POMDPs in OpenMarkov and ProbModelXML

M. Arias[1] F. J. Díez[1] M. A. Palacios-Alonso[2] M. Yebra[1] J. Fernández[3]

[1]Dept. Artificial Intelligence. UNED. Juan del Rosal, 16. 28040 Madrid. Spain

[2]Computer Science Department. National Institute of Astrophysics, Optics and Electronics. Luis Enrique Erro 1. 72840 Tonantzintla. Mexico

[3]HC Energía. Plaza de la Gesta, 2. 33007 Oviedo. Spain

ABSTRACT

OpenMarkov is an open-source tool for editing and evaluating probabilistic graphical models, such as Bayesian networks, influence diagrams, MDPs, POMDPs, Dec-POMDPs, etc. ProbModelXML is a format for encoding probabilistic graphical models. In this paper we explain how to edit MDPs and POMDPs using OpenMarkov's graphical user interface, and how these models can be stored in ProbModelXML.

1. INTRODUCTION

Markov decision processes (MDPs) were developed around the mid-20th century as a tool for planning, more specifically, for solving multistage decision problems in which the outcomes are partly random and partly under the control of a decision maker [4]. The main limitation of those models was the assumption that the state of the system is always known with certainty, which is unrealistic in most cases. The relaxation of this assumption led to the emergence of partially observable Markov decision processes (POMDPs) [3], in which the state of the system is not directly observable, but there is a variable that correlates probabilistically with it.[1]

A probabilistic graphical model (PGM) consists of a probability distribution P defined on a set of variables $\mathbf{V}$ and a graph G such that each node in the graph represents one of the variables of $\mathbf{V}$ and the structure of the graph represents the probabilistic relations in P. Roughly speaking, we can say that in general each link in G represents a dependency in P and each link absent in G represents a relation of conditional independency in P. The exact relation between G and P depends on the type of PGM, mainly on whether the links in G are directed or undirected. The first types of PGMs were influence diagrams [16] and Bayesian networks [22].

Some types of PGMs have a dynamic version (in this context, "dynamic" is a synonym for "periodic"), in which each node represents a variable (a real-world property) in a particular instant of time, as shown in Figure 1. These dynamic PGMs generalize Markov models developed several decades earlier. Thus, dynamic Bayesian networks [8] generalize Markov chains and hidden Markov models [20]; factored MDPs [5, 6] and factored POMDPs [7] generalize flat (i.e., non-factored) MDPs and POMDPs, respectively.

Factored (PO)MDPs can model efficiently many problems that in practice could not be represented with flat models, and new algorithms developed in the last years are able to solve problems several orders of magnitude bigger than those affordable in the recent past [14, 23, 25].

In practice, the use of PGMs and (PO)MDPs requires two steps: building a model and doing inference on that model. There are many tools for building Bayesian networks and influence diagrams using a graphical user interface (GUI)—see Section 2.2—but to our knowledge only one of them, Netica, claims that it can be used to build (PO)MDPs; however it uses a non-standard representation, and is not open-source (see again Sec. 2.2). Therefore, the usual way of building a (PO)MDP is to generate an ASCII file using a text editor—or an XML editor, in its case—, which makes the process difficult, time-consuming, and prone to errors. In the field of robotics, where the practitioners of (PO)MDPs are usually computer scientists or engineers, this task is a burden but not an obstacle for the use of this type of models. On the contrary, it is extremely rare to find a health professional with the expertise and the patience necessaries to build a (PO)MDP using a text editor. For this reason, it might be very helpful to have a software tool with a friendly GUI for building (PO)MDPs, especially in some domains, such as medicine [12]. That tool should be open-source in order to allow researchers to modify and extend it.

We have mentioned in passing the need of a format for storing the models. Several formats have been proposed for PGMs, but only of them is able to encode Bayesian networks, influence diagrams, and factored (PO)MDPs: DNET, which is the native format of Netica. However, to our knowledge this format has never been used to build a (PO)MDP for a real-world application; in Sections 2.2 and 2.3 we discuss the reasons that may explain it.

In this context, we decided to build OpenMarkov[2], an open-source tool for building and evaluating several types of PGMs, including Bayesian networks, influence diagrams, and several types of Markovian models. We are also developing ProbModelXML, an XML format for encoding those models, with a wide variety of potentials, including tables, trees, ADDs, and canonical models, such as the noisy OR, noisy MAX, noisy AND, etc. [10].

The purpose of presenting our work at the MSDM-2012 workshop is threefold. First, to offer our tools to the com-

[1]In this paper we reserve the term "MDP" for fully-observable Markov decision processes, and will write "(PO)MDP" to refer to both MDPs and POMDPs.

[2]www.openmarkov.org.

The Seventh Annual Workshop on Multiagent Sequential Decision-Making Under Uncertainty (*MSDM-2012*), held in conjunction with *AAMAS*, June 2012, in Valencia, Spain.

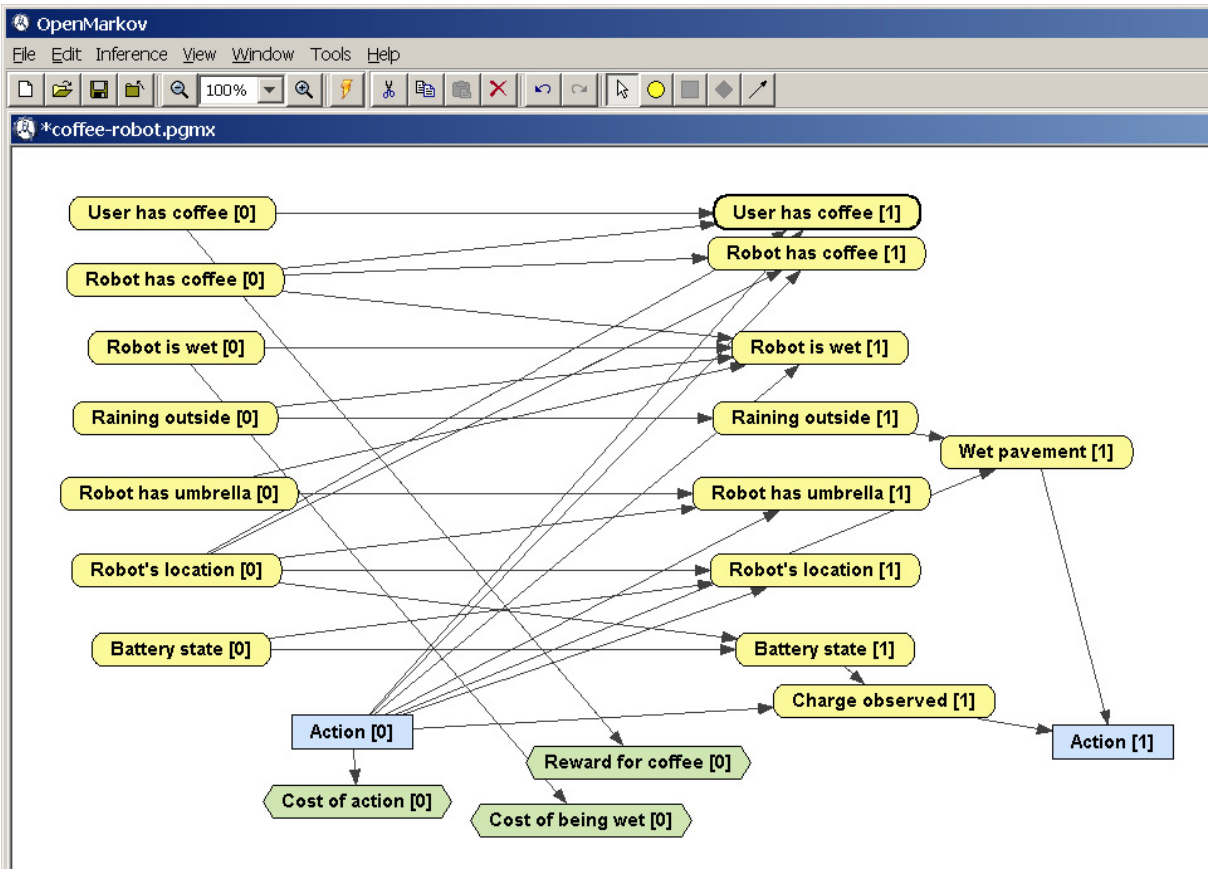

Figure 1: A factored POMDP as shown in OpenMarkov.

munity of (PO)MDPs practitioners. Second, to receive their feedback: if a significant number of potential users demand a new feature in OpenMarkov and/or in ProbModelXML, we will consider the possibility of adding it. And third, to find researchers interested in integrating in OpenMarkov the algorithms they have already implemented. This would not only be useful for evaluating (PO)MDPs inside OpenMarkov's GUI; it would permit to use OpenMarkov as a workbench for comparing the performance of different algorithms, provided that they use the same basic data structures, to make the comparison meaningful.

The rest of this paper is structured as follows: Section 2 presents background material about dynamic models and reviews some of the software tools and formats for PGMs and (PO)MDPs developed in the past. Section 3 offers an overview of OpenMarkov and Section 4 describes briefly the ProbModelXML format. Section 5 explains how to build (PO)MDPs using OpenMarkov's GUI and how multiagent models are stored in ProbModelXML. Finally, Section 7 summarizes the conclusions and discusses some lines for future research.

2. STATE OF THE ART

2.1 Dynamic models

As mentioned in the introduction, in a dynamic model, some variables are indexed by time: $t \in \{0, \ldots, h\}$, where h is the horizon of the model, which can be infinite.[3] If there is a variable X^t for a certain t, then there is also a variable $X^{t'}$ for each $t' \in \{0, \ldots, h\}$—see Figure 1, where the number between brackets denotes the index t. The subgraph that contains all the nodes X^t having the same temporal index t and the links between them is known as the t-th *time slice*.

A constraint of dynamic models is that they can not have links to the past, i.e., of the form $X^t \rightarrow Y^{t'}$ with $t > t'$. If there is a $k \in \mathbb{N}^+$ such that every link $X^t \rightarrow Y^{t'}$ satisfies that $t' - t \leq k$ (which means that all the parents of a node $Y^{t'}$ are in the t'-th slice or in the previous k slices), and this property is not satisfied for any smaller value of k, we say that the model is *of k-th order*. In practice, it is usual to work with first-order models ($k = 1$).

A node X is *stationary after k* if it has the same neighbors and the same potentials (probabilities and utilities) after the k-th slice. Put formally, for each $t > k$,

- there is a link between X^t and Y^t if and only if there is a link of the same type between X^k and Y^k;
- if there is a conditional probability $P(x^k|pa(X^k))$, then $P(x^t|pa(X^t)) = P(x^k|pa(X^k))$;

[3]In standard MDPs and POMDPs *all* the variables are indexed by time. However, we admit the possibility of atemporal variables, whose value does not change with time—see [2].

- if there is a utility function $U^k(pa(U^k))$, then $U^t(pa(U^t)) = U^k(pa(U^k))$;

and some of these properties are not fulfilled for any $k' < k$.

A dynamic model is *stationary after* k if all its nodes are stationary after k and there is no other $k' < k$ that makes the model stationary. Such a model admits a *compact representation* that contains only the first k slices. Stationary first-order models, which constitute the most common case, can be represented by only the first two time slices. In particular MDPs and POMDPs are almost always stationary and first-order, and usually have infinite horizons, and for these reasons their standard representation is a two-slice model, as in Figure 1.

2.2 Software tools for PGMs and (PO)MDPs

Several computer packages for PGMs have been developed in the last years; nearly all of them are contained in Kevin Murphy's list.[4] Table 3 shows those that are open-source. Some of them have GUIs for building PGMs, but all of them, except OpenMarkov, limit themselves to Bayesian networks and influence diagrams, i.e., they offer no support for (PO)-MDPs.

Conversely, there exist several open-source tools for (PO)-MDPs, such as Anthony Cassandra's pomdp-solve[5], Jesse Hoey's SPUDD [14][6], Pascal Poupart's Symbolic Perseus [23][7], and the APPL tool developed at the National University of Singapore[8], but none of them has a GUI for editing and evaluating this type of models.

There is only one tool for Bayesian networks and influence diagrams that claims to offer support for POMDPs: Netica[9], but it is not open-source; it is a commercial program developed by a private company, Norsys. Additionally, the ability to build (PO)MDPs is claimed only once: in the introduction of the specification of Netica's DNET format (see below). In any case, Netica does not represent POMDPs in the standard way, i.e., in the form of a two-slice model (see Fig. 1), but by means of labeled links, such that each variable X is represented by a single node in the model (instead of having a node X^t for each time slice t) and a link from X to Y with a label n in the compact model represents a link from X^t to Y^{t+n} for every t in the expanded model. Another limitation of Netica and DNET is that they cannot represent trees nor ADDs, which are almost indispensable when building real-world applications. Finally, the most important limitation of Netica with respect to (PO)MDPs is that apparently it offers no specific algorithm for evaluating this type of models; the only way of evaluating them consists of expanding the network up to a certain horizon and obtaining its policies as if it were an influence diagram, but in general this approach is unfeasible except for very small problems and very short horizons.

2.3 Formats for PGMs and (PO)MDPs

Several formats have been proposed for storing PGMs. In general, each format has been developed for a particular software tool: DNET was developed for Netica, DSC and XBN for Microsoft's MSBNx, XDSL for Smile and GENIE, etc. (The references for these formats and tools can be found in [2].) The only format intended to become a kind of standard is XMLBIF[10], proposed by Fabio Cozman with suggestions from a few other people. Unfortunately, this format is restricted to the representation of Bayesian networks with finite-state variables.

Similarly, several formats have been developed for (PO)-MDPs, each one designed for a particular software tool: there is format for Cassandra's pomdp-solve, another one for Hoey's SPUDD, the PomdpXML format for APPL... (Again, the references for these formats and tools can be found in [2].) SPUDD's format is able to encode factored POMDPs, but Cassandra's format and PomdpXML can only represent flat POMDPs. A drawback of SPUDD's format is that it is not XML; it uses a Lisp-like syntax with some peculiarities that make it difficult to build a parser.

Two additional formats were proposed to encode the problems proposed at the Probabilistic Planning Track of the International Planning Competition (IPC). The Probabilistic Planning Domain Definition Language (PPDDL),[11] used at the 4th and 5th IPC in 2004 and 2006 respectively, was able to encode factored MDPs with finite-state variables. The Relational Dynamic Influence Diagram Language (RDDL), used at the 7th IPC in 2011, was able to represent relational (PO)MDPs with both finite-state and continuous variables.[12]. These formats were not intended to encode non-temporal Bayesian networks and influence diagrams, but they has enough expressive power to represent them as well.

3. OPENMARKOV

This project started in 2003 at the Department of Artificial Intelligence of the Universidad Nacional de Educación a Distancia (UNED), in Madrid, Spain. Its original name was Carmen [1], but in 2010 it was renamed as OpenMarkov. We departed from our experience in the construction of Elvira [13],[13] an open-source tool begun in 1997 as a join project of several Spanish universities, but everything in the new program was redesigned and the code of OpenMarkov was built from scratch.

Programming language

The development language for OpenMarkov is Java, mainly in order to allow it to run on different platforms. Since the beginning we used version 1.5, which introduced new syntactical features, such as typed collections (for instance, `ArrayList<CertainClass>`) and enhanced loops, which significantly facilitate the iteration on lists.

Data structures

A probabilistic network is represented in OpenMarkovas a generic data structure consisting mainly of a graph, a set of variables, and a set of potentials. Each type of network is defined by a set of constraints, as shown in Table 2. It is possible to implement new types of networks easily by combining the existing constraints and, if necessary, by adding new ones.

[4] `www.cs.ubc.ca/~murphyk/Software/bnsoft.html`.
[5] `pomdp.org/pomdp/code`.
[6] `www.computing.dundee.ac.uk/staff/jessehoey/spudd`.
[7] `www.cs.uwaterloo.ca/~ppoupart/software.html`.
[8] `bigbird.comp.nus.edu.sg/pmwiki/farm/appl`.
[9] `www.norsys.com/netica.html`.
[10] `sites.poli.usp.br/p/fabio.cozman/Research/InterchangeFormat`.
[11] `www.tempastic.org/papers/CMU-CS-04-167.pdf`.
[12] `users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf`.
[13] `www.ia.uned.es/~elvira/`.

	Weka	JavaBayes	Elvira	BNT	Riso	UnBBayes	**OpenMarkov**	BayesLine	PNL	BNJ	OBP
Bayesian networks	yes	yes	yes	yes	yes	yes	**yes**	yes	yes	yes	yes
Influence diagrams	no	no	yes	yes	no	yes	**yes**	no	no	no	no
Dynamic models	no	no	no	yes	no	no	**yes**	no	no	no	no
Programming language	Java	Java	Java	Matlab	Java	Java	**Java**	Java	C++	Java	Python
License	GPL	GPL	?	GPL	GPL	GPL	**EUPL**	LGPL	IOSL	GPL	GPL
User manuals	yes	yes	yes	yes	yes	yes	**no***	no	no	yes	yes
Developer manuals	yes	no	no	no	no	yes	**no***	no	no	no	no
Users list/forum	yes	no	no	yes	yes	yes	**yes**	yes	no	yes	yes
Developers list/forum	yes	no	yes	yes	yes	yes	**yes**	yes	no	yes	yes
Source HTML docs	yes	yes	yes	no	yes	yes	**yes**	yes	no	no	no
Version control	yes	no	yes	no	yes	yes	**yes**	yes	no	yes	yes
Bug tracker	yes	yes	no	no	yes	yes	**yes**	yes	no	yes	yes
Start	1993	1996	1997	1999	2000	2000	**2002**	2003	2003	2004	2006
Stopped	—	2001	—	2007	2004	—	—	2003	2005	2004	2007

Table 1: Open-source packages for Bayesian networks. Some of those tools can also build and evaluate influence diagrams. The URLs for these packages can be found in K. Murphy's list, at `www.cs.ubc.ca/~murphyk/Software/bnsoft.html`, except for UnBBayes (`unbbayes.sourceforge.net`) and OpenMarkov (`www.openmarkov.org`). A star in a cell means that this feature will be available in the near future.

OpenMarkov accepts three types of variables: finite-states, numerical, and discretized, and two types of links: directed and undirected. It also has several types of potentials: uniform (mainly used to assign a default potential to each node in networks that only contain directed links), table (the most frequently used potential), delta (i.e., Kronecker delta for finite-state variables and Dirac delta for numeric variables), tree/ADD, several canonical models (OR, AND, MAX, MIN, etc.), sum, product, linear combination, conditional Gaussian, exponential, mixture of exponentials, and logistic regression. There are also three potentials for dynamic models: same as previous, cycle length shift, and Weibull distribution. In the future we will add other types of potentials, such as mixtures of polynomials. (See [2] for a detailed description of the types potentials and the network types, with bibliographical references.)

OpenMarkov is able to represent several types of networks, such as Bayesian networks, Markov networks, influence diagrams, LIMIDs, and decision analysis networks (DANs), as well as several types of temporal models: dynamic Bayesian networks, simple Markov models, MDPs, POMDPs, Dec-POMDPs, and dynamic LIMIDs.[14] Currently OpenMarkov can only evaluate Bayesian networks, influence diagrams, and simple Markov models, but it can be used to build several types of models, such as (PO)MDPs, that might be read by any other tool able to parse the ProbModelXML format.

Algorithms

We have implemented in OpenMarkov the most usual algorithms for Bayesian networks and influence diagrams, such as variable elimination [27, 9] and clique tree propagation [19, 17]. We also programmed some algorithms for evaluating MDPs, such as value iteration [4], policy iteration [15], and modified policy iteration [24], but they are not integrated in the last version of OpenMarkov, which underwent a major refactoring at the end of 2011. In OpenMarkov it is possible to learn Bayesian networks interactively from databases by applying the two most popular methods: search-and-score, with several metrics, and the PC algorithm [21].

Graphical user interface

The graphical user interface (GUI) is very similar to those of other software tools for PGMs, especially to that of Elvira. It has two main working modes: edition and inference. It has been designed for internationalization; currently messages can be displayed in English or Spanish, and other languages can be easily added using Java's facilities.

Testing

In order to guarantee as much as possible the robustness of the tool, we have built a test suite for each class in OpenMarkov with JUnit.[15] After introducing a modification in OpenMarkov, we run the battery of tests in order to detect possible bugs in the program.

Code hosting and version control system

As a version control tool, we initially chose subversion, installed on a local server, but in October 2011 we reorganized the code into a set of Maven subprojects and migrated the code to Bitbucket, a code-hosting system similar to SourceForge, JavaSource, or Google Code. It offers two control version systems: Mercurial and Git; OpenMarkov uses Mercurial. It also offers wikis, issue trackers, and other facilities. We use Mercurial to store in Bitbucket a working copy of OpenMarkov's Java source code. There is also a wiki and an issue tracker for each subproject.

Using Maven, OpenMarkov is deployed on a local nexus repository [16], which contains three types of files: Java source code of snapshots and releases (in contrast with Bitbucket,

[14] Three of these models have been proposed originally by our group: DANs, simple Markov models, and dynamic LIMIDs.

[15] `www.junit.org`.

[16] `nexus.openmarkov.org`.

	BayesianNetwork	MarkovNetwork	InfluenceDiagram	LIMID	DecisionAnalysisNetwork	DynamicBayesianNetwork	SimpleMarkovModel	MDP	POMDP	Dec-POMDP	DynamicLIMID
NoEmptyName	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DistinctVariableNames	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
OnlyFiniteStatesVariables	O	O	O	O	O	O	O	O	O	O	O
OnlyNumericVariables	O	O	O	O	O	O	O	O	O	O	O
OnlyChanceNodes	**Y**	**Y**	N	N	N	**Y**	N	N	N	N	N
OnlyOneUtilityNode	–	–	O	O	O	–	O	O	O	O	O
OnlyAtemporalVariables	Y	Y	Y	Y	Y	**N**	**N**	**N**	**N**	**N**	**N**
OnlyTemporalVariables	N	N	N	N	N	**Y**	N	**Y**	**Y**	**Y**	**Y**
OnlyOneAgent	Y	Y	Y	Y	Y	Y	Y	Y	Y	**N**	Y
DistinctLinks	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
NoMultipleLinks	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
OnlyDirectedLinks	Y	**N**	Y	Y	Y	Y	Y	Y	Y	Y	Y
OnlyUndirectedLinks	N	**Y**	N	N	N	N	N	N	N	N	N
NoRestriction	Y	Y	Y	Y	**N**	Y	Y	Y	Y	Y	Y
NoRevelationArc	Y	Y	Y	Y	**N**	Y	Y	Y	Y	Y	Y
NoSelfLoop	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
NoCycle	Y	**O**	Y	Y	Y	Y	Y	Y	Y	Y	Y
NoClosedPath	O	O	O	O	O	O	O	O	O	O	O
MaxNumParents	O	O	O	O	O	O	O	O	O	O	O
NoUtilityParent	–	–	O	O	O	–	O	O	O	O	O
NoSupervalueNode	–	–	O	O	O	–	O	O	O	O	O
NoMixedParents	–	–	O	O	O	–	O	O	O	O	O
NoBackwardLink	–	–	–	–	–	Y	Y	Y	Y	Y	Y

Table 2: Constraints used in OpenMarkov. The letter in each cell indicates whether a constraint is associated with a network type: Y = yes, N = no, O = optionally. A dash means that a constraint does not make sense because of the presence of another constraint—see [2]. Each constraint has a default behavior; a bold letter in this table means that the default behavior has been overriden for a particular type of network.

which stores the whole version history, including the most recent files not yet deployed), compiled Java binaries, and Javadoc HTML files. For further details, see OpenMarkov's wiki.[17]

4. PROBMODELXML

4.1 Motivation

As mentioned in the introduction, we decided to develop a new format for PGMs aimed at becoming a common language for several research groups and several software tools. The first requisite was that the format should accommodate different types of models (Bayesian networks, Markov networks, influence diagrams, POMDPs, etc.), three types of variables (finite-states, numerical, and discretized), a variety of potentials (table, tree/ADD, canonical models...), and a wide range of properties. However, given that it is impossible to foresee from the beginning all the needs that will emerge in the future, the format should be extensible, i.e., it should be able to represent new types of models and new properties without changing the specification of the format. Second, the syntax and the semantics of the format should be clearly documented, in order to avoid ambiguities and misinterpretations. Third, the syntax of the format should be based on the Extensible Markup Language (XML) specification produced by the World Wide Web Consortium (W3C),[18] mainly because XML is much easier to parse than other types of syntaxes: there exist many tools for parsing XML from several programming languages (Java, C++, etc.), with the corresponding utilities for writing XML files from those languages, as well as other tools for specifying XML formats and for validating them (DTD, XML Schema, Relax NG, ISO DSDL...).

4.2 Specification of models

The skeleton of a ProbModelXML file is:

```
<?xml version="1.0" encoding="UTF-8" ?>
<ProbModelXML formatVersion=string >
  <ProbNet type=enumNetworkType />0..1
  <InferenceOptions />0..1
  <Policies />0..1
  <Evidence />0..1
</ProbModelXML>
```

and the extension we propose for those files is .pgmx. The skeleton of a probabilistic network is:

[17] wiki.openmarkov.org.

[18] www.w3.org/XML.

```
<ProbNet type=enumNetworkType >
  <AdditionalConstraints />0..1
  <Comment />0..1
  <DecisionCriteria />0..1
  <Agents />0..1
  <Language />0..1
  <AdditionalProperties />0..1
  <Variables />
  <Links />
  <Potentials />
</ProbNet>
```

The tag `<AdditionalProperties>` is used to extend the ProbModelXML format by representing other properties not explicit in this format. An additional property can appear in the context of a *ProbNet*, an *Evidence Case*, a *Criterion* (for multicriteria decision making), an *Agent* (in multi-agent models), a *Variable*, a *State* of a variable, a *Potential* and a *Policy*. In all the cases, the skeleton for encoding additional properties is as follows:

```
<AdditionalProperties>
  <Property name=string value=string />1..n
</AdditionalProperties>
```

A complete specification of the ProbModelXML format, with detailed explanations and many examples can be found in [2].

5. (PO)MDPS IN OPENMARKOV AND PROBMODELXML

5.1 Editing (PO)MDPs with OpenMarkov's GUI

The code of OpenMarkov's GUI does not depend directly on the type of network, but on the constraints. Therefore it is very easy to extend the GUI to deal with new types of networks: it suffices to modify the aspects that depend on the new constraints, if any. Given that the constraints that define an influence diagram are almost the same as those for a POMDPs or a Dec-POMDP (cf. Table 2), the edition of these three types of models is very similar. Figure 1 shows a factored POMDP that models a robot designed for serving coffee to a single user. This example, proposed by Jesse Hoey and encoded in SPUDD format [14], is included with the source code of Pascal Poupart's software Symbolic Perseus.[19]

As that figure shows, in OpenMarkov, rectangles represent decisions (actions), rounded rectangles represent chance variables, and hexagons represent utility nodes, i.e., costs and rewards. In this figure, the number in square brackets indicates the time slice to which a node belongs. A link from a node in a time slice to a node in the same slice is said to be *intratemporal*; a link from a time slice to the next one is said to be *intertemporal*. Links from the second time slice to the first one are not allowed, because it would contradict our notion of causality. A link $X \rightarrow D$, where D is a decision, is called an *informational link* and means that the value that variable X has taken is observed when making decision D. In this example, there are two observable variables, Charge observed, that gives information about the state of the battery, and Wet pavement, which provides indirect evidence on whether it is raining or not.[20]

As in the case of Bayesian networks and influence diagrams, each chance node in the first time slice has an associated probability potential, which in this example is a prior probability, because no chance node in that slice has parents, but in other examples there may be intratemporal links in the first slice. If a node has parents, either in the same time slice or in the previous one, its associated probability is conditioned on that parent. For instance, the probability of Wet pavement in this example is conditioned on the current value of Raining outside and on the Action chosen in the previous time slice; when Action[0] was different from measurewet, the probabilities of the two values of this variable, namely wet and dry, are the same, and therefore this "observation" does not affect the probability of Raining outside.

As in the case of influence diagrams, each utility node has an associated utility potential that depends on the parent of that node in the graph. In this example there is one reward that takes a positive value when the chance variable User has coffee takes the value yes, and four costs, that depend on the Action chosen. In other examples, a utility (node) may depend on both chance variables and actions.

The edition of a potential in OpenMarkov depends on the type of potential. If the potential is a table, its edition is very similar to that of other software tools for PGMs. If the potential is a tree or an ADD, the window for editing it is based on Java's JTree class and the corresponding panels, listeners, etc., as shown in Figure 2.

5.2 Multiagent POMDPs

Multiagent POMDPs can be specified in the ProbModelXML format by using the `Agents` tag that we have seen above in the specification of probabilistic networks. The skeleton for this tag is:

```
<Agents>
  <Agent name=string>
    <AdditionalProperties />0..1
  </Agent>2..n
</Agents>
```

For example, in a surveillance system we might have two agents:

```
<Agents>
  <Agent name="video camera" />
  <Agent name="mobile robot" />
</Agents>
```

Then, each decision variable can be assigned to a particular agent, as in the following example:

[19] `http://www.cs.uwaterloo.ca/~ppoupart/software.html`. A direct link for this example is `www.cs.uwaterloo.ca/~ppoupart/software/symbolicPerseus/problems/coffee/coffee3po.txt`.

[20] Information arcs were used by the first time in an influence diagram [16]. Later, they have been used in other PGMs, such as factored POMDPs [7], LIMIDs [18], and DLIMIDs [26, 12] to indicate which variables are observable.
On the contrary there are other models that do not require information links. One type of them are MDPs [4, 6], which assume that all the variables are observable. Other type are decision analysis networks, which use *revelation arcs* to indicate when some variables become observed [11].

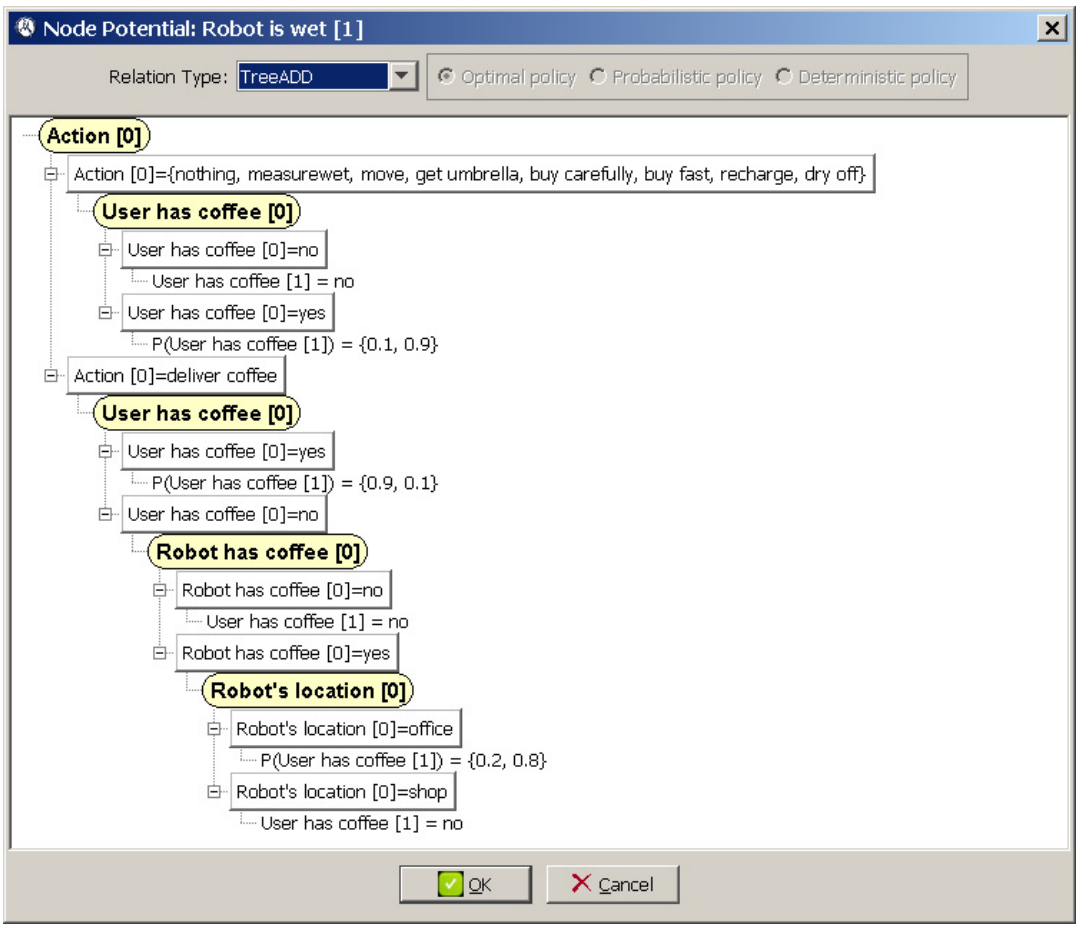

Figure 2: Edition of a tree/ADD potential in OpenMarkov.

```
<Variable name="Switch" type="FiniteStates"
      role="Decision" timeSlice="0" >
   <Agent name="video camera" >
   <States>
      <State name="off" />
      <State name="on" />
   </States>
</Variable>
```

Multiagent models can be edited in OpenMarkov by first defining a list of agent and then associating each decision to an agent by means of a push-down widget.

6. RELATED WORK

ProbModelXML is similar to the RDDL format mentioned in Section 2.3, which can represent relational models and logic relations, while ProbModelXML cannot. On the other hand the latter can represent more types of networks and potentials, including Dec-POMDPs. Another advantage of ProbModelXML is its XML syntax. It would be interesting to extend ProbModelXML to cover all the features included in RDDL.

The RDDL simulator[21] is a program that can show graphically the models written in this format, but it cannot edit them. In contrast, it can evaluate those models, while OpenMarkov cannot.

[21] code.google.com/p/rddlsim.

7. CONCLUSIONS AND FUTURE WORK

OpenMarkov is an open-source tool for editing and evaluating probabilistic graphical models. If permits to edit several types of probabilistic networks, including MDPs, POMDPs and Dec-POMDPs, and it is easy to extend it to deal with other types of networks. It also implements several types of potentials, including tables, trees, and ADDs. The native format for OpenMarkov is ProbModelXML. Given that the format and the tool have been developed in parallel, the types of variables, potentials, and networks are essentially the same in both of them, even though some of the features included in the specification of the format have not yet been implemented in OpenMarkov. This is one of the first tasks in the agenda of OpenMarkov's developers. Other tasks are to continue debugging OpenMarkov, mainly its GUI, which is the most difficult component to debug, and to extend its documentation. Later, we will consider the possibility of integrating in OpenMarkov some of the existing algorithms for evaluating POMDPs; we have received some nice offers from researchers of other groups, but currently we cannot address that issue given the limited amount of resources in our group.

Acknowledgments

We thank Jesse Hoey for providing with us the coffee robot example and several useful comments. We thank one of the anonymous reviewers for making us aware of the existence of PPDDL and RDDL. We are grateful to all the people who have collaborated in the OpenMarkov project.

This work has been supported by grants TIN-2006-11152 and TIN2009-09158, of the Spanish Ministry of Science and Technology, and by FONCICYT grant 85195.

8. REFERENCES

[1] M. Arias and F. J. Díez. Carmen: An open source project for probabilistic graphical models. In *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models (PGM'08)*, pages 25–32, Hirtshals, Denmark, 2008.

[2] M. Arias, F. J. Díez, and M. P. Palacios. ProbModelXML. A format for encoding probabilistic graphical models. Technical Report CISIAD-11-02, UNED, Madrid, Spain, 2011.

[3] K. J. Åström. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.

[4] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[5] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI–95)*, pages 1104–1111, Montreal, Canada, 1995.

[6] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121:49–107, 2000.

[7] C. Boutilier and D. Poole. Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96*, pages 1168–1175, Portland, OR, 1996. AAAI Press.

[8] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150, 1989.

[9] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the Proceedings of the Twelfth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pages 211–219, San Francisco, CA, 1996. Morgan Kaufmann.

[10] F. J. Díez and M. J. Druzdzel. Canonical probabilistic models for knowledge engineering. Technical Report CISIAD-06-01, UNED, Madrid, Spain, 2006.

[11] F. J. Díez and M. Luque. Representing decision problems with decision analysis networks. Technical Report CISIAD-10-01, UNED, Madrid, Spain, 2010.

[12] F. J. Díez and M. A. J. van Gerven. Dynamic LIMIDs. In L. E. Sucar, J. Hoey, and E. Morales, editors, *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*, pages 164–189. IGI Global, Hershey, PA, 2011.

[13] The Elvira Consortium. Elvira: An environment for creating and using probabilistic graphical models. In J. A. Gámez and A. Salmerón, editors, *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM'02)*, pages 1–11, Cuenca, Spain, 2002.

[14] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 279–288, Stockholm, Sweden, 1999. Morgan Kaufmann, San Francisco, CA.

[15] R. A. Howard. *Dynamic Programming and Markov Processes*. The MIT Press, Cambridge, MA, 1960.

[16] R. A. Howard and J. E. Matheson. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *Readings on the Principles and Applications of Decision Analysis*, pages 719–762. Strategic Decisions Group, Menlo Park, CA, 1984.

[17] F. V. Jensen, K. G. Olesen, and S. K. Andersen. An algebra of Bayesian belief universes for knowledge-based systems. *Networks*, 20:637–660, 1990.

[18] S. L. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Management Science*, 47:1235–1251, 2001.

[19] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50:157–224, 1988.

[20] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, Computer Science Division, University of California, Berkeley, 2002.

[21] R. E. Neapolitan. *Learning Bayesian Networks*. Prentice-Hall, Upper Saddle River, NJ, 2004.

[22] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[23] P. Poupart. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. PhD thesis, Dept. of Computer Science, University of Toronto, Canada, 2005.

[24] M. L. Puterman and M. Shin. Modified policy iteration algorithms for discounted Markov decision processes. *Management Science*, 24:1127–1137, 1978.

[25] M. T. J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[26] M. A. J. van Gerven, F. J. Díez, B. G. Taal, and P. J. F. Lucas. Selecting treatment strategies with dynamic limited-memory influence diagrams. *Artificial Intelligence in Medicine*, 40:171–186, 2007.

[27] N. L. Zhang and D. Poole. Intercausal independence and heterogeneous factorization. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence (UAI'94)*, pages 606–14, Seattle, WA, 1994. Morgan Kaufmann, San Francisco, CA.