Flexible Planning Knowledge Acquisition for Industrial Processes

Luiz Edival de Souza[*]
Departamento de Eletrônica
Escola Federal de Engenharia de Itajubá
37500-000 Itajubá - MG, Brazil
edival@iee.efei.br

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

ABSTRACT

The acquisition of planning knowledge for industrial processes 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. In this paper, we report on our work on flexibly representing a functional model of a chemical process in an Artificial Intelligence (AI) planning system. We show the generality and flexibility of our new planning operator representation. Operators are designed for general functions which can be defined for hierarchical levels of functional detail. We present how the planning domain can be used with little or no modification, depending on the level of abstraction, to other engineering systems. This planning domain is fully implemented in a domain-independent AI planning system.

KEYWORDS: Planning system, knowledge acquisition, artificial intelligence, industrial process.

1 Introduction

Developing AI planning systems involves representing the domain actions corresponding to the operation procedures of the industrial processes. This knowledge acquisition process has been shown to be particularly hard in terms of obtaining robust task knowledge from human experts. Improving knowledge acquisition is an area of great interest for AI researchers and practitioners due of the well-recognized knowledge acquisition bottleneck in the task of automating or support human tasks. The knowledge acquisition in AI planning is more difficult than ordinary knowledge acquisition because of the complexity of the representation.

Several AI planning researchers, present authors included, envision the application of their planning systems to real 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 [3], [2], [7].

In this paper, we report on our work on representing the functional model of a chemical process in a domain-independent planning system. Lind introduced a multi-level flow modeling (MFM) for representing the functionality of engineering systems [5]. This model constitutes a significant advance in abstracting the high-level functional description of a system from the low-level technical implementation details. In [4], Larsen showed how the MFM of a boiler power plant could be translated into a specific set of data structures. We build upon this work and extend it to a domain-independent planning framework.

The paper is organized as follows: First, we overview briefly the MFM concepts and identify the limitations of Larsen's data structures. Second, we present our general model of knowledge acquisition of an MFM-based AI

[*]This work was done while the first was a visiting researcher at Carnegie Mellon University with a grant from CNPq, Brasília, DF. This research is sponsored in part by the Wright Laboratory, USAF, and the Advanced Research Projects Agency (ARPA) under grant number F33615-93-1-1330. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Wright Laboratory or the U. S. Government.

planning domain. We concretely present how the domain is a set of flexible domain-independent planning operators. Third, we show concrete examples and results, including a simple trace of part of a generated plan. Finally, we draw conclusions on this work and discuss our current research directions.

2 Outline of the Multi-level Flow Model

The MFM captures the functionality of any man-made engineering system. The MFM uses the means-ends and whole-part decompositions to represent a system by the goals of its designer or user, by its functions, and by its physical components. The goals are derived directly from the assumption that a system has specific purposes. For example, one of the goals of an air conditioner is to keep the temperature of a room within comfortable limits. The functions represent the intentions of the designer about the purpose of the components or of the sub-system. The functions are the means by which the goals are achieved. The MFM defines a set of mass and energy functions to describe the functional structure of a system. These functions are related in complementary pairs, namely source–sink, storage–balance, and transport–barrier.

Figure 1 presents the main symbols used to represent mass and energy functions. There is a set of relations which is used to describe the relationship between functions and goals. The MFM represents the functional structure of a system by a set of mass and energy flow structures on multiple levels of abstraction. These flow structures are inter-related by an *achieve relation* and by a *condition relation*.

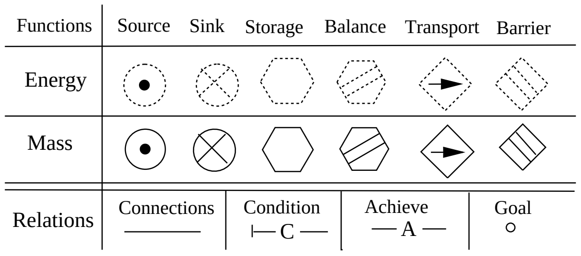

Figure 1: Graphic Symbols used by MFM.

The chemical process shown in Figure 2(a) will be used to illustrate an application of the modeling methodology. The main objective of this process is to produce a third product through of a reaction of two chemical products. The reaction takes place inside the reactor tank that provides a cooling jacket. The desired product can be removed through the outlet mixture pipe and valve V-M. The reaction produces energy that must be controlled by circulation of water in the cooling jacket. This process can be decomposed into three MFM structures.

Structure 1 represents the raw material flow into and out of a reactor tank. As can be seen, the transport functions, Tr11 and Tr12, represent the functionalities of valve V-A and V-B. The reactor tank is represented by the storage function Stg1 that is conditioned by the goal G2-ProvideSafeReaction. The transport Tr13 and the sink Si11 represent the outlet valve of the tank.

Structure 2 represents the energy flow between the mixture and the cooling water. The energy produced by the reaction is transported by Tr21. The balance function splits the energy flow into two flows, one going to the cooling water and one that is absorbed by the atmosphere. The Tr23 function represents the transport of energy going to the cooling water and is conditioned by the goal G1.

Structure 3 represents the functional structure of the cooling system.

Clearly the mapping between an MFM and the corresponding engineering system is not straightforward and is not an one-to-one mapping. On the other hand, the level at which the MFM describes the process allows reasoning about the high-level function of the system.

In [4], Larsen applied the MFM theory to acquire specific knowledge for generating a sequence of actions to start up a power plant. According to Larsen, the process of developing a start up plan for a plant that is modeled by an MFM may be compared to the problem of *establishing* the functions of the plant as described by the model.

A function can be established when both the underlying components (or subsystems) of the plant and the process are in states as intended by the designer and contribute to achieving the operational goals. However, the components or the subsystems may play several roles in a system; therefore establishing a function is a quite complicated task

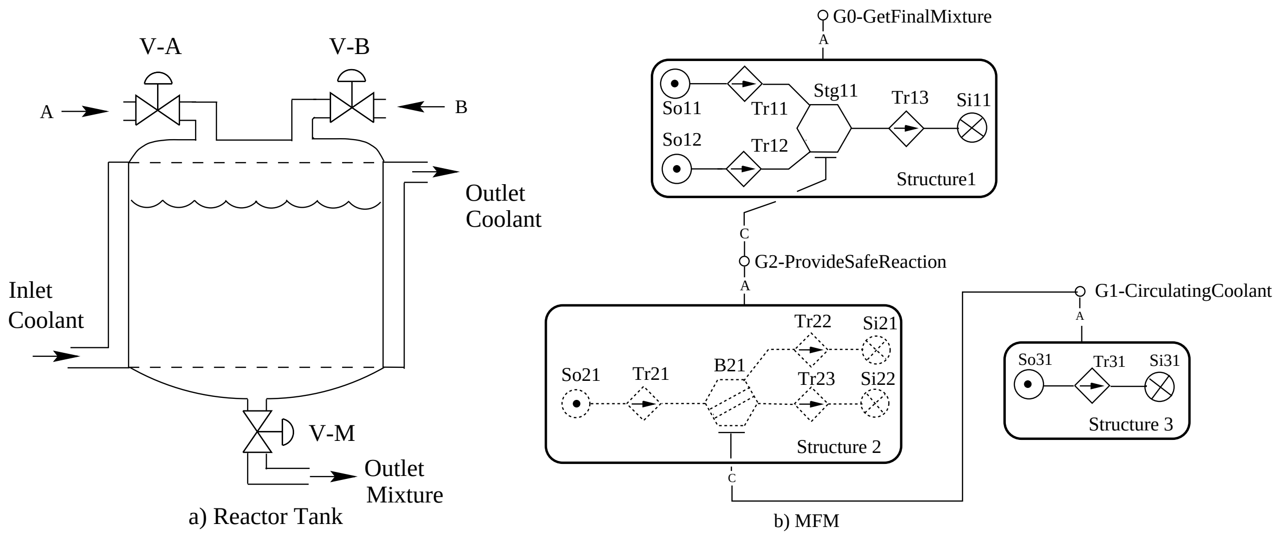

Figure 2: Example of a chemical process and the corresponding MFM model.

and can not be done in one step. Before establishing a function, it must be *enabled*. For example, one tank that is performing a storage function can receive material (if it is not full, the discharge valve is closed, etc.) but can not provide access to its contents until the minimum level is reached. In other words, it is enabled to receive material but is not yet established.

Larsen's work represents the first attempt of using the MFM for planning purposes. The methodology suffers some drawbacks, however:

- The knowledge acquired is only for an application in the start-up of systems. The inverse actions for enabling and establishing functions are not considered.
- The proposed system is domain-dependent and the proposed methodology only considers solutions based on a linear planning approach.
- For each MFM function defined in a model, two structures are defined. This approach can result in a complex knowledge database for a wide scale system.

The next section describes our approach and how these disadvantages can be overcome.

3 Knowledge Acquisition of a Planning Domain

Our approach to mapping the MFM functions to a planning representation owes much to the previous work developed by Larsen. We have defined one approach more suitable to applications in domain independent planning. We extended the establish and enable tasks to a more flexible and powerful structure embedded in planning operators. The strengths of our approach are based on flexibility in building the domain using the MFM and the complex resources provided by our planner.

We have successfully implemented a flexible approach to building a planning domain based on two different frameworks, one related to the MFM and the other related to the planning system. The integrated framework is shown in Figure 3. The modeling system defines a structure, called MFM knowledge representation, to acquire the planning knowledge into more manageable entities, using especially the means-ends and whole-part decomposition of MFM. We use a hierarchical approach to acquire the specific knowledge that is associated with each MFM function defined in the model. The planning system defines a planning domain, called MFM-based planning domain, that is composed of a set of generic and specific operators. These operators contain variables that will be instantiated by mapping functions according to the MFM knowledge representation.

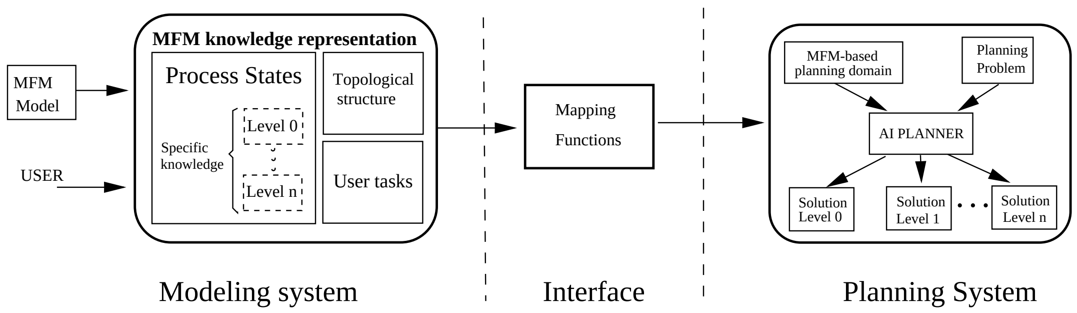

Figure 3: Modeling and AI Planning Systems.

3.1 Modeling System

To capture the knowledge from the MFM for planning purposes, we translate the MFM functions to an internal representation that captures the topology, the specific knowledge of the domain, and the human activities. Therefore, we defined the following structures:

- The topological structure. This structure describes the physical topology of the model. For each function of the model, we have to specify the neighboring functions in downstream and upstream flow.
- The process states structure. This structure captures the specific knowledge of the domain. These specific knowledge represents the conditions needed to enable and establish the MFM functions. The process states can be represented in an hierarchical approach which allows the planner to reason about refinements needed to achieve a solution at different levels of abstraction. For example, at a higher level of abstraction one function may have a precondition that is true in the initial state, this means that no extra actions is taken by the planner. But, if at lower level of abstraction the precondition is not true then the planner starts a subgoal process to search for actions that make this precondition true. This subgoal process will provide more details about how to enable or establish a function. This process is the implementation of the MFM whole-part decomposition.
- The user tasks structures. The objective of this structure is to capture the knowledge that express the activities that will be executed by the human operator. These knowledge must be acquired when an external action is required to enable or establish an MFM function.

3.2 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 variables of the preconditions in the planning operator.

3.3 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 by a sequence of actions based on the operators defined in the domain. These operators are ordered and instantiated according to the sequence of enabling and establishing the MFM functions.

The planning required to find a solution will proceed hierarchically: one solution can be proposed at a high level of abstraction, i.e., an operation like “execute-procedure start-reaction” may be proposed as a single operation, whereas the task of starting the reaction process in a reactor tank is a quite complex task and may require more detailed subtasks. In the case of refining that particular operation, one level in the hierarchy may be defined in which one or more MFM functions has a precondition that is not true. This precondition then becomes a subgoal to the planner. In the subgoaling process, the details for starting the chemical reaction can be specified and more information can be presented in the solution.

The planning problem describes the initial state and a goal state to be satisfied. For instance, when applying planning to a continuous process, the initial state contains the temperature values, pressures, flow-rates, valve states, etc.; and the final state contains the state variable values at the steady running state.

The planning domain specifies the legal actions that can be used to perform changes in the process states. These actions are represented in terms of operators with preconditions and effects. We have defined a flexible structure for the planning operators to enable and establish the MFM. We discuss this structure in the next section.

4 Implementation using the Prodigy Planner

Our implementation is based on the Prodigy architecture, a domain-independent planning and learning system that provides a base planning module and many capabilities implemented as integrated modules [6]. To illustrate our implementation, the example shown in Figure 2(a) will be used. We will detail the operator to enable a function, called *enable operator*, and the operator to establish a function, called *establish operator*, as shown in Figure 4.

Operator	Variable		Preconditions
	Id	Function	
enable	<func>	-	
	<f-d1>	get-function-e	enabled <f-d1>
effects:	<f-d2>	get-function-e	enabled <f-d2>
Add the state	<goal>	get-goal-e	MFM-pre <goal> <func>
enabled <func>	<state>	get-state-e	structural-pre <state> <func>
	<act>	get-activity-e	activity <act> <func>
establish	<func>	-	enabled <func>
	<f-u1>	get-function	established <f-u1>
Effects:	<f-u2>	get-function	established <f-u2>
Add the state	<f-u3>	get-function	established <f-u3>
established <func>	<state>	get-state	fitness-pre <state>
	<act>	get-activity	activity <act> <func>

Function	Argument			
	Tr11	Tr12	Tr13	Stg11
get-function-e	Stg11	Stg11	Si11	-
get-goal-e	-	-	-	G2
get-state-e	-	-	-	ready
get-activity-e	-	-	-	-
get-function	So11	So12	Stg11	Tr11/Tr12
get-goal	-	-	-	-
get-state	-	-	-	-
get-activity	open	open	open	-

Figure 4: General structure of the operators to enable and establish an MFM function.

This representation provides an abstraction at the function level, i.e, there is only one enable operator and only one establish operator that can be applied to any MFM function defined in the model. To set the appropriate preconditions, we use the Prodigy capabilities of constraining planning variables with functions. These functions are shown on the right side of Figure 4. This figure shows the values returned by the functions when a transport function or a storage function is passed as an argument.

As an example, let us suppose that our current goal is to have “enable Stg11.” This precondition is achieved by setting the corresponding enable operator. We can see in Figure 4 that there are two enabled-downstream preconditions. In general, a storage function do not needed this type of precondition. The get-function-e returns a special binding to the variables that set true these preconditions. Storage Stg11 is conditioned by the goal G2-ProvideSafeReaction, so the function get-goal-e returns G2 to bound the variable <goal>. The next precondition, structural-pre, captures the promptness of the reactor tank. In this case, the get-state-e function returns the value “ready” to represent this condition. There is no activity associated with the storage Stg11, the function get-activity-e returns a special value to set true the corresponding precondition.

5 Results

Considering the chemical process in Figure 2(a), we have applied our approach to several different scenarios. As an example, Figure 5(b) shows a solution to the problem specified in Figure 5(a). We have demonstrated the flexibility of our approach through a modification in the chemical process. A third product is added implying in a modification in the model topology. Following our approach, we included this modification in the MFM knowledge representation. Once these changes have been performed, our planning system can generate the procedures to route the products with no modification in the mapping functions or in the planning domain.

In attempting to use our approach for the synthesis of operating procedures of a different application, we found that a minimal set of changes must be made in the mapping functions and planning domain. The main reason for these

```
INITIAL STATE:              GOAL:
  open-valve v-m Stg11        established Stg11 AB
  close-valve v-a Tr11
  close-valve v-b Tr12
```

(a)

```
Achieved top-level goals.            <establish tr21>
Solution:                            <establish b21>
<exec-proc close v-m stg11>          <establish tr23>
<enable si22>                        <enable stg11>
<enable tr23>                        <enable tr11>
<enable si21>                        <enable so11>
<enable tr22>                        <establish so11>
<enable si31>                        <exec-proc open v-a tr11>
<enable tr31>                        <establish tr11>
<enable so31>                        <enable tr12>
<establish so31>                     <enable so12>
<establish tr31>                     <establish so12>
<establish si31>                     <exec-proc open v-b tr12>
<enable b21>                         <establish tr12>
<enable tr21>                        <establish stg11>
<enable so21>
<establish so21>                     #<PRODIGY result: T, 13.7 secs, 196 nodes, 1 sol>
```

(b)

Figure 5: (a) Problem specification. (b) Complete plan generated by the planner.

changes is due to the specific knowledge that must be acquired and stored in the MFM knowledge representation. We have developed a planning domain to generate the sequence of actions to start up a power plant [1]. Based on the same approach described in this paper, our system generates the start-up plan of a boiler power plant at a high level of abstraction.

6 Conclusion

We have presented a flexible approach to acquire knowledge about the planning domain of industrial processes from their corresponding functional model. We have shown the flexibility of our approach through two simple chemical process examples, where the same domain is used. The domain-specific knowledge is captured by the MFM knowledge representation and transferred to the planning operators through the mapping functions which assure independence between the planning system and the functional model. The generality of our approach allows us to apply the same concepts to other engineering systems, such as a power plant. Previous works in this area showed some drawbacks to which our approach presents some improvement.

References

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

[2] R. H. Fusillo and G. J. Powers. Computer-aided planning of purge operations. *AIChE Journal*, 34(4):558–566, 1990.

[3] R. Lakshmanan and G. Stephanopoulos. Synthesis of operating procedures for complete chemical plants - i. hierarchical, structured modelling for nonlinear planning. *Comp. Chemical Engineering*, 12:985–1002, 1988.

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

[5] M. Lind. Modeling goals and functions of complex industrial plants. *Applied Artificial Intelligence*, 8:259–283, 1994.

[6] 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*, 7(1):81–120, 1995.

[7] David Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann, 1988.