AI Planning in Supervisory Control Systems

Luiz Edival de Souza
Departamento de Eletronica
Escola Federal de Engenharia de Itajubá
37500-000 Itajubá - MG, Brazil
ledival@cs.cmu.edu

Manuela M. Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
veloso@cs.cmu.edu

Abstract

This paper presents an AI-planning-based framework to support the activities of a human operator in a supervisory control system. The framework uses an AI planning and learning substrate architecture and is designed for integration within a general third-party real time software. Our goal is to build a test-bed architecture for research in AI planning applications, such as electrical and industrial processes. AI planning techniques, as opposed to the more traditionally used rule-based systems, can be useful in the automation of the supervision of process systems, as they provide rich planning representations and algorithms. We present our work developing an AI planning system for a boiler power plant domain. We develop a set of planning operators from an extended multilevel flow modeling (MFM) of the plant. Our planner reasons about goals and its subgoals, generates plans for different scenarios, including the sequence for start-up the plant. We show our approach to acquire the domain knowledge, which is a well-known difficult enterprise for real-world applications. We demonstrate that our modeling approach built upon MFM is successful in mapping the supervisory system knowledge into a planning representation.

1. Introduction

Due to the increasing complexity of the industrial processes, computerized-based supervisory control systems are being used to provide support to a human operator or a team of operators in a control room. The automation is intended to reduce or completely eliminate the continuous attention of the operator when the process state is in normal conditions. The human operator acts then as a supervisor. When the supervisory system detects an anomaly, the human operator must act as a controller and must react to the plant abnormal conditions. The operator's own understanding of the process states is used in this control task. However, paradoxically, an automated system aimed at reducing the human efforts, overloads the human operator with a lot of information during the abnormal conditions.

Artificial Intelligence (AI) planning techniques have the potential to be powerful tools to support the human operators to perform their tasks less repetitively and more accurately. This combination of AI planning with real engineering systems has been shown to be particularly hard in terms of acquiring robust task knowledge from human experts. Furthermore, this knowledge, when acquired, is rather domain dependent and the acquisition effort is very seldom amortized over other applications.

Many AI planning researchers, present authors included, envision the application of their planning systems to real world applications. A few efforts have been made in this direction and, in general, the large acquisition effort from human experts is neither compensated by complete and reliable representations, nor transferable to other similar applications. Nevertheless, many AI and control engineering researchers pursue the investigation of this potentially powerful combination of AI and real systems ([1], [2], [3], [4]).

In this paper, we report on our work on flexibly representing a functional model of a supervisory control system, such as a power plant in an AI planning system. We build upon the multilevel flow model (MFM) and its initial proposed application to planning. Our planning mechanism is based on the Prodigy architecture, a domain independent planning and learning system [5].

The article is organized as follows: Section 2 overviews the proposed supervisory system. Section 3 describes briefly the MFM concepts and we present our planning knowledge acquisition approach through a simple industrial process. Section 4 introduces the general framework to acquire the planning model. The domain-independent planning operators are derived from the MFM functional topology, the process states and the user tasks, extending therefore the means-ends description of the industrial process. In Section 5, we show how our planning system generates the start-up plan of a power plant. We present our techniques to produce hierarchical planning. Finally, Section 6 draws conclusions.

2. Overview of the proposed system

Figure 1 shows our approach to give support to the operator's activities in a supervisory control system, we are investigating the combination of a planning system and a real time software. The planning system is responsible for generating a series of complex control plans based on the current state of the system and according to the domain knowledge specified. The real time software is responsible for gathering the data from the industrial process and for presenting information to the human operator.

In a supervisory system, a human operator performs important tasks such as process monitoring, fault detection, diagnosis and planning, all of which imply the use of a high cognitive level. In general, the human supervisor needs to generate plans, consisting of a sequence of actions that when executed leads the system from an initial

This work was done while the first was a visiting researcher at Carnegie Mellon University with a grant from CNPq, Brasília, DF.

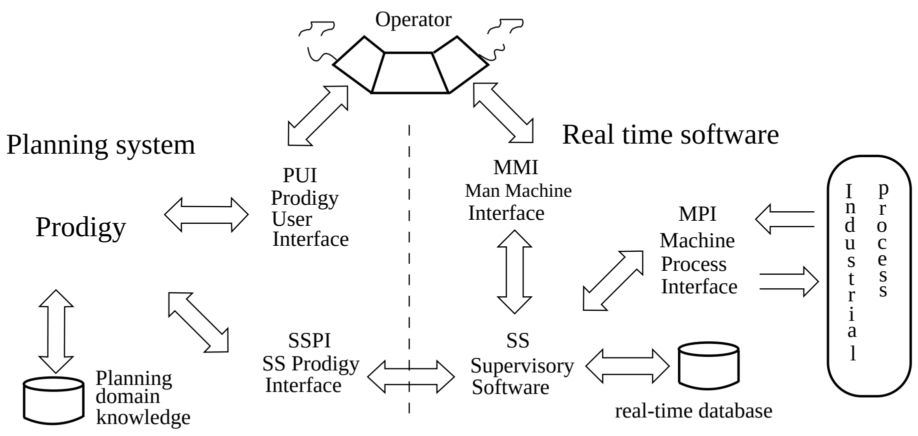

Fig. 1. Supervisory software and planning system.

state to a goal state. Typical actions in industrial processes can be to open or close a valve or to start or stop a motor. Depending on the complexity of the system, an action can be expressed at a high level of abstraction such as "start subsystem A" or "execute procedure X." Actions can also be expressed at lower levels of abstraction, for example specifying directly by how much and for how long to open a valve.

Planning system

Planning at the supervisory level in industrial processes can occur in two different forms, namely: (a) monolithic procedures compiled in operating manual; and (b) online planning responding to the feedback of information about the process state.

Compiled monolithic procedures are appropriate for planning operations such as startup, shutdown, or changing a system's general operating state. This approach is typically found in industrial processes such as power plants. A human operator can follow a procedure without fully understanding its limits of applicability. As a result, there is no actual knowledge of the reasoning behind the procedure. It is impossible to determine the exact effect of the procedures, such as whether steps can be reordered or if it is possible to "work around" a newly discovered fault. The procedure gives step by step instructions, for example for starting a pump, but it will not help the operator understand the deep knowledge about why the pump must be started in this way, and the consequences of not doing so. In abnormal conditions, the operator must perform reason about the process states to generate a solution to the problem. This analysis requires functional understanding of the current response of the system, which is considered an activity of the online planning.

Online planning is necessary when a process breaks down or is stopped in an abnormal situation. A wide variety of operating conditions may occur and the human operator needs to do online decision making. Depending on the fault, it might require a complex plan to bring the system back to the normal operational state. This is a very demanding task to the operator because of the limited time and information-processing resources. The operator needs efficient strategies to cope with tasks that require complex interpretations of real-time data and complex coordination of actions.

Although our proposal is well suited to investigate online planning, our main concern is to develop tools to give support during start-up of a plant or to help the operator in off-line planning. In spite of the challenging problems found in online planning, we believe that this step is important and fundamental to future researches.

Prodigy: Prodigy is a domain-independent problem-solving architecture used primarily as a test-bed for research in planning, machine learning, and knowledge acquisition. The Prodigy architecture consists of a core general-purpose state space nonlinear planner and several learning modules that refine both the planning domain knowledge and the control knowledge to guide the search process effectively. The nonlinear planner follows a means-ends analysis backward chaining search procedure reasoning about multiple goals and multiple alternative operators relevant to achieving the goals. The nonlinear character of the planner stems from its dynamic goal selection which enables the planner to fully interleave plans, exploiting common subgoals and addressing issues of resource contention.

User Prodigy Interface: We are currently developing extensions to the user interface provided by Prodigy [6] in order to provide a friendly communication to the human operator. Basically, this extensions will allow the human operator to set a problem and to receive a solution.

SS Prodigy Interface: This interface will be responsible for maintaining the internal state of the planner according to the real state of the system. It will be necessary to define a communication protocol between Prodigy and the real time system.

Planning domain knowledge: A planning domain is defined by a set of types of objects, i.e. classes, used in the domain, a library of operators and inferences rules that act on these objects. Prodigy's language for describing operators is based on the STRIPS domain languages, extended to express disjunctive and negated preconditions, universal and existential quantification, and conditional effects. Each operator is defined by its *preconditions* and *effects*. The descriptions of preconditions and effects can contain typed variables. This typed variables, called *planning variables*, can be constrained by functions.

Defining a planning domain is a difficult task. In the next section, we describe our approach to define a planning domain based on the multilevel flow modeling.

Real time software

Real time software has been built to reduce the human efforts in processing information, and to relieve the operator from the repetitive tedious tasks. The current systems focus on information gathering and visualization. In abnormal conditions, they provide very little guidance, if any, in terms of eventually needed high-level planning actions. The most sophisticated systems support a complete application development environment for building and deploying intelligent applications, such as G2. However, in general, they are based on generic rules and procedures to represent the expert knowledge.

3. Multilevel Flow Modeling

In [7], Lind describes a multilevel flow model (MFM) for engineering systems. This model constitutes a significant advance in abstracting the high-level functional description of a system from the low-level technical implementation details. MFM introduces *goals* as necessary concepts to capture the purposes of an engineering system. The goals are achieved by *functions* which describe the intentions of the system designer. The functions are the result of the behavioral interactions between the physical components of the system.

Basic principles of MFM

MFM uses the whole-part and means-ends decompositions to represent a man-made engineering system. The whole-part decomposition is used to describe refinements of a function in order to present more details of its realization. This process creates hierarchical structures and new functions and goals are generated. The means-ends decomposition seeks to identify the functions of the components and their relations to the goals of the system. The functions are ascribed to the system to achieve a specific goal and there is no meaning to specify functions which do not contribute to achieving goals.

The whole-part decomposition can be illustrated considering a pumping force in an industrial process. This function can be provided either by a pump or by natural circulation due to a difference in temperature or in height. To show the functional details of this individual function, a lower level of abstraction about the pumping force must be defined. For instance, a pump-based system will perform its function only if the electrical subsystem that supports the pump is working correctly. To represent this knowledge, a lower level decomposition can be defined with the details about the start-up system required by the pump.

The means-ends decomposition represents the functionality of a system through a set of structures that captures the *mass* and *energy* flow. Hence, the MFM is well suited to model plants that involve energy converting processes such as power plants and some chemical processes. The mass and energy flow structures are built based on a set of functions which are related in complementary pairs, namely source–sink, storage–balance, and transport–barrier. MFM uses a graphical representation and distinguishes between functions that process energy and functions that process mass as illustrated in Figure 2.

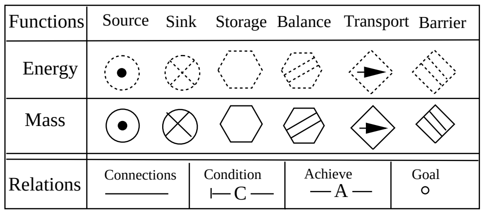

Fig. 2. Graphic symbols used by MFM the means-ends decomposition.

The structures are interrelated by goals through two means-end relations: achieve relation and condition relation. The achieve relation is used to describe the MFM goals that a structure achieves. Usually, a goal is achieved by one or more functions. The condition relation is used to connect an MFM goal as a precondition of a function.

Building a multilevel flow model

Building a model based on MFM is not straightforward. Most of the concepts are borrowed from thermodynamics. However, the functional concepts are not necessary related directly with the physical properties of the components or subsystem. They should be interpreted as more general functional concepts. A comprehensive understanding of these differences requires more experience and more studies. Acquiring the MFM models is beyond the scope of our work. Others, including [8] and [3], have studied methods for building a model based on MFM.

Figure 3(a) shows an industrial process which consists of a tank that stores some material and the inlet valve (valve-A) and the outlet valve (valve-D). The tank has two level sensors, namely a high-level (HL) and a low-level sensor (LL), to signal full and empty tank, respectively. This process can be modeled through a mass flow structure as illustrated in Figure 3(b). The functions represented in this structure captures the material flow and the storage function of the tank. The storage Stg1 function represents the tank, the transport Tr1 the function of the input valve, valve-A, and the transport Tr2 the function of the discharge valve, valve-D. So1 represents the source of material and Si1 the system that consumes the material.

Deriving planning knowledge from MFM

In [3], Larsen applied the MFM methodology to acquire specific knowledge for generating a sequence of actions to start up a boiler power plant. According to Larsen, the process of developing a start-up plan for a plant that is

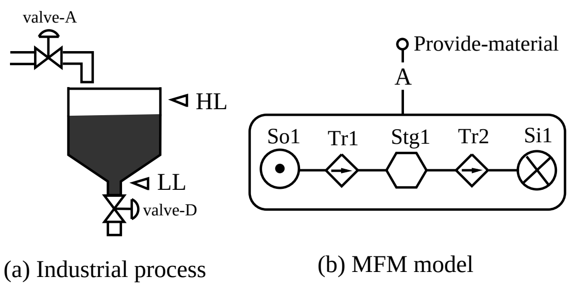

Fig. 3. (a) Example of a simple industrial process. (b) Typical MFM model

modeled by MFM may be compared to the problem of *establishing* each function in the model.

A function can be established when both the underlying neighboring functions in the model are in states as intended by the designer and contribute to achieving the operational goals. However, establishing a function is a quite complicated task and can not be done in one step. Before establishing a function, it must be *enabled.* A function can be enabled when it is ready to be integrated with other functions. This means that the function is ready to provide an useful behavior but is not yet working.

Based on this concept, we have developed a flexible approach to building a planning domain composed of operators to enable and establish the MFM functions. The generality of these operators allows their application to other engineering systems modeled by MFM. The strength of our approach results from the combination of the flexibility in building the domain using MFM and the rich representational framework provided by our planner.

Figure 4 shows a generic representation of a operator to establish an MFM transport function. To set the appropriate preconditions for a specific transport function, we use the Prodigy capabilities of constraining planning variables with functions. These functions are shown on the right side of Figure 4. This table shows the values returned by the functions when a transport function is passed as an argument, corresponding to the industrial process and the MFM model shown in Figure 3.

```
Operator  establish -transp <tr>
Var:<tr> Transp
      <so> Source : get-source
      <stg> Storage : get-storage
      <bal> Balance : get-balance
      <st> Process-state : get-state
      <act> User-task : get-act
Pre:
   (enabled-transp <tr>)          } topology
   (established-source <so>)      } dependent
   (established-storage <stg>)    }
   (established-bal <bal>)        }
   (fitness-pre <st>) ——> Process state
   (activity <act>) ——> User-task
 Add:
      (established-transp <tr>)
```

Constraint function	Argument	
	Tr1	Tr2
get-source	So1	NA
get-storage	NA	Stg1
get-bal	NA	NA
get-state	tank-not-full	NA
get-act	valve-A-open	valve-D-open

OBS: NA - not applicable

Fig. 4. Example of a generic operator to establish an MFM transport function

We have defined three types of precondition for mapping knowledge from an MFM model: (a) preconditions that can be derived from the model topology, preconditions to supports the basic behavior of a MFM function, called process state preconditions and preconditions associated to the human operator activities, called user task preconditions. The next section describes our general approach to acquire this kind of knowledge of any engineering system modeled by MFM.

4. General framework to acquire a planning model

We have successfully implemented a flexible approach to acquire a planning model of industrial processes based on two different frameworks, one related to the modeling system and the other related to the planning system. The integrated framework is shown in Figure 5. The modeling system provides an intermediate step to acquire the planning knowledge. However, we assume that the model exists already. This means that someone have identified the function and goals according to the method presented in [8]. We have defined a model representation for planning purposes based on the conditions required to enable and establish the MFM functions. The conditions for the existence of each function are organized in three structures (process states, topological, and user tasks) that will be mapped to the planning operator through the mapping functions. This representation provides a means to acquire the planning knowledge into more manageable entities, using especially the means-end and whole-part decomposition of MFM. We use a hierarchical approach to acquire the specific knowledge which is associated with each MFM function.

The planning system generates solutions according to the several levels of abstraction used to acquire the specific knowledge. Our planning domain does not encode the planning knowledge for a particular domain directly in the planning representation. Instead we have implemented a planning domain that is tightly coupled to the MFM functions, so that part of the planning domain can be reusable to other applications modeled by MFM. This domain, called MFM-based planning domain, is specified as a set of generic operators and a set of specific operators. The generic operators are composed of enable and establish operators. The specific operators are defined according to a domain; because they are domain dependent, our MFM-based planning domain can not be totally reusable to other applications. Therefore, a modification of the specific operators is necessary.

The focus of our work is to show that deep knowledge may be captured and included in the MFM knowledge representation in order to guide the searching process to come up with a sequence of enabling and establishing the MFM functions. From that sequence, we can find what the operators are needed to change a system state and in what order they must be executed.

Modeling system

The modeling system provides an intermediate step to acquire planning knowledge while maintaining an easy way to add or modify knowledge of each MFM function. Furthermore, this planning knowledge is represented in a way that can be easily readable, because it is captured

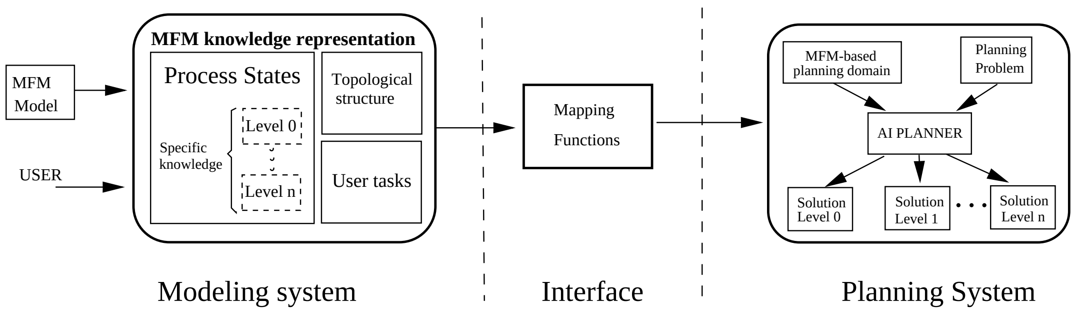

Fig. 5. Integrated framework to derive a planning model based on MFM

based on the MFM functions. To capture the knowledge from MFM for planning purposes, we must represent the model in a way that enables an AI planner to reason. Therefore, we translate the model into an internal representation, called MFM knowledge representation, that captures the specific knowledge of the domain (process states structure), the model topology (topological structure), and the activities of the human operator (user tasks structure).

Interface

The interface between the MFM and the planning system is realized by a set of mapping functions. These functions perform an important task in our approach; they maintain an *independence* between both frameworks and provide constraints on the planning variables.

Planning system

The planning system performs a reasoning process to solve a problem guided by the model along the means-end and whole-part decompositions. The planning system presents a solution which is composed of a sequence of actions based on the specific operators defined in the domain. These operators are ordered according to the sequence of enabling and establishing the MFM functions. As cited earlier, we use in our approach the Prodigy architecture. It is important to notice that a main factor responsible for the strength of our approach is the capability of constraining planning variables by arbitrary functions in an operator-based planner, such as the Prodigy planner. Other operator-based planners without this powerful feature could also benefit from our methods, though possibly obtaining more domain-specific operators.

6. Empirical results

Our approach has been successfully applied to two industrial case studies: power plants and chemical processes. In the following, we described our results to the power plant domain.

Power plant domain

We have developed an MFM-based planning domain to be applied in a boiler power plant [9]. The planning system generates a sequence of actions to start-up the plant based on high level procedures that will be executed by a

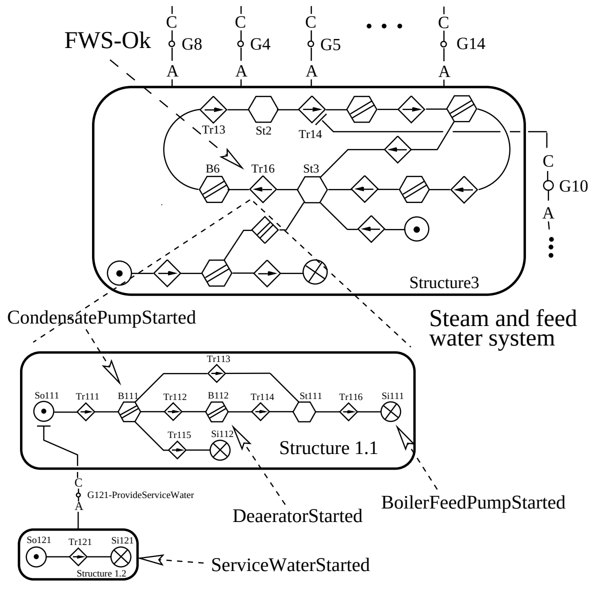

Fig. 6. Partial MFM of the steam and feed water system and the refinement of the transport function Tr16.

human operator. Hence, a solution is intended to lead the power plant from a state where no systems of the plant are working to an operating state. This view requires a solution based on a transition of states at a very large scale, from inoperative to fully operational. Therefore, the problem of controlling small changes and precisely controlling the subsystem states are ignored. Instead the focus is on the main functions of the plant and how to establish them. However, we show our approach to decompose the functions toward a lower level of abstraction in order to provide more details in the solution. We show one refinement in the feed water system to illustrate how our approach explores the MFM whole-part decomposition.

In [3], Larsen built an MFM model of a boiler power plant. The complete model that is composed of three MFM structures will not be shown here; however, Figure 6 shows the MFM model of the steam and the feed water system that will be used to illustrate our approach.

Structure 3 is responsible for capturing the functions of the water and steam cycles that are formed by a hot-well condenser, a economizer, a boiler, and the turbines. This structure guides the planner during the search process. The complete plan generated is too long to be presented here. Figure 7 shows the simplified plan composed of a sequence of procedures to startup the plant.

```
1. exec-proc put-plant-aux-in-service normal
2. exec-proc start-circulating-water-system normal
3. exec-proc start-feed-water-system normal
4. exec-proc start-boiler-water-system economizer-ok
5. exec-proc start-air-fuel-system normal
6. exec-proc start-steam-system normal
7. exec-proc turbine-generator-synchronization normal
```

Fig. 7. Sequence of procedures to startup a boiler power plant

We included in this plan an additional parameter to provide alternative solutions. This is the case of the boiler water system (item 3). Two solutions can be provided to the boiler water system. It can be started with the economizer present (as illustrated) or not. Whether or not the economizer will be present depends on the current state. If the economizer is specified in the current state, then it will be included in the solution. If it is not specified, it will be ignored.

Planning refinement: The above solution proposes an operator like "exec-proc start-feed-water-system" as a single action, whereas the procedure of start-up the feed water system of a boiler power plant is a quite complex task and may require more details. In order to present more details the MFM whole-part decomposition can be used. As an example, the feed water system is decomposed as illustrated in Figure 6.

We have associated with the transport Tr16 one process state precondition (FWS-Ok). If this precondition is false, then a subgoal process starts. To guide this subgoal process one lower level decomposition is defined. Tr16 is decomposed in two MFM structures. These structures describe a lower level of detail of the feed water system. The planner searches for a sequence of enabling and establishing the MFM functions defined. During this search new human activities are defined as illustrated in figure. According to this lower level of abstraction, the planning system generates the solution shown in Figure 8.

```
1. exec-proc put-plant-aux-in-service normal
2. exec-proc start-circulating-water-system normal
3.1. exec-proc start-service-water normal
3.2. exec-proc turn-on-condensate-pump normal
3.3. exec-proc start-vacuum-system normal
3.4. exec-proc turn-on-boiler-feed-pump normal
4. exec-proc start-boiler-water-system economizer-ok
5. exec-proc start-air-fuel-system normal
6. exec-proc start-steam-system normal
7. exec-proc turbine-generator-synchronization normal
```

Fig. 8. Solution to the startup problem considering the refinement of the feed water system

This plan shows that the feed water system was decomposed into four activities. These activities were ordered according to the sequence required to enable and establish the MFM functions.

Chemical process domain

To show the flexibility of our approach we have developed an MFM-based planning domain to generate operating procedures of a chemical process [2] . The power plant domain has been modified to support the specific activities of purge operations. In this domain, we address the problem of mixing constraints which commonly arise in chemical process planning problems.

6. Conclusion

We have proposed one alternative approach to give support to the human operator's activities in a control room of a supervisory system based on the AI planning techniques. In proposing this approach, we have addressed an important problem when applying AI planning techniques in real world applications - the planning knowledge acquisition.

We have defined a framework to acquire planning knowledge based on a modeling system on a planning system. The modeling system consists of a functional model, called multilevel flow modeling (MFM), and its representation. The model topology and the domain-specific knowledge are captured by the MFM knowledge representation and transferred to the planning operators through the mapping functions that assure independence between the planning system and the functional model.

References

[1] A. Tate, B. Drabble, and R. B. Kirby, "O-plan2: an open archichecture for command, planning and control," in *Intelligent Scheduling*, M. Fox and M. Zweben, Eds. 1994, Morgan Kaufmann.

[2] L. E. de Souza and M. Veloso, "Flexible approach to acquire ai planning knowledge of industrial processes," in *To appear at the 1st Annual International Conference on Industrial Engineering Applications and Practice*, December 1996.

[3] M. N. Larsen, *Deriving action sequences for start-up using multilevel flow models*, Ph.D. thesis, Technical University of Denmark, 1993.

[4] David Wilkins, *Practical Planning*, Morgan Kaufmann, Palo Alto, CA, 1988.

[5] M. Veloso, J. Carbonell, M. A. Pérez, D. Borrajo, E. Fink, and J. Blythe, "Integrating planning and learning: The PRODIGY architecture," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 7, no. 1, pp. 81–120, 1995.

[6] J. Blythe, M. Veloso, and L. E. de Souza, "The prodigy user interface," in *Artigo submetido ao SBAI-96*, 1995.

[7] M. Lind, "Modeling goals and functions of complex industrial plants," *Applied Artificial Intelligence*, vol. 8, pp. 259–283, 1994.

[8] M. Lind, "Representing goals and functions of complex systems," Tech. Rep. 90-D-381, Institute of Automatic Control Systems, Technical University of Denmark, November 1990.

[9] L. E. de Souza and M. Veloso, "Flexible planning representation of a boiler power plant," Tech. Rep. CMU-CS-95, Computer Science Department, Carnegie Mellon University, December 1995.