructs that correspond to an operator's (domain-based) understanding, and vice versa.

[1] C-PRS also supports parallel goal achievement, but that capability does not affect this discussion.

Theory 2: Pseudo-Projection

To work around this problem, we have developed a *pseduo-projection* method that allows GPE to appear partially projective without making any changes to the reactive PRS interpreter. Pseudo-projection allows the operator to see as far into the future, and with as much detail as is possible, given the reactive procedural paradigm.

We implement pseudo-projection using a procedure annotation syntax that allows the author to annotate each procedure with a series of comments that the AEGIS user will see at runtime when the procedure is chosen by PRS. We refer to these annotations, which reside entirely within PRS comments, as *metacomments.* In this way, we can load the annotated procedures into PRS directly and all of their behaviors will operate normally except for the user interface functions. These interface functions are generated automatically from the metacomments.

The *Magical MetaComment Parser* (MMCP) reads the metacomments and converts them into active PRS statements that display information to the user and allow the user to interact with the procedure (e.g., granting permissions or assuming responsibility). First, the modified procedure sends a summary of its own structure to the user interface, allowing the UI to produce a skeletal display of the procedure. Then the modified procedure allows the user to assume responsibility for the procedure's goal, if the procedure has not already begun. The MMCP looks at both the raw procedure code and the metacomments, extracting control structure from the procedure code and appropriate labels for blocks of code from the metacomments.

The MMCP and appropriate metacommenting allows PRS to appear partially projective to the user. As soon as a procedure is selected, the user can see the entire structure of the procedure, as well as the status of the different code blocks (goals, atomic actions, conditionals, etc.). The MMCP also automatically wraps the procedures in code that interacts with the user interface to allow the user to reassign responsibility for achieving the goal himself. To maintain the appropriate succeed/fail semantics, the user interface then allows the user to specify when he has completed a task he earlier assumed, to allow the higher-level procedure that invoked the goals to continue. For example, a procedure may post a goal to close a valve and the user may decide to perform that action himself. When the user completes the action, he indicates whether he has succeeded or failed. In the event of success, PRS can assume the procedure has completed (although it will not assume the goal has been achieved until that has been independently confirmed). If instead the user has failed, and PRS has other methods available to achieve the goal, it will try them, as usual.

Pseudo-Projection: What doesn't work

The MMCP is a temporary approach to the problem of user awareness in a reactive system, and suffers from several serious deficiencies. First, it adds complexity to the process of writing procedures, although the metacomment syntax itself is quite simple. In part, this added complexity is unavoidable if we wish the user to see a representation of the procedure

that is somehow simplified, abstracted, or in different terms than the raw procedure code itself. However, the MMCP approach is less than ideal even assuming that some annotation must be done, because its operation is entirely external to the PRS interpreter. This means that the MMCP must do a lot of work (e.g., assigning unique identifiers to procedure steps) in procedure code that runs relatively inefficiently. If the MMCP functionality was pushed into the PRS interpreter, these common operations could be automatically performed whenever a procedure is loaded (at compile time), without the overhead associated with each procedure goal statement.

Theory 3: Mini-Models

While pseudo-projection techniques provide a form of lookahead for the user, other limited forms of model-based projection can be exploited which allow more intelligent control by the reactive system itself. Consider the following simplified procedure segment for responding to a loss of combustion air:

Procedure Novice-Air-Loss-Response

1. Cut riser temperature to 930 degrees F.
2. Eliminate all residual feed.
3. Eliminate all slurry pumparound feed.
4. Cut main feed to 20,000 barrels/day.
5. Add pure oxygen up to 30% enrichment.

This procedure fragment is a typical example from an SOP manual. Such procedures are characterized by simple instructions, designed to be understandable by even the most novice operator. They are straightforward, safe, static, and suboptimal. For instance, all residual and slurry feeds are eliminated to allow the operator to concentrate on cutting and monitoring only the main feed.

While these procedures provide a starting point for encoding executable procedures, they do not accurately reflect the complexity of most operators' response to an abnormal situation. As operators gain experience, their knowledge of the underlying plant process and DCS response grows, and their response becomes more model-based. In our example scenario, experienced operators would generally leave in some residual feed to keep the coke component higher, keeping the riser temperature higher. This is an optimization step that, while still safe, maintains a higher level of production, and thus reduces the cost of the disruption.

In general, the more experienced the operator, the more context-sensitive is his response to an abnormal situation. We view our GPE procedures as evolving in the same way, incorporating more of what we have called *mini-models* directly within PRS procedures. As the authors of the procedures gain a better understanding of the process and control system, we expect the procedures to rely less on static responses, and more on computing over a simplified model to generate a context-sensitive response. The following is a more model-based version of the same procedure, emulating the expert-operator approach:

Procedure Expert-Air-Loss-Response

1. Compute amount of O2 in lost air.
2. Add pure O2 to replace lost O2, up to 30% enrichment max.
3. Compute O2 left to replace.
4. Compute amount of carbon this corresponds to.
5. For each feed source, cut source according to carbon factor.
6. Set riser temperature setpoint based on remaining carbon.

This procedure concentrates on balancing carbon content of the current feed sources with the amount of oxygen available, while staying within safety limits of 30% enrichment. It is based on a simplified mini-model involving a handful of important factors in the process, and is thus much more tailored to the actual circumstances at the process at the time of its invocation. In this example, GPE can greatly assist the operator by easily and automatically computing parameters to the situation response (e.g., correct riser temperature), and provide the option for GPE to take the actions autonomously and monitor the effects of these actions over time.

Theory 4: Existing Predictive Models

In addition to mini-models directly implemented with PRS procedures, small predictive models exist as black-box applications for very limited pieces of the refinery. While these models are quite small (e.g., ten inputs, four outputs), in certain contexts they can be invoked from within a PRS procedure to provide several valuable types of information. In many circumstances, GPE has a choice of action. By projecting these models forward for each option, GPE can assess the effectiveness of each alternative and choose the best one. Also, the specific results of the projections can often be valuable information to the operator and to GPE. In close cases, the operator might prefer one method over another for less tangible reasons than GPE is able to consider. From GPE's perspective, the results form a rank ordering of the options, which can be cached and used if the first goal-achievement method fails. Finally, the specific expected results can inform GPE in establishing its own monitoring parameters.

Next Steps: Main Line Projection

While pseudo-projection provides a higher degree of lookahead than native PRS by presenting a skeletal plan to the user, it still does not provide a completely instantiated plan artifact. As stated earlier, we believe this to be a good trade-off from an engineering standpoint, sacrificing complete lookahead for the flexibility of an appropriate, context-specific response. However, refinery personnel and government agencies like OSHA are still accustomed to working with procedures documented from beginning to end. We conclude by sketching one possible approach to overcome this conflict between engineering progress and industrial convention, which we will call *main line projection:*

1. Project an entire plan forward based on accepted plant policy and assumptions of how the underlying process will respond. Ignore all but the most likely plant response to every action, but carefully note the assumptions upon which later actions depend.
2. Explicitly track the assumptions made so that when one is violated, elements of the plan based on that assumption can be evaluated for possible replanning.
3. Modify only the affected procedure steps *in place* (i.e., without a complete replan) if possible from the current state forward to reflect the changing state of the process. This is possible because PRS understands the relationship between its high-level goals and lower-level actions meant to achieve those goals.
4. Maintain explicit information about which steps or actors require resources (e.g., storage tanks, feed, personnel, tools, time) and allocate them to enable higher priority goals to be satisfied first (current GPE uses a simple "first generated, first-served" scheme now).
5. Re-evaluate the allocation of resources when the plan changes to accommodate contingencies. This allocation can be done much more intelligently with pseudo-projection than with native PRS, since the system will always have access to the entire current plan.
6. Develop a user interface to clarify what is happening at any point in an evolving plan.

Conclusion: Multiple Theories

We argue that providing decision support in a domain as complex as refinery control requires combining multiple theories of planning and acting (e.g., reactivity and projection), supplemented by domain-specific techniques developed for well-understood sub-problems (e.g., mini-models and existing predictive control techniques). The primary challenge we raise is how best to combine these heterogeneous techniques into a cohesive, comprehensible system capable of performing well and predictably under adverse, high-risk conditions, and propose a set of next steps to resolve conflicts that still exist in the application of AI techniques to real-world, industrial, decision support problems.

References

Fikes, R., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:3:189–208.

Georgeff, M., and Lansky, A. 1986. Procedural knowledge. *IEEE Special Issue on Knowledge Representation* 74:1383–1398.

Ingrand, F.; Georgeff, M.; and Rao, A. 1992. An architecture for real-time reasoning and system control. *IEEE Expert* 7:6:34–44.

Ingrand, F. F. 1994. *C-PRS Development Environment (Version 1.4.0)*. Labege Cedex, France: ACS Technologies.

Krebsbach, K. D., and Musliner, D. J. 1997. A refinery immobot for abnormal situation management. In *AAAI '97 Workshop on Robots, Softbots, and Immobots: Theories of Action, Planning, and Control, Providence*.

Leffler, W. L. 1985. *Petroleum Refining for the Non-Technical Person*. Tulsa, OK: PennWell Publishing.

Musliner, D. J., and Krebsbach, K. D. 1998. Applying a procedural and reactive approach to abnormal situations in refinery control. In *Proc. Conf on Foundations of Computer-Aided Process Operations (FOCAPO)*.

Wilkins, D. 1988. *Practical Planning: Extending the Classical AI Planning Paradigm*. San Mateo, CA: Morgan Kaufmann.