A New Action Description Scheme for Informal Reasoning

Umair Rafique and Shell Ying Huang
School of Computer Engineering, Nanyang Technological University, Singapore

Abstract—Reasoning about Actions has an important place in the development of Intelligent Agents. While reasoning about Actions, Action Representation plays an important role. General practice is to use Preconditions and Post-effects for defining an Action. We believe that for an agent that assists humans in making decisions for daily life in a Smart Home environment, Preconditions are not adequate to represent the complexity due to the huge number of different situations resulting from various factors affecting a decision. We propose an extension to this basic idea by introducing the concept of Hindering and Supporting factors for Actions and argue that this can describe many Informal Reasoning situations more succinctly than traditional Action description schemes. We present the reasoning algorithm based on this new Action Description and show with examples how this new scheme helps in reaching more appropriate decisions in a Smart Home environment and hence improving the quality of living for its inhabitants.

Keywords: Action Description, Smart Homes, Informal Reasoning

1. Introduction

The term Smart Home is used to describe a hi-tech home environment where various technologies, including AI, are used to improve quality of living for its inhabitants. Smart home can be a viable solution for providing assistance to the elderly and the disabled for their Activities of Daily Living (ADL). It opens the way for a wide range of research directions for example, Behavior Reasoning [1], Plan Recognition [2], Planning [3] and Delegative Assistance [4]. Another direction within this domain can be development of agents who can assist humans in daily life Decision Making and Reasoning. Examples of such decisions can be “how to dry wet cloths?” or “How to go to market?” with suggestions by the agent like “Use cloth dryer since humidity is high outside” and “Walk to the market since weather is very pleasant, market is not far and cost is lower than taking a taxi” respectively. Development of such agents can be useful in many ways. Firstly such assistance in reasoning can be of great help for the elderly and the disabled by making them more independent in their ADL. Secondly, if various tasks are done in the most appropriate way possible, it can have some economic benefits. And lastly with such routine matters taken care of by the artificial agent, the Smart Home inhabitants can enjoy a more relaxed and comfortable living and focus more on the activities they enjoy. It is quite established in the AI research community that such Informal or Everyday reasoning [5] [6] is not trivial and involves reasoning about many complex common sense issues. Common Sense Problems Page 1 lists a number of benchmark common sense reasoning problems which researchers in this area have tried to solve over time (e.g. [7]). We believe that Smart Homes can prove to be a useful test bed for Informal Reasoning research due to the complex daily life reasoning situations.

In this paper we focus on reasoning about Actions for daily life Decision Making in the context of Smart Homes. Existing Action Description schemes (See for example [8] [9] [10]) support reasoning about Actions which is based on the concept of State Transition System [11]. The execution of an Action takes the system from one state to another. Each state represents a certain setting of the environment in the reasoning agent’s view at a particular instance of time and Actions bring changes to this view either by changing the environment or by changing agent’s beliefs. This idea of a state transition system is also used in Automated Planning (See for example [12]) where Actions are referred to as Operators. These Operators provide the steps to achieve goal states when executed in a specific sequence where this sequence is determined by the planning algorithm. Planning approaches in BDI [13] based systems (e.g. [14]) are broadly similar too. This idea of Actions leading to transitions can also be found in the area of Cognitive Architectures (CA) (where Actions are referred to as skills [15]), specifically in symbolic CA (e.g. SOAR [16]) which are generally based on Production Systems [17].

Although all these systems are considerably different and very rich in their own setting, they have quite similar basic representations for Actions. Generally, an Action (Operator in Automated Planning terms, Plan in BDI terms or Skill in CA terms) is described by an identifier (name), some Preconditions which tell when this Action can be executed and some Post-effects, which are changes to the environment (or to agent’s own beliefs) brought by execution of the Action. These post-effects and preconditions can be represented using logic formulae and a change in state is represented by adding or deleting literals from a list representing the current state [12]. Since Actions bring changes to the state of the system and in the new state different Actions can be applied (i.e. the ones whose preconditions match this new state), this results in the formation of the well known “State

1http://www-formal.stanford.edu/~leora/commonsense/
Space”. Finding a plan (i.e. ordered set of Actions) that can take the system from an initial state \( S_0 \) to a final goal state \( S_G \) then becomes the task of finding a path from \( S_0 \) to \( S_G \) in the state space graph. This solution works well for domains which can naturally be represented by a “not very complex” state space (like for example the type of problems used in Automated Planning Competitions). But for Informal reasoning domains, like Smart Homes, this method of search in state space, with logic formulae to represent states, does not seem appropriate due to the daunting complexity involved in representing informal concepts [18]. Attempts to formalize Common Sense Reasoning problems using logic, like e.g. [7], show how complex such formalizations can be even for small problems like cracking an egg and putting its contents into a bowl.\(^2\)

Our approach is to identify the underlying patterns for reasoning situations in the context of Smart Homes and then to develop an Action description scheme that supports reasoning for these patterns. These patterns involve certain aspects which cannot be handled easily using existing Action Description mechanisms. We observe that there are certain factors which are not part of preconditions of an Action but these, if present, can make execution of the Action easier or more desirable. For example if the agent has to make a decision about how to dry wet cloths, the presence of sunlight makes the task easier, although it is not a requirement. Also there can be identified factors for Actions whose absence is not a precondition but these, if present, can hinder the execution of the Action. For example if the agent is to decide how to go to market (by walk, by bus, by car), factors like rain and hot sun although are not required to be absent in case the agent chooses to walk, these can make walking less desirable and hence affect the final choice the agent may make. We call such factors Supporting Factors (SF) and Hindering Factors (HF), respectively. It should be noted that whether a factor supports or hinders an Action may depend on the particular situation the agent is in. For example very light rain is not a hindering factor if an umbrella is available for going to market. For drying cloths, sunlight might be supportive as long as it is not so bright that it can tan the cloths.

Representing such factors with all these varying roles for different situations using logic will result in an enormous number of formulae. If there are \( n \) such factors for an Action with \( m \) distinct possible values for each factor, then theoretically \( m^n \) different situations are possible. Using preconditions and posteffects alone will either result in long and complicated specification of preconditions or inadequate description that constrains the flexibility of the agent in making decisions. In this paper we propose an Action description scheme for this informal reasoning setting of a Smart Home. Such an action description scheme allows for the factors affecting a decision to be represented without adding much complexity. We show examples where using this scheme an artificial agent can handle the complexity of various factors affecting Actions and can make appropriate decisions according to a given situation.

The basic idea of this approach was proposed in [19]. In this paper we elaborate upon the details of our framework. In Section 2, we identify information/concepts that are commonly considered while reasoning about Actions in daily life. In Section 3 we give a brief overview of our framework. In Section 4 we provide the detailed description of knowledge structures we have proposed. In Section 5 we present the reasoning algorithm using our proposed scheme and use a few sample scenarios to show how appropriate decisions may be made for different situations. We end by providing some future directions in which we are continuing our work.

2. Action Description for Informal Reasoning

Representation of Action in our framework is in the form of Activity. An Activity essentially is an Action in the traditional sense (or an English language Transitive Verb) combined with the entity it is applied on (i.e. an Object in English grammar terms) e.g. “Clean-Room”, “Move-Table”, “Wash-Cloths”, “Go-Office”. For the representation of real life entities (i.e. general real life objects or Nouns in English language) we also use a specific structure and call them Objects. We first identify information/concepts that are commonly considered in the reasoning about actions in daily life.

Basic Requirements: An Activity may require certain things to be available for its execution. For example “Play-Tennis” requires Racket, Balls, Booking Court etc. Without these, the execution of the Activity is not possible. We call such things Basic Requirements or BR of the Activity. BR can be other Objects, like Racket and Ball in this case, or other Activities which need to be completed before the execution of this Activity can be started, which in this case is “Book-Court”. If any of the Object BR is not available, other Activities can be launched to make it available. For example Activity “Buy-Racket” can be started to make the racket available in case it is not. Storing BR in this way automatically handles the order in which these things are to be made available. In this case if an Activity requires certain Objects to be available or other Activities to be executed before it can start its execution, these can be made its BR. For any Activity \( Act \) in its BR, if some other Object or Activity is required before \( Act \) can be started, these can be made \( Act \)’s BR and so on.

Basic Obstacles: An Activity can have a few things which, if present, will render the execution of the Activity as impossible. We call such things Basic Obstacles or BO.

of the Activity. For example “Play-Tennis” cannot be done if it is raining. BO for an Activity are other Objects only.

**Alternative Ways:** An Activity can be done in more than one ways. For example “Go-Office” can be done as “Take-Bus”, “Take-Car”, “Do-Walk”. We refer to these different ways of doing an Activity as Alternative Ways or AW of the Activity. Storing AW for an activity leads to the hierarchical structure of skills where an Activity can be done in a number of ways, each of these ways can be done in other ways and so on. For some Activities there is no such AW in which case these will be declared as Primitive Activities.

**Factors:** For an Activity there can be a few factors which, if present, hinder the execution of the Activity. For example “Dry-Cloths” can be hindered by “Humidity”, “Read-Book” can be hindered by “Noise”. Such Objects whose presence hinders an Activity are what we call Hindering Factors or HF for that Activity. Also there can be a few such factors which, if present, support the execution of the Activity. These are the factors which are not required for the Activity’s execution but these, if present, would ease its execution. For example “Dry-Cloths” can be supported by the presence of “Wind” or “Sunlight”, although these are not required for this Activity. We call such factors Supporting Factors or SF for that Activity. Since an SF can become HF under certain conditions (e.g. strong wind is not supportive for “Dry-Cloths” anymore) and vice versa, we refer to these only as “Factors”. Their role as HF or SF will be determined for every given situation.

**Post-Effects:** Execution of an Activity may bring some changes to the environment (or to agent’s own beliefs). We refer to these changes as Post-Effects or PE of the Activity. For example execution of the Activity “Buy-Racket” would result in decrease in “Money” and addition of “Racket” to agent’s possession.

**Cost:** An Activity has a Cost attached to it which is the cost of executing this Activity. It is the sum of the costs of providing all the Objects that are required for the execution of this Activity. It is not static over time and in different situations it will be different. For example if “Buy-Racket” is executed for “Play-Tennis”, the cost will be higher while for subsequent instances it would be lower since Racket would already be available.

We also identify the information related to objects that are considered in decision making as follows.

**Attributes:** An Object can be described by its features which we call Attributes of the Object. For example Water can have attribute “Temperature”, Wind can have Attribute “Intensity”. An Attribute can be either an English language Noun (as Temperature in case of Water) or an Adjective as Big in case of “Big-Room”. The Attributes of an Object can be distinguished into two types, “Spatial Attributes” which describe its spatial properties and “Property Attributes” which describe its other features. The whole set of attributes for an Object “describe” that Object. Objects present at an instance of time with specific values for their attributes altogether determine the state of the system at that instance of time.

**Hindering Factors:** An Object can be hindered by other Objects. That is, the presence of certain Objects in a situation can make the presence of some other Objects difficult. For example Clouds can hinder Sunlight. We call such factors Hindering Factors or HF for an Object.

**Saving Factors:** Harmful Effects of an Object, in a given situation, can be eliminated using other Objects. That is if an Object O is hindering some other Activity or is an HF for another Activity, the presence of these saving Objects for O can eliminate its effects. For example if Rain hinders “Do-Walk”, Umbrella can save from it. Such Objects which save from another Object’s effects are what we call the Saving Factors for the Object or SavF.

**Cost:** An Object can have certain cost attached to it. This is the cost of getting this Object. This is only applicable to the Objects which are (hypothetically) under the control of the artificial agent. For example cost for Rain is meaningless since the agent cannot make it available.

We have used these concepts for a new representation of Activities and Objects in a framework called SMART (Symbolic Memory for Abstract Reasoning and Thinking). In the next Section we give a brief description of our framework.

### 3. SMART Cognitive Architecture

SMART is a Cognitive Architecture (CA) with two different types of memories for storing knowledge, Long Term Memory (LTM) and Short Term Memory (STM), and a reasoning module, Central Executive (CE), which operates on these memories (these terms are inspired from literature in Cognitive Psychology). Although the notion of having STM and LTM seems similar to the usage of memories in other CAs like e.g. SOAR [16], Icarus [20] and ACT-R [21], the way these memories are used in SMART is quite different. Unlike [20], where skills and facts are explicitly separated in different memories, and [21], where such distinction is made using procedural and declarative knowledge, SMART does not have separate LTMs for skills and facts. Facts about the world are combined with knowledge about skills in LTM of SMART using Objects and Activities. With the interconnection of Objects and Activities a network is formed which we call SMART Net. Figure 1 shows the structure of SMART at the macro level.

STM is separated into two parts as Perceptual Memory (PM) and Working Memory (WM). PM contains the perceived states for different Objects from the environment. It is similar in purpose to Perceptual Buffer in [20]. WM stores the specific instances of LTM concepts for a given situation (which is perceived in PM). Although both LTM and WM store knowledge as per the structure of SMART Net, the difference is that LTM stores general concepts while WM
contains their specific instances. For example, LTM can store concept like “Rain is a Factor for going out” but whether it is hindering or not in a particular situation will be determined for that situation at runtime. This will include reasoning based on the information from PM about the intensity of the rain, its location etc. Once the role of Rain is determined, this instance will be kept in WM for further reasoning. CE is the main reasoning module and works with STM and LTM. The part of CE that reasons about activities is summarized in Algorithm 1.

**Algorithm 1: Working of CE**
1. foreach task T do
2.    Fetch knowledge from LTM about T to WM;
3.    In WM, create specific instances of LTM knowledge using PM information;
4.    Run the reasoning module for finding best possible way of performing T;

4. Building Blocks of SMART Net

Actions represented as Activities and real life entities represented as Objects are the main building blocks of SMART Net. Each Object and Activity is represented as a node with the name of the node identifying the entity. A node is connected with other nodes with each connection representing a special relationship and hence the network SMART Net is formed. The relationships between these nodes are based on the concepts presented in Section 2.

4.1 Object

An Object represents a real life entity (or in other words it has the same meaning as a Noun in English language) like e.g. Umbrella, Rain, Market, etc. An Object node will be identified with the name of the Noun it represents. The meaning of an Object within the context of SMART Net is achieved by its connections with other Objects (and Activities). Figure 2a shows a general Object node along with description of the links through which other Objects (hindering and saving) are connected to it.

Each Object is described using Attributes. An Attribute is a pair <AttributeID, Value> where AttributeID is the name of the Noun in case it is a Noun attribute while it is some unique identifier in case it is an Adjective. The reason of using an ID for Adjective Attributes is to differentiate between different Attributes although only the value of the Adjective can describe the Object as e.g. “Clean Floor”. The Value in an Adjective Attribute is the Adjective itself (e.g. Clean in case of “Clean Floor”) while in a Noun attribute, its value is some numerical measurement. For example <Temperature, 30> represents that the room temperature is 30 degrees. Cost for an Object is the cost of providing this Object. Hence the template for an Object is, <NounID, [Attributes], [SavF], [HF], Cost>.

**Example 1.** Object “Sunlight” in LTM can be represented as, <Sunlight, {(Intensity, NA), (Location, NA)}, [Shelter], [Clouds], NA> while its instance in WM for some given situation can be represented as, <Sunlight, {(Intensity, 7), (Location, Outdoor)}, [Shelter], [Clouds], NA>

4.2 Activity

An Activity is an Action (or an English language Transitive Verb) combined with the Noun (i.e. “Object” of this Action in English grammar terms) it is applied on. Hence an Activity is identified as an Action-Noun pair. “Play-Tennis”, “Move-Table”, “Read-Book”, “Go-Office” are a few examples of Activities. Figure 2b shows a general Activity node with details of all the connections such a node can have with other Objects and Activities.

**Fig. 2: Object and Activity Nodes**

We consider PE of an Activity as changes to the attributes (spatial or property) of other Objects caused by the execution of the Activity. For example PE for “Wash-Cloths” can be changes to two attributes for cloths as after this Activity they are “Wet” and “Clean”. PE of “Buy-Racket” can be that “Location” Attribute of “Racket” is changed (from shop to home) and “Amount” Attribute of “Money” has got a new lower value. PE is a tuple <Object, AttrID, newVal> where Object is the identifier of the Object whose Attribute’s value has been changed, AttrID is the ID of the Attribute and newVal is the new value it has got as the result of the execution of the Activity.
As we have seen, an Activity can have other Objects or Activities as its BR, BO, SF, AW and HF. Hence the Cost of executing an Activity “X” or \( \text{Cost}_X \) will be,

\[
\text{Cost}_X = \text{Cost}_{BR} + \text{Cost}_{SavF(BO)} + \text{Cost}_{SF} + \text{Cost}_{AW} + \text{Cost}_{SavF(HF)}
\]

where \( \text{Cost}_{BR} \) is the cost of providing Activity X’s BR, \( \text{Cost}_{SavF(BO)} \) is the cost of providing the SavF for its BO, \( \text{Cost}_{SF} \) is the cost of providing its chosen SF, \( \text{Cost}_{AW} \) is the cost of the chosen way to perform this Activity and \( \text{Cost}_{SavF(HF)} \) is the cost of providing SavF for its HF. If any of these sub-entities is an Activity then its cost has to be computed first in the computation of the cost of this main Activity. This means the cost for an Activity is calculated starting from the primitive Activities. Hence template for an Activity is \(<\text{Action-Noun}, \{BR\}, \{BO\}, \{F\}, \{AW\}, \{PE\}, \text{Cost}>\).

**Example 2.** Activity “Hang Dry-Cloths” in LTM can be represented as, \(<\text{Hang Dry-Cloths}, \{\text{Hanging String, Cloth Clips}\}, \{\text{Rain}\}, \{\text{Wind, Sunlight, Humidity}\}, \{\text{NA}\}, \{<\text{Cloths, Attr1, Dry}>\}, \text{Cost}>\). It would have the same representation in WM just that the Objects involved would get specific values for their Attributes according to the given situation.

### 4.3 Behavior Description Information Links

The hindering or supporting behavior of an Object for an Activity may not remain the same always and it can become more supportive, less supportive, hindering or neutral with changes in the environment. For example we may know that “Sunlight” is supportive for “Dry-Cloths” but if it is too bright, it may tan the cloths (hence hindering) and if it is too light, it may not have any supportive effect (hence neutral). This change in behavior depends upon the changes in values of the involved Objects’ attributes. For example in the cloth drying example above, it is ‘intensity’ attribute of “Sunlight” that determines its role for drying cloths; Hindering for high values, Neutral for low and Supportive for intermediate values. To capture this information within SMART Net, we use special links for connecting Factors to Activities. We call such a link “Behavior Determination Information Link” or BDIL. For connecting Factors to an Activity, the BDIL used will be denoted as \( \text{BDIL}_A \).

\( \text{BDIL}_A \) is a tuple \(<\text{Act, Obj, Attr, } f(\text{AttrVal})>\) where \( \text{Obj} \) is a Factor for \( \text{Act} \). \( \text{Attr} \) tells which Attribute of \( \text{Obj} \) this link describes. \( f(\text{AttrVal}) \) is a function of the values of \( \text{Attr} \) and it defines the role of \( \text{Obj} \) (Hindering, Supporting or Neutral) for \( \text{Act} \) at various values of \( \text{Attr} \). Figures 3a and 3b present examples of such a function. Value of \( \text{Attr} \) changes along the x-axis while value of the function tells the role of \( \text{Obj} \) for \( \text{Act} \). The point where transition in behavior occurs is what we call “Transition Point”. For the value of Support and Hindrance along y-axis, we use a fixed scale from -10 to 10 where -10 means maximum hindrance and +10 means maximum support. \( f(\text{AttrVal}) \) can take any shape depending on how the behavior of \( \text{Obj} \) changes for \( \text{Act} \) due to the values of \( \text{Attr} \).

This change in behavior because of the change in the involved Objects’ attribute values also affects Object-Object relationships and an Object saving from (or hindering) another Object may not do so always. For example we may know that Umbrella can save from Rain but if the rain is very heavy, using Umbrella is not very effective. For an Object-Object relationship where an Object \( O \) is Saving from (or Hindering) another Object \( O' \), the attribute determining the behavior of \( O \) may belong to \( O \) or \( O' \). In some cases two attributes, one belonging to \( O \) and the other to \( O' \), together may determine this relationship. As an example of the change in behavior due to an attribute of \( O' \), consider the fact “Umbrella can save from Rain”. Umbrella (of a fixed size) can only be used as long as ‘Intensity’ of rain is below some specific point. Hence the change in Rain’s attribute ‘Intensity’ determines the umbrella’s role. On the other hand for a fixed intensity of Rain, the ‘Size’ of Umbrella can determine whether it can be used as a saving Object or not. When taken together, Size of Umbrella and Intensity of Rain both determine the effectiveness of Umbrella for saving from Rain. This information is also stored in the links connecting these Objects. BDIL connecting SavF of an Object will be called \( \text{BDIL}_S \) and BDIL connecting HF of an Object will be called \( \text{BDIL}_H \).

For cases when a single attribute, belonging to either \( O \) or \( O' \), determines the change in behavior, \( \text{BDIL}_S \) and \( \text{BDIL}_H \) both can be represented by the tuple \(<\text{Obj}_1, \text{Obj}_2, \text{Attr}, \text{Val}, f(\text{AttrVal})>\). For \( \text{BDIL}_S \), \( \text{Obj}_2 \) is a Saving Object for \( \text{Obj}_1 \) while for \( \text{BDIL}_H \), it is a Hindering Object. \( \text{Attr} \) tells which attribute (can belong to either \( \text{Obj}_1 \) or \( \text{Obj}_2 \), is causing this change in behavior. \( O \) represents which Object \( \text{Attr} \) belongs to since both Objects can have some attribute with the same name. \( f(\text{AttrVal}) \) models the role of \( \text{Obj}_2 \) for \( \text{Obj}_1 \). For \( \text{BDIL}_S \), value of this function at a particular value of \( \text{Attr} \) tells whether \( \text{Obj}_2 \) is Saving or not for \( \text{Obj}_1 \) while for \( \text{BDIL}_H \) its value tells...
5. Reasoning in SMART

In the reasoning in SMART, spatial reasoning is essential since only those Objects which are at one place at one time will matter. For example for drying clothes by hanging, rain can be an obstacle if the hanging string is in the open while it will have no effect if the clothes are hanged in a covered place. In real life situations, such reasoning is not simple. Different Smart Homes can have different maps and for such reasoning the agent has to have a “mental model” tailored to that particular house. Here is not the place to discuss it in detail. For simplicity, we assume that all the relevant entities are at the same place. As a continuation to Algorithm 1, Algorithm 2 summarizes how a decision is made.
singled out Situation 4 even though Situations 2, 3, 4 and 5 are quite similar. Situation 6 presents a special case for “Hang Dry-Cloths” where “Shelter” will be used for Hanging Cloths since the Sunshine is too bright for the cloths. This set of examples demonstrates how various factors for an action, with varying behavior in different situations, can be easily handled using this Action Description Scheme and appropriate decisions can be made for each situation.

Table 1: Example Situations from PM

<table>
<thead>
<tr>
<th>Object</th>
<th>Situation 1</th>
<th>Situation 2</th>
<th>Situation 3</th>
<th>Situation 4</th>
<th>Situation 5</th>
<th>Situation 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electricity</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Rain Intensity</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Clouds</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>String</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Wind Intensity</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Sunlight Intensity</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Humidity Intensity</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

| Shelter Size       | Y 7 0       | Y 7 0       | Y 7 0       | Y 7 0       | Y 7 0       | Y 7 0       |

6. Conclusion and Future Work

We proposed an extension to the idea of Preconditions and Post-effects for an Action by introducing the concept of HF and SF. We showed that this can represent certain Informal Reasoning situations more succinctly than traditional representation schemes and allows for more flexibility in making decisions. We are continuing our work in two directions. First is developing an implemented prototype of our proposed framework to comprehensively test its usefulness in large number of situations. Second is to develop algorithms to facilitate learning \( f(AttrVal) \) functions in BDIL.

References