Chapter 8
Solving Relational and First-Order Logical Markov Decision Processes: A Survey

Martijn van Otterlo

Abstract In this chapter we survey representations and techniques for Markov decision processes, reinforcement learning, and dynamic programming in worlds explicitly modeled in terms of *objects* and *relations*. Such relational worlds can be found everywhere in planning domains, games, real-world indoor scenes and many more. Relational representations allow for expressive and natural datastructures that capture the objects and relations in an explicit way, enabling generalization over objects and relations, but also over similar problems which differ in the number of objects. The field was recently surveyed completely in (van Otterlo, 2009b), and here we describe a large portion of the main approaches. We discuss model-free – both value-based and policy-based – and model-based dynamic programming techniques. Several other aspects will be covered, such as models and hierarchies, and we end with several recent efforts and future directions.

8.1 Introduction to Sequential Decisions in Relational Worlds

As humans we speak and think about the world as being made up of *objects* and *relations* among objects. There are books, tables and houses, tables *inside* houses, books *on top of* tables, and so on. *"We are equipped with an inductive bias, a predisposition to learn to divide the world up into objects, to study the interaction of those objects, and to apply a variety of computational modules to the representation of these objects"* (Baum, 2004, p. 173). For intelligent agents this should not be different. In fact, *"... it is hard to imagine a truly intelligent agent that does not conceive of the world in terms of objects and their properties and relations to other objects"* (Kaelbling et al, 2001). Furthermore, such representations are highly ef-

Martijn van Otterlo
Radboud University Nijmegen, The Netherlands
e-mail: m.vanotterlo@donders.ru.nl

fective and compact:*"The description of the world in terms of objects and simple interactions is an enormously compressed description"* (Baum, 2004, p. 168).

The last decade a new subfield in reinforcement learning (RL) has emerged that tries to endow intelligent agents with both the knowledge representation of (probabilistic) first-order logic – to deal with objects and relations – and efficient RL algorithms – to deal with decision-theoretic learning in complex and uncertain worlds. This field – **relational RL** (van Otterlo, 2009b) – takes inspiration, representational methodologies and algorithms from diverse fields (see Fig. 8.2(**left**)) such as RL (Sutton and Barto, 1998), *logic-based artificial intelligence* (Minker, 2000), knowledge representation (KR) (Brachman and Levesque, 2004), *planning* (see Russell and Norvig, 2003), *probabilistic-logical machine learning* (De Raedt, 2008) and *cognitive architectures* (Langley, 2006). In many of these areas the use of objects and relations is widespread and it is the assumption that if RL is to be applied in these fields – e.g. for cognitive agents learning to behave, or RL in tasks where communicating the acquired knowledge is required – one has to investigate how RL algorithms interact with expressive formalisms such as first-order logic. Relational RL offers a new representational paradigm for RL and can tackle many problems more compactly and efficiently than state-of-the-art *propositional* approaches, and in addition, can tackle new problems that could not be handled before. It also offers new opportunities to inject additional *background knowledge* into algorithms, surpassing *tabula rasa* learning.

In this chapter we introduce relational RL as new representational paradigm in RL. We start by introducing the elements of generalization in RL and briefly sketch historical developments. In Section 8.2 we introduce relational representations for Markov decision processes (MDPs). In Sections 8.3 and 8.4 we survey model-based and model-free solution algorithms. In addition we survey other approaches, such as hierarchies and model learning, in Section 8.5, and we describe recent developments in Section 8.6. We end with conclusions and future directions in Section 8.7.

8.1.1 MDPs: Representation and Generalization

Given MDP $M = \langle S, A, T, R \rangle$ the usual goal is to compute value functions (V or Q) or directly compute a policy π. For this, a plethora of algorithms exists which all more or less fall in the category of *generalized policy iteration* (GPI) (see Sutton and Barto, 1998), which denotes the iterative loop involving **i)** *policy evaluation*, computing V^π, and **ii)** *policy improvement*, computing an improved policy π' from V^π. GPI is formulated for opaque states and actions, not taking into account the use of representation (and generalization).

In (van Otterlo, 2009b) we formulated a more general concept PIAGET, in which we make the use of representation explicit (see Fig. 8.1). It distinguishes four ways in which GPI interacts with representation. PIAGET-0 is about first generating a compact representation (an *abstraction level*), e.g. through *model minimization*, or *feature extraction*. PIAGET-1 and PIAGET-2 are closest to GPI in that they as-

sume a *fixed* representation, and learn either MDP-related functions (Q, V) over this abstraction level (PIAGET-1) or they learn *parameters* (e.g. neural network weights) of the representation simultaneously (PIAGET-2). PIAGET-3 methods are the most general; they *adapt* abstraction levels while solving an MDP.

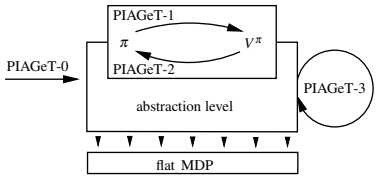

Fig. 8.1: ***P**olicy **I**teration using **A**bstraction and **Ge**neralization **T**echniques* (PIAGET).

Since relational RL is above all a representational upgrade of RL we have to look at concepts of generalization (or abstraction) in MDPs. We can distinguish five types typically used in MDPs: **i)** state space abstraction, **ii)** factored MDP representations, **iii)** value functions, **iv)** policies, and **v)** hierarchical decompositions. Relational RL focuses on representing an MDP in terms of objects and relations, and then employing *logical generalization* to obtain any of the five types, possibly making use of results obtained with propositional representations.

Generalization is closely tied to representation. In other words, *one can only learn what one can represent*. Even though there are many ways of representing knowledge, including *rules*, *neural networks*, *entity-relationship models*, *first-order logic* and so on, there are few *representational classes*:

Atomic. Most descriptions of algorithms and proofs use so-called atomic state representations, in which the state space S is a discrete *set*, consisting of discrete states s_1 to s_N and A is a set of actions. No abstraction (generalization) is used, and all computation and storage (e.g. of values) happens *state-wise*.

Propositional. This is the most common form of MDP representation (see Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998; Boutilier et al, 1999, for thorough descriptions). Each state consists of a *vector* of *attribute-value pairs*, where each attribute (*feature*, *random variable*) represents a *measurable quantity* of the domain. Features can be binary or real-valued. Usually the action set A is still a discrete set. Propositional encodings enable the use of, e.g. *decision trees* or *support vector machines* to represent value functions and policies, thereby generalizing over parts of the state space.

Deictic. Some works use *deictic* representations (e.g. Finney et al, 2002) in which propositional encodings are used to *refer* to objects. For example, the value of the feature *"the color of the object in my hand"* could express something about a specific object, without explicitly naming it. Deictic representations, as well as *object-oriented* representations (Diuk et al, 2008) are still propositional, but bridge the gap to explicit relational formalisms.

Relational. A relational representation is an expressive *knowledge representation* (KR) format in which *objects* and *relations* between objects can be expressed in an explicit way, and will be discussed in the next section.

Each representational class comes with its own ways to generalize and abstract. Atomic states do not offer much in that respect, but for propositional represen-

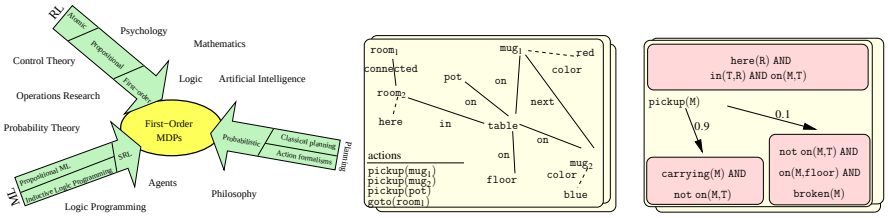

Fig. 8.2: (**left**) The material in this chapter embedded in various subfields of artificial intelligence (AI), (**middle**) relational representation of a scene, (**right**) generalization over a probabilistic action.

tations much is possible. Generalization makes use of *structure* inherent in problem domains. Such structure can be found and exploited in two distinct ways. One can *exploit structure in representations*, making solutions compact, but also *employ structured representations in algorithms*, making RL algorithms more efficient.

8.1.2 Short History and Connections to Other Fields

The research described in this chapter is a natural development in at least three areas. The core element (see Fig. 8.2) is a *first-order* (or *relational*) MDP.

Reinforcement learning. Before the mid-nineties, many *trial-and-error learning*, *optimal control*, *dynamic programming* (DP) and *value function approximation* techniques were developed. The book by Sutton and Barto (1998) marked the beginning of a new era in which more sophisticated algorithms and representations were developed, and more theory was formed concerning approximation and convergence. Around the turn of the millennium, relational RL emerged as a new subfield, going beyond the propositional representations used until then. Džeroski et al (1998) introduced a first version of Q-learning in a relational domain, and Boutilier et al (2001) reported the first value iteration algorithm in first-order logic. Together they initiated a new *representational* direction in RL.

Machine Learning. Machine learning (ML), of which RL is a subfield, has used relational representations much earlier. *Inductive logic programming* (ILP) (Bergadano and Gunetti, 1995) has a long tradition in learning logical concepts from data. Still much current ML research uses purely propositional representations and focuses on probabilistic aspects (Alpaydin, 2004). However, the last decade logical and probabilistic approaches are merging in the field of *statistical relational learning* (SRL) (De Raedt, 2008). Relational RL itself can be seen as an additional step, adding a *utility framework* to SRL.

Action languages. Planning and cognitive architectures are old AI subjects, but before relational RL surprisingly few approaches could deal efficiently with decision-theoretic concepts. First-order action languages ranging from STRIPS

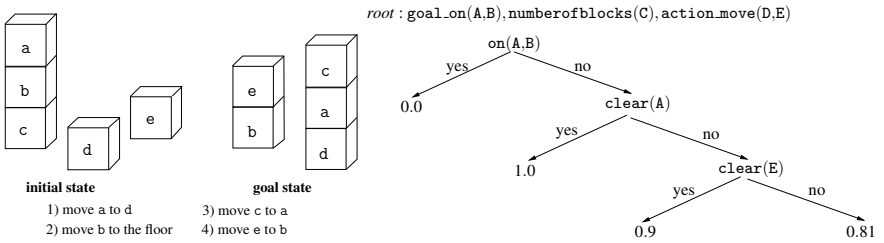

Fig. 8.3: (**left**) a Blocks World planning problem and an optimal plan, (**right**) a logical Q-tree.

(Fikes and Nilsson, 1971) to *situation calculus* (Reiter, 2001) were primarily used in deterministic, goal-based planning contexts, and the same holds for cognitive architectures. Much relational RL is based on existing action formalisms, thereby extending their applicability to solve decision-theoretic problems.

8.2 Extending MDPs with Objects and Relations

Here we introduce relational representations and logical generalization, and how to employ these for the formalization of *relational* (or, *first-order*) MDPs. Our description will be fairly informal; for more detailed treatment see (van Otterlo, 2009b) and for logical formalizations in general see (Brachman and Levesque, 2004).

8.2.1 Relational Representations and Logical Generalization

Objects and relations require a formalism to express them in explicit form.
Fig. 8.2(**middle**) depicts a typical indoor scene with several *objects* such as $\texttt{room}_1$, $\texttt{pot}$ and $\texttt{mug}_2$. *Relations* between objects are the solid lines, e.g. the relations $\texttt{connected}(\texttt{room}_1,\texttt{room}_2)$ and $\texttt{on}(\texttt{table},\texttt{floor})$. Some relations merely express *properties* of objects (i.e. the dashed lines) and can be represented as $\texttt{color}(\texttt{mug}_1,\texttt{red})$, or as a *unary* relation $\texttt{red}(\texttt{mug}_1)$. Actions are represented in a similar way, rendering them *parameterized*, such as $\texttt{pickup}(\texttt{mug}_1)$ and $\texttt{goto}(\texttt{room}_1)$.

Note that a propositional representation of the scene would be cumbersome. Each relation between objects should be represented by a binary feature. In order to represent a state by a feature *vector*, all objects and relations should first be fixed, and ordered, to generate the features. For example, it should contain $\texttt{on\_mug}_1\texttt{\_table} = 1$ and $\texttt{on\_mug}_2\texttt{\_table} = 1$, but also useless features such as $\texttt{on\_table\_table} = 0$ and $\texttt{on\_room}_1\texttt{\_mug}_2 = 0$. This would blow up in size since it must contain many irrelevant relations (many of which do not hold in a state), and would be very inflexible

if the number of objects or relations would vary. Relational representations can naturally deal with these issues.

An important way to generalize over objects is by using *variables* that can *stand for* different objects (denoted by uppercase letters). For example, in Fig. 8.2(**right**) an action specification makes use of that in order to `pickup` *an* object denoted by `M`. It specifies that the object should be `on` some object denoted by `T`, in the same current location `R`, and `M` could be e.g. $\texttt{mug}_1$ or $\texttt{mug}_2$. The bottom two rectangles show two different *outcomes* of the action, governed by a probability distribution.

Out of many artificial domains, the Blocks World (Slaney and Thiébaux, 2001, and Fig. 8.3(**left**)) is probably the most well-known problem in areas such as KR and planning (see Russell and Norvig, 2003; Schmid, 2001; Brachman and Levesque, 2004) and recently, relational RL. It is a computationally hard problem for general purpose AI systems, it has a relatively simple form, and it supports meaningful, systematic and affordable experiments. Blocks World is often used in relational RL and we use it here to explain important concepts.

More generally, a relational representation consists of a *domain of objects* (or, *constants*) C, and a set of relations $\{\texttt{p}/\alpha\}$ (or, *predicates*) of which each can have several arguments (i.e. the *arity* α). A *ground atom* is a relation applied to constants from C, e.g. $\texttt{on}(\texttt{a},\texttt{b})$. For generalization, usually a *logical language* is used which – in addition to the domain C and set of predicates P – also contains *variables*, *quantifiers* and *connectives*. In the sentence (i.e. *formula*) $\mathbb{Z} \equiv \exists \texttt{X},\texttt{Y}\ \texttt{on}(\texttt{X},\texttt{Y}) \wedge \texttt{clear}(\texttt{X})$ in such a language, $\mathbb{Z}$ represents all states in which *there are* (expressed by the quantifier $\exists$) two objects (or, blocks) `X` and `Y` which are `on` each other, *and also* (expressed by the connective $\wedge$, or, the *logical AND*) that `X` is clear. In Fig. 8.3(**left**), in the initial state `X` could only be `a`, but in the goal state, it could refer to `c` or `e`. Note that $\mathbb{Z}$ is not ground since it contains variables.

Logical abstractions can be *learned* from data. This is typically done through methods of ILP (Bergadano and Gunetti, 1995) or SRL (De Raedt, 2008). The details of these approaches are outside the scope of this chapter. It suffices here to see them as *search algorithms* through a vast space of logical abstractions, similar in spirit to propositional tree and rule learners (Alpaydin, 2004). The structure of the space is given by the expressivity of the logical language used. In case of SRL, an additional problem is learning the *parameters* (e.g. probabilities).

8.2.2 Relational Markov Decision Processes

We first need to represent all the aspects of the *problem* in relational form, i.e. MDPs in relational form, based on a domain of objects D and a set of predicates P.

Definition 8.2.1 *Let $P = \{\texttt{p}_1/\alpha_1,\ldots,\texttt{p}_n/\alpha_n\}$ be a set of predicates with their arities, $C = \{\texttt{c}_1,\ldots,\texttt{c}_k\}$ a set of constants, and let $A' = \{\texttt{a}_1/\alpha_1,\ldots,\texttt{a}_m/\alpha_m\}$ be a set of actions with their arities. Let S' be the set of all ground atoms that can be constructed from P and C, and let A be the set of all ground atoms over A' and C.*

*A **relational Markov decision process (RMDP)** is a tuple $M = \langle S, A, T, R \rangle$, where S is a subset of $2^{S'}$, A is defined as stated, $T :: S \times A \times S \to [0,1]$ is a probabilistic transition function and $R :: S \times A \times S \to \mathbb{R}$ a reward function.*

Thus, each state is represented by the set of ground relational atoms that are true in that state (called an *interpretation* in logic, or a *ground state* here), actions are ground relational atoms, and the transition and reward function are defined as usual, but now over ground relational states and actions. This means that, unlike propositional representations, states are now *unordered* sets of relational atoms and the number of atoms is not fixed.

Example 8.2.1 *Using an example of a five-block world based on the predicate set $P = \{\texttt{on/2}, \texttt{cl/1}\}$, the constant set $C = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}, \texttt{floor}\}$ and the action set $A' = \{\texttt{move/2}\}$ we can define the blocks world containing 5 blocks with $|S| = 501$ legal states. T can be defined such that it complies with the standard dynamics of the Blocks World* `move` *operator (possibly with some probability of failure) and R can be set positive for states that satisfy some goal criterium and 0 otherwise.*

A concrete state $\texttt{s}_1$ *is* $\{\texttt{on(a,b)}, \texttt{on(b,c)}, \texttt{on(c,d)}, \texttt{on(d,e)}, \texttt{on(e,floor)}, \texttt{cl(a)}\}$ *in which all blocks are stacked. The action* `move(a,floor)` *moves the top block* `a` *on the floor. The resulting state* $\texttt{s}_2$ *would be* $\{\texttt{on(a,floor)}, \texttt{cl(a)}, \texttt{on(b,c)}, \texttt{on(c,d)}, \texttt{on(d,e)}, \texttt{on(e,floor)}, \texttt{cl(b)}\}$ *unless there is a probability that the action fails, in which case the current state stays* $\texttt{s}_1$. *If the goal would be to stack all blocks, state reaching* $\texttt{s}_1$ *would get a positive reward and* $\texttt{s}_2$ *0.*

8.2.3 Abstract Problems and Solutions

Relational MDPs form the core model underlying all work in relational RL. Usually they are not posed in ground form, but specified by logical languages which can differ much in their expressivity (e.g. which quantifiers and connectives they support) and reasoning complexity. Here we restrict our story to a simple form: an **abstract state** is a conjunction $\mathbb{Z} \equiv \mathbb{Z}_1 \wedge \ldots \wedge \mathbb{Z}_m$ of logical atoms, which can contain variables. A conjunction is implicitly *existentially quantified*, i.e. an abstract state $\mathbb{Z}_1 \equiv \texttt{on(X,Y)}$ should be seen as $\exists \texttt{X} \exists \texttt{Y} \mathbb{Z}_1$ which reads as *there are two blocks, denoted* `X` *and* `Y` *AND block* `X` *is* `on` *block* `Y`. For brevity we will omit the quantifiers and connectors from now. An abstract state $\mathbb{Z}$ *models* a ground state $\mathtt{z}$ if we can find a *substitution* of the variables in $\mathbb{Z}$ such that all the substituted atoms appear in $\mathtt{z}$. It is said that $\mathtt{Z}$ *θ-subsumes* $\mathtt{z}$ (with θ the substitution). For example, $\mathbb{Z}_1$ models (subsumes) the state `clear(a)`, `on(a,b)`, `clear(c)`, ... since we can substitute `X` and `Y` with `a` and `b` and then `on(a,b)` appears in $\mathtt{z}$. Thus, an abstract state generalizes over a *set* of states of the underlying RMDP, i.e. it is an *aggregate* state.

Abstract states are also (partially) ordered by a *θ-subsumption*. If an abstract state $\mathbb{Z}_1$ subsumes another abstract state $\mathbb{Z}_2$, then $\mathbb{Z}_1$ is *more general* than $\mathbb{Z}_2$. A *domain theory* supports reasoning and constraint handling to check whether states are legal (wrt. to the underlying semantics of the domain) and to *extend* states (i.e. derive new

facts based on the domain theory). This could for the *Blocks World* allow to derive from $\texttt{on(a,b)}$ that also $\texttt{under(b,a)}$ is true. In addition, it would also exclude states that are easy to generate from relational atoms, but are impossible in the domain, e.g. $\texttt{on(a,a)}$, using the rule $\texttt{on(X,X)} \rightarrow \texttt{false}$.

The transition function in Definition 8.2.1 is usually defined in terms of *abstract actions*. Their definition depends on a specific *action logic* and many different forms exists. An action induces a mapping between abstract states, i.e. from a *set* of states to another set of states. An abstract action defines a *probabilistic* action operator by means of a probability distribution over a set of deterministic action outcomes. A generic definition is the following, based on *probabilistic STRIPS* (Hanks and McDermott, 1994).

$$\begin{array}{l} \texttt{cl(X),cl(Y),on(X,Z)}, \\ \texttt{X} \neq \texttt{Y},\texttt{Y} \neq \texttt{Z},\texttt{X} \neq \texttt{Z} \end{array} \xrightarrow{0.9 : \texttt{move(X,Y)}} \begin{array}{l} \texttt{on(X,Y),cl(X),cl(Z)}, \\ \texttt{X} \neq \texttt{Y},\texttt{Y} \neq \texttt{Z},\texttt{X} \neq \texttt{Z} \end{array} \tag{8.1}$$

$$\begin{array}{l} \texttt{cl(X),cl(Y),on(X,Z)}, \\ \texttt{X} \neq \texttt{Y},\texttt{Y} \neq \texttt{Z},\texttt{X} \neq \texttt{Z}. \end{array} \xrightarrow{0.1 : \texttt{move(X,Y)}} \begin{array}{l} \texttt{cl(X),cl(Y),on(X,Z)}, \\ \texttt{X} \neq \texttt{Y},\texttt{Y} \neq \texttt{Z},\texttt{X} \neq \texttt{Z}. \end{array}$$

which moves block $\texttt{X}$ on $\texttt{Y}$ with probability 0.9. With probability 0.1 the action fails, i.e., we do not change the state. For an action rule pre $\rightarrow$ post, if pre is θ-subsumed by a state $\mathtt{z}$, then in the resulting state $\mathtt{z}'$ is $\mathtt{z}$ with pre$\,\theta$ (applied substitution) removed, and post$\,\theta$ added. Applied to $\mathtt{z} \equiv \texttt{cl(a)}, \texttt{cl(b)}, \texttt{on(a,c)}$ the action tells us that $\texttt{move(a,b)}$ will result in $\mathtt{z}' \equiv \texttt{on(a,b)}, \texttt{cl(a)}, \texttt{cl(c)}$ with probability 0.9 and with probability 0.1 we stay in $\mathtt{z}$. An action defines a probabilistic mapping over the set of state-action-state pairs.

Abstract (state) **reward functions** are easy to specify with abstract states. A generic definition of R is the set $\{\langle \mathbb{Z}_i, r\rangle \mid i = 1 \ldots N\}$ where each $\mathbb{Z}_i$ is an abstract state and r is a numerical value. Then, for each relational state $\mathtt{z}$ of an RMDP, we can define the reward of $\mathtt{z}$ to be the reward given to that abstract state that generalizes over $\mathtt{z}$, i.e. that subsumes $\mathtt{z}$. If rewards should be unique for states (not additive) one has to ensure that R forms a partition over the complete state space of the RMDP.

Overall, by specifying **i)** a domain C and a set of relations P, **ii)** an abstract action definition, and **iii)** an abstract reward function, one can define RMDPs in a compact way. By only changing the domain C one obtains RMDP variants consisting of different sets of objects, i.e. a *family* of related RMDPs (see van Otterlo, 2009b, for more details).

For solution elements – policies and value functions – we can employ similar representations. In fact, although value functions are different from reward functions, we can use the same representation in terms of a set of abstract states with values. However, other – more compact – representations are possible. To represent a state-action **value function**, Džeroski et al (1998) employed a *relational decision tree*, i.e. a compact partition of the state-action space into regions of the same value. Fig. 8.3(**right**) depicts such a tree, where the root node specifies the action $\texttt{move(D,E)}$. All state-action pairs that try to move block $\texttt{D}$ on $\texttt{E}$, in a state where a block $\texttt{A}$ is not on $\texttt{B}$ and where A is $\texttt{clear}$, get a value 1. Note that, in contrast to propositional trees, node tests share variables.

Policies are mappings from states to actions, and they are often represented using relational decision lists (Mooney and Califf, 1995). For example, consider the following *abstract* policy for a Blocks World, which optimally encodes how to reach states where on(a,b) holds:

$\mathtt{r}_1$: move(X,floor) $\leftarrow$ onTop(X,b)
$\mathtt{r}_2$: move(Y,floor) $\leftarrow$ onTop(Y,a)
$\mathtt{r}_3$: move(a,b) $\leftarrow$ clear(a),clear(b)
$\mathtt{r}_4$: noop $\leftarrow$ on(a,b)

where noop denotes doing nothing. Rules are read from top to bottom: given a state, the first rule where the abstract state applies, generates an optimal action.

Relational RL, either model-free or model-based, has to cope with *representation generation* and *control learning*. A starting point is the RMDP with associated value functions V and Q and policy π that correspond to the problem, and the overall goal of learning is an (optimal) abstract policy $\tilde{\Pi}$.

$$RMDP\ \mathtt{M} = \langle \mathtt{S}, \mathtt{A}, \mathtt{T}, \mathtt{R} \rangle \xrightarrow{\text{control learning + representation learning}} \tilde{\Pi}^* \ : \ \mathtt{S} \rightarrow \mathtt{A} \tag{8.2}$$

Because learning the policy in the ground RMDP is not an option, various routes can be taken in order to find $\tilde{\Pi}^*$. One can first construct *abstract value functions* and deduce a policy from that. One might also inductively learn the policy from optimal ground traces. Relational abstraction over RMDPs induces a number of *structural* learning tasks. In the next sections we survey various approaches.

8.3 Model-Based Solution Techniques

Model-based algorithms rely on the assumption that a full (abstract) model of the RMDP is available, such that it can be used in DP algorithms to compute value functions and policies. A characteristic solution pattern is the following series of purely deductive steps:

$$\begin{array}{ccccccccccc} \tilde{V}^0 \equiv \mathscr{R} & \xrightarrow{\mathbf{D}} & \tilde{V}^1 & \xrightarrow{\mathbf{D}} & \tilde{V}^2 & \xrightarrow{\mathbf{D}} & \ldots & \xrightarrow{\mathbf{D}} & \tilde{V}^k & \xrightarrow{\mathbf{D}} & \tilde{V}^{k+1} \xrightarrow{\mathbf{D}} \ldots \\ \downarrow \mathbf{D} & & \downarrow \mathbf{D} & & \downarrow \mathbf{D} & & & & \downarrow \mathbf{D} & & \downarrow \mathbf{D} \\ \tilde{\Pi}^0 & & \tilde{\Pi}^1 & & \tilde{\Pi}^2 & & & & \tilde{\Pi}^k & & \tilde{\Pi}^{k+1} \end{array} \tag{8.3}$$

The initial specification of the problem can be viewed as a *logical theory*. The initial reward function $\mathscr{R}$ is used as the initial zero-step value function $\tilde{V}^0$. Each subsequent abstract value function $\tilde{V}^{k+1}$ is obtained from $\tilde{V}^k$ by *deduction* (**D**). From each value function $\tilde{V}^k$ a policy $\tilde{\Pi}^k$ can be deduced. At the end of the section we will briefly discuss additional sample-based approaches. Relational model-based solution algorithms generally make explicit use of *Bellman optimality equations*. By turning these into *update rules* they then heavily utilize the KR capabilities to exploit *structure* in both *solutions* and *algorithms*. This has resulted in several relational versions of traditional algorithms such as *value iteration* (VI).

8.3.1 The Structure of Bellman Backups

Let us take a look at the following Bellman update rule , which is usually applied to all states $s \in S$ simultaneously in an iteration:

$$V^{k+1}(s) \;=\; (\mathbf{B}^* V^k)(s) \;=\; R(s) + \gamma \max_a \sum_{s' \in S} T(s,a,s') V^k(s') \qquad (8.4)$$

When used in VI, it extends a k-steps-to-go horizon of the value function V^k to a $(k+1)$-steps-to-go value function V^{k+1} using the *backup operator* $\mathbf{B}^*$. When viewed in more detail, we can distinguish the following sub-operations that together compute the new value $V^{k+1}(s)$ of state $s \in S$:

1. **Matching/Overlap**: First we look at the states s' that are reachable by some action, for which we know the value $V^k(s')$.
2. **Regression**: The value of s is computed by *backing up the value* of a state that can be reached from s by performing an action a. Equivalently, if we know $V^k(s')$ and we know for which states s, $T(s,a,s') > 0$ then we can infer a *partial Q-value* $Q(s,a) = \gamma \cdot T(s,a,s') \cdot V^k(s')$ by reasoning "backwards" from s' to s.
3. **Combination**: Since an action a can have multiple, probabilistic outcomes (i.e. transitions to other states) when applied in state s, the partial Q-values are to be combined by summing them, resulting in "the" Q-value $Q(s,a) = \sum_{s' \in S} T(s,a,s') V^k(s')$. Afterwards the reward $R(s)$ can be added.
4. **Maximization**: In each iteration the highest state value is wanted, and therefore the final value $V^{k+1}(s)$ maximizes over the set of applicable actions.

In a similar way algorithms such as *policy iteration* can be studied.

Intensional Dynamic Programming

Now, classical VI computes values *individually* for each state, even though many states will share the exact same transition pattern to other states. Structured (relational) representations can be used to avoid such redundancy, and compute values for many states simultaneously. Look again at the abstract action definition in Equation 8.1. This compactly specifies many transition probabilities in the form of rules. In a similar way, abstract value functions can represent compactly the values for a set of states using just a single abstract state. **Intensional dynamic programming** (IDP) (van Otterlo, 2009a) makes the use of KR formalisms explicit in MDPs, and provides a unifying framework for structured DP. IDP is defined *representation-independent* but can be instantiated with any atomic, propositional or relational representation. The core of IDP consists of expressing the four mentioned computations (overlap, regression, combination, and maximization) by *representation-dependent* counterparts. Together they are called *decision-theoretic regression* (DTR). A simple instantiation of IDP is *set-based* DP, in which value functions are represented

as discrete sets, and DP employs *set-based* backups. Several other algorithms in the literature can be seen as instantiations of IDP in the propositional context.

A first development in IDP is *explanation-based* RL (EBRL) (Dietterich and Flann, 1997). The inspiration for this work is the similarity between Bellman backups and *explanation-based generalization* (EBG) in *explanation-based learning* (EBL) (see Minton et al, 1989). The representation that is used consists of propositional rules and *region-based* (i.e. for 2D grid worlds) representations. Boutilier et al (2000) introduced *structured* value and policy iteration algorithms (SVI and SPI) based on *propositional trees* for value functions and policies and *dynamic Bayesian networks* for representing the transition function. Later these techniques were extended to work with even more compact *algebraic decision diagrams* (ADD). A third instantiation of IDP are the (hyper)-rectangular partitions of continuous state spaces (Feng et al, 2004). Here, so-called *rectangular piecewise-constant* (RPWC) functions are stored using *kd*-trees (splitting hyperplanes along k axes) for efficient manipulation. IDP techniques are conceptually easy to extend towards POMDPs (Boutilier and Poole, 1996). Let us now go to the relational setting.

8.3.2 Exact Model-Based Algorithms

Exact model-based algorithms for RMDPs implement the four components of DTR in a specific *logical* formalism. The starting point is an abstract action as specified in Equation 8.1. Let us assume our reward function R, which is the initial value function V^0, is $\{\langle \texttt{on(a,b)}, \texttt{on(c,d)}, 10\rangle, \langle \texttt{true}, 0\rangle\}$. Now, in the first step of DTR, we can use the action specification to find abstract states to backup the values in R to. Since the value function is not specified in terms of individual states, we can use **regression** to find out *what are the conditions for the action in order to end up in an abstract state*. Let us consider the first outcome of the action (with probability 0.9), and take $\texttt{on(a,b)}, \texttt{on(c,d)}$ in the value function V^0. **Matching** now amounts to checking whether there is an *overlap* between states expressed by $\texttt{on(X,Y)}, \texttt{cl(X)}, \texttt{cl(Z)}$ and $\texttt{on(a,b)}, \texttt{on(c,d)}$. That is, we want to consider those states that are modeled by both the action effect and the part the value function we are looking at. It turns out there are four ways of computing this overlap (see Fig. 8.4) due to the use of variables (we omit constraints here).

Let us pick the first possibility in the picture, then what happened was that $\texttt{on(X,Y)}$ was matched against $\texttt{on(a,b)}$, which says that we consider the case where the action performed has *caused* a being on b. Matching has generated the substitutions $\texttt{X/a}$ and $\texttt{Y/b}$ which results in the matched state being $\mathbb{Z}' \equiv \texttt{on(a,b)}, \texttt{cl(a)}, \texttt{cl(Z)}$ "plus" $\texttt{on(c,d)}$ which is the part the action did not cause. Now the regression step is *reasoning backwards* through the action, and finding out what the possible abstract state should have been in order for the action $\texttt{move(a,b)}$ to have caused $\mathbb{Z}'$. This is a straightforward computation and results in $\mathbb{Z} \equiv \texttt{on(a,Z)}, \texttt{cl(b)}$ "plus" $\texttt{on(c,d)}$. Now, in terms of the DTR algorithm, we can compute a partial Q-value $Q(\mathbb{Z}, \texttt{move(a,b)}) = \gamma \cdot T(\mathbb{Z}, \texttt{move(a,b)}, \mathbb{Z}') \cdot V^k(\mathbb{Z}') = \gamma \cdot 0.9 \cdot 10$.

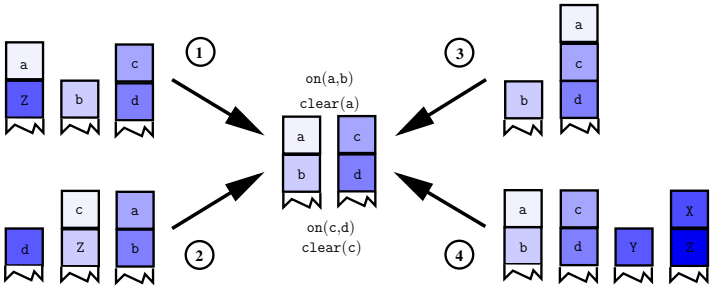

Fig. 8.4: **Examples of regression in a Blocks World**. All four configurations around the center state are possible 'pre-states' when regressing the center state through a standard `move` action. The two left configurations are situations in which either block `a` is put onto `b` (**1**), or block `c` is put onto `d` (**2**). Block `Z` in both these configurations can be either a block or the `floor`. In the top right configuration (**3**) block `a` was on block `c` and is now put onto `b`. The fourth configuration (**4**) is interesting because it assumes that none of the blocks `a`, `b`, `c`, `d` is actually moved, but that there are additional blocks which were moved (i.e. blocks `X` and `Y`).

Doing these steps for all actions, all action outcomes and all abstract states in the value function V^k results in a set of abstract state-action pairs $\langle \mathtt{z}, \mathtt{a}, Q \rangle$ representing partial Q-values. The **combination** of partial Q-values into a Q-function is again done by computing an overlap, in this case between states appearing in the partial Q-function. Let us assume we have computed another partial Q-value $Q(\mathbb{Z}^2, \mathtt{move(a,b)}) = 3$ for $\mathbb{Z}^2 \equiv \mathtt{on(a,Z),cl(b),on(c,d),on(e,f)}$, now for the second outcome of the action (with probability 0.1). Now, in the overlap state $\mathbb{Z}^3 \equiv \mathbb{Z} \wedge \mathbb{Z}^2 \equiv \mathtt{on(a,Z),cl(b),on(c,d),on(e,f)}$ we know that doing action $\mathtt{move(a,b)}$ in state $\mathbb{Z}^3$ has an expected value of $10+3$. The combination of partial Q-values has to be performed for all possible combinations of action outcomes, for all actions.

The last part of DTR is the **maximization**. In our example case, this is fairly easy, since the natural subsumption order on abstract states does most of the work. One can just sort the Q-function $\{\langle \mathbb{Z}, \mathbb{A}, Q \rangle\}$ and take for any state-action pair the first rule that subsumes it. For efficiency reasons, one can remove rules in the Q-function that are subsumed by a higher-valued rule. For other formalisms, the maximization step sometimes involves an extra reasoning effort to maximize over different variable substitutions. The whole procedure can be summarized in the following algorithm, called *first-order decision-theoretic regression* (FODTR):

Require: an abstract value function V^n
1: **for each** action type $\mathbb{A}(\mathbb{X})$ **do**
2: compute $\mathbb{Q}^{\mathbb{A}(\mathbb{X})}_{\mathbb{V}^k} = R \oplus \left[\gamma \otimes \oplus_j \left(\text{prob}(\mathbb{A}(\mathbb{X})) \otimes \text{Regr}(\mathbb{V}^k, \mathbb{A}_j(\mathbb{X}))\right)\right]$
3: $Q^{\mathbb{A}}_{\mathbb{V}^k} = \text{obj-max}\left(\mathbb{Q}^{\mathbb{A}(\mathbb{X})}_{\mathbb{V}^k}\right)$
4: $\mathbb{V}^{k+1} = \max_{\mathbb{A}} \mathbb{Q}^{\mathbb{A}}_{\mathbb{V}^k}$

where $\otimes$ and $\oplus$ denote multiplication and

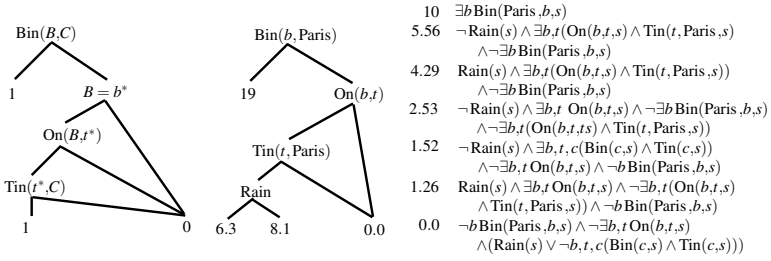

Fig. 8.6: **Examples of structures in exact, first-order IDP**. Both are concrete examples using the logistics domain described in our experimental section. The left two figures are first-order decision diagrams taken from (Wang et al, 2007). On the far left, the transition function (or, *truth value diagram*) is depicted for Bin(B,C) under action choice unload(b*,t*). The right diagram depicts the value function $\mathbb{V}^1$, which turns out to be equivalent to $\mathbb{Q}^1_{\texttt{unload}}$. The formulas on the right represent the final value partition for the logistics domain computed in (Boutilier et al, 2001).

summation over Cartesian products of abstraction-value pairs, Regr denotes a regression operator, and obj-max denotes the maximization over Q-functions.

Currently, four published versions of exact, first-order IDP algorithms have appeared in the literature. The first method is the *symbolic dynamic programming* (SDP) approach (Boutilier et al, 2001) based on the *situation calculus* language (see Reiter, 2001). The second approach, FOVI (Hölldobler and Skvortsova, 2004), is based on the *fluent calculus* (Thielscher, 1998). The third approach is ReBel (Kersting et al, 2004). The most recent approach is based on *first-order decision diagrams* (see Groote and Tveretina, 2003), and will be called FODD here (Wang et al, 2008a).

SDP was the first system but it came without a practical implementation. An important reason was the expressivity of the logic used in SDP, which makes all operations in FODTR computationally expensive. The other three approaches have been shown to be computationally feasible, though at the expense of lower expressivity. Fig. 8.5 shows a computed optimal value function for the goal on(a,b) for a 10-block world (which amounts

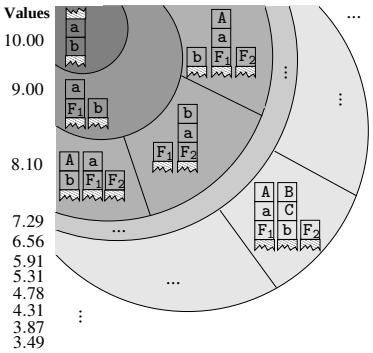

Fig. 8.5: **Blocks World abstract value function** Parts of the abstract value function for on(a,b) after 10 iterations (values rounded). It contains just over a hundred rules. F_i can be a block or a floor block. States more than 10 steps away from the goal get value 0.0.

to almost 60 million ground states). Both ReBel and FOVI operate on conjunctive states with limited negation, whereas FODD uses very compact data structures. This comes with the price of being more expensive in terms of keeping the representation small. Just like its propositional counterpart (SVI) the FODTR procedure in FODD operates on the entire value function, whereas ReBel and FOVI (and SDP) work at the level of individual abstract states. Transition functions in all formalisms are based on the underlying logic; probabilistic STRIPS rules for ReBel, decision diagrams in FODD, and both SDP and FOVI use their underlying fluent and situation calculus action specifications. Figure 8.6 shows some examples of representations used in SDP and FODD. Note how exact, but also how complex, a simple value function in SDP becomes. FODD's decision diagrams are highly compact, but strictly less expressive than SDP state descriptions. *Regression* is built-in as a main reasoning method in SDP and FOVI but for ReBel and FODD specialized procedures were invented.

All four methods perform IDP in first-order domains, *without first grounding the domain*, thereby computing solutions directly on an abstract level. On a slightly more general level, FODTR can be seen as a means to perform (*lifted*) *first-order* reasoning over decision-theoretic values, i.e. as a kind of decision-theoretic logic. One can say that all four methods *deduce* optimal utilities of states (possibly in infinite state spaces) through FODTR, using the action definitions and domain theory as a set of axioms.

Several extensions to the four systems have been described, for example, *policy extraction* and the use of *tabling* (van Otterlo, 2009b), search-based exploration and efficient subsumption tests (Karabaev et al, 2006), *policy iteration* (Wang and Khardon, 2007), *factored* decomposition of first-order MDPs and *additive reward models* (Sanner and Boutilier, 2007), and universally quantified goals (Sanner and Boutilier, 2006).

8.3.3 Approximate Model-Based Algorithms

Since exact, optimal value functions are complex to compute and store (and may be even infinite, in case of an unlimited domain size), several works have developed approximate model-based algorithms for RMDPs. The first type of approach starts from FODTR approaches and then approximates, and a second type uses other means, for example sampling and planning.

The *first-order approximate linear programming* technique (FOALP) (Sanner and Boutilier, 2005) extends the SDP approach, transforming it into an *approximate* value iteration (AVI) algorithm (Schuurmans and Patrascu, 2001). Instead of exactly representing the complete value function, which can be large and fine-grained (and because of that hard to compute), the authors use a fixed set of *basis functions*, comparable to abstract states. That is, a value function can be represented as a *weighted sum of k first-order basis functions* each containing a *small* number of formulae that provide a first-order abstraction (i.e. partition) of the state space. The backup

of a linear combination of such basis functions is simply the linear combination of the FODTR of each basis function. Unlike exact solutions where value functions can grow exponentially in size and where much effort goes into logical simplification of formulas, this feature-based approach must only look for good weights for the case statements. Related to FOALP, the *first-order approximate policy iteration* algorithm (FOAPI) (Sanner and Boutilier, 2006) is a first-order generalization of approximate policy iteration for factored MDPs (e.g. see Guestrin et al, 2003b). It uses the same basis function decomposition of value functions as the FOALP approach, and in addition, an explicit policy representation. It iterates between two phases. In the first, the value function for the current policy is computed, i.e. the weights of the basis functions are computed using LP. The second phase computes the policy from the value function. Convergence is reached if the policy remains stable between successive approximations. Loss bounds for the converged policy generalize directly from the ones for factored MDPs. The PRM approach by Guestrin too provides bounds on policy quality. Yet, these are PAC-bounds obtained under the assumption that the probability of domains falls off exponentially with their size. The FOALP bounds on policy quality apply equally to all domains. FOALP was used for *factored* FOMDPs by Sanner and Boutilier (2007) and applied in the SysAdmin domain. Both FOALP and FOAPI have been entered the *probabilistic* part of the *international planning competition* (IPPC). In a similar AVI framework as FOALP, Wu and Givan (2007) describe a technique for generating first-order features. A simple ILP algorithm employs a *beam-search* in the feature space, guided by how well each feature correlates with the ideal Bellman residual.

A different approach is the approximation to the SDP approach described by Gretton and Thiébaux (2004a,b). The method uses the same basic setup as SDP, but the FODTR procedure is only partially computed. By employing multi-step *classical* regression from the goal states, a number of structures is computed that represent abstract states. The combination and maximization steps are not performed, but instead the structures generated by regression are used as a *hypothesis language* in the higher-order inductive tree-learner ALKEMY (Lloyd, 2003) to induce a tree representing the value function.

A second type of approximate algorithm does not function at the level of abstraction (as in FODTR and extensions) but uses sampling and generalization in the process of generating solutions. That is, one can first generate a solution for one or more (small) ground instances (plans, value functions, policies), and then use inductive generalization methods to obtain generalized solutions. This type of solution was pioneered by the work of Lecoeuche (2001) who used a solved instance of an RMDP to obtain a generalized policy in a dialogue system. Two other methods that use complete, ground RMDP solutions are based on *value function* generalization (Mausam and Weld, 2003) and *policy* generalization (Cocora et al, 2006). Both approaches first use a general MDP solver and both use a relational decision tree algorithm to generalize solutions using relatively simple logical languages. de la Rosa et al (2008) also use a relational decision tree in the ROLLER algorithm to learn *generalized* policies from examples generated by a heuristic planner. The *Relational Envelope-Based Planning* (REBP) (Gardiol and Kaelbling, 2003) uses

a representation of a limited part of the state space (the envelope) which is gradually expanded through sampling just outside the envelope. The aim is to compute a policy, by first generating a good initial plan and use envelope-growing to improve the robustness of the plans incrementally. A more recent extension of the method allows for the representation of the envelope using varying numbers of predicates, such that the representational complexity can be gradually increased during learning (Gardiol and Kaelbling, 2008).

In the following table we summarize the approaches in this section.

method	**representation**	**type**	**algorithm**	**applications**
IDP (van Otterlo, 2009b)	any	E	IDP	—
SDP (Boutilier et al, 2001)	FOL	E	IDP	logistics
ReBel (Kersting et al, 2004)	Conj	E	IDP	BW, logistics
FOVI (Hölldobler and Skvortsova, 2004)	Conj	E	IDP	BW
FODD (Wang et al, 2008a)	FO-ADD	E	IDP	BW
FODD (Wang and Khardon, 2007)	FO-ADD	E	IDP/API	BW
(Guestrin, 2003)	PRM	A	sampling	FreeCraft, SysAdmin
FOALP (Sanner and Boutilier, 2005)	FOF	A	AVI	elevator
FOAPI (Sanner and Boutilier, 2006)	FOF	A	API	PP-IPC
(Gretton and Thiébaux, 2004a)	adaptive FOF	A	FODTR/VFA	logistics, BW
(Wu and Givan, 2007)	adaptive FOF	A	Bellman residuals	PP-IPC
FOLAO* (Karabaev and Skvortsova, 2005)	adaptive FOF	A	FO-LAO*	PP-IPC
REBP (Gardiol and Kaelbling, 2003)	envelopes	A	planning	BW
adaptive **REBP** (Gardiol and Kaelbling, 2008)	envelopes	A	planning	BW

Table 8.1: Main model-based approaches that were discussed. Legend: BW=Blocks World, Cnj=conjunction of logical atoms, Q=Q-learning,PS=prioritized sweeping, E=exact, A=approximate, PP-IPC=planning problems from the planning contest (IPC), FOF=first-order (relational) features, PRM=probabilistic relational model, FOL=first-order logic.

8.4 Model-Free Solutions

Here we review techniques that do not assume availability of an abstract model of the underlying RMDP. Many approaches use the following typical Q-learning pattern:

$$\begin{array}{ccccccccccc}
\tilde{Q}^0 & \xrightarrow{\mathbf{S}} & \{\langle s,a,q\rangle\} & \xrightarrow{\mathbf{I}} & \tilde{Q}^1 & \xrightarrow{\mathbf{S}} & \{\langle s,a,q\rangle\} & \xrightarrow{\mathbf{I}} & \tilde{Q}^2 & \xrightarrow{\mathbf{S}} & \dots \\
\downarrow{\scriptstyle \mathbf{D/I}} & & \| & & \downarrow{\scriptstyle \mathbf{D/I}} & & \| & & \downarrow{\scriptstyle \mathbf{D/I}} & & \\
\tilde{\Pi}^0 & \longrightarrow & \{\langle s,a,q\rangle\} & & \tilde{\Pi}^0 & \longrightarrow & \{\langle s,a,q\rangle\} & & \tilde{\Pi}^0 & \longrightarrow & \{\langle s,a,q\rangle\}
\end{array}$$

Here, an initial abstract Q-function is used to get (**S**) biased learning samples from the RMDP. The samples are then used to learn (or, *induce* (**I**)) a new Q-function structure. Policies can be computed (or, *deduced* (**D**)) from the current Q-function. A restricted variation on this scheme, discussed in the next section, is to fix the logical abstraction level (e.g. $\tilde{Q}$) and only sample the RMDP to get good estimates of the values (e.g. parameters of $\tilde{Q}$).

8.4.1 Value-Function Learning with Fixed Generalization

Several techniques use relational formalisms to create a compact representation of the underlying RMDP, fix that abstraction level, and then apply standard value learning algorithms such as Q-learning. An example abstraction for a three-block Blocks World in the CARCASS representation (van Otterlo, 2003, 2004) is:

state ($\mathbb{S}_1$):	$\big($on(A,B),on(B,floor),on(C,floor),A $\neq$ B,B $\neq$ C$\big)$
actions ($\mathbb{A}_{11},\ldots,\mathbb{A}_{13}$):	move(A,C),move(C,A),move(A,floor)
state ($\mathbb{S}_2$):	$\big($on(A,floor),on(B,floor),on(C,floor), A $\neq$ B,B $\neq$ C, A $\neq$ C$\big)$
actions ($\mathbb{A}_{21},\ldots,\mathbb{A}_{26}$):	move(A,B),move(B,A),move(A,C), move(C,A),move(B,C),move(C,B)
state ($\mathbb{S}_3$):	$\big($on(A,B),on(B,C),on(C,floor), A $\neq$ B, B $\neq$ C, C $\neq$ floor$\big)$
actions ($\mathbb{A}_{31}$):	move(A,floor)

It transforms the underlying RMDP into a much smaller abstract MDP, which can then be solved by (modifications of) RL algorithms. For example, abstract state $\mathbb{S}_3$ models all situations where all blocks are stacked, in which case the only action possible is to move the top block to the floor ($\mathbb{A}_{31}$). In this case three abstract states generalize over 13 RMDP states. The abstraction levels are *exact aggregations* (i.e. partitions) of state-action spaces, and are closely related to *averagers* (Gordon, 1995). van Otterlo (2004) uses this in a Q-learning setting, and also in a *model-based* fashion (prioritized sweeping, Moore and Atkeson, 1993) where also a transition model between abstract states is learned. In essence, what is learned is the best abstract policy among all policies present in the representation, assuming that this *policy space*, can be obtained from a domain expert, or by other means.

Two closely related approaches are LOMDPs by Kersting and De Raedt (2004) and rQ-learning by Morales (2003). LOMDP abstraction levels are very similar to CARCASS, and they use Q-learning and a *logical* TD(λ)-algorithm to learn state values for abstract states. The rQ-framework is based on a *separate* definition of abstract states (*r-states*) and abstract actions (*r-actions*). The product space of r-states and r-actions induces a new (abstract) state-action space over which Q-learning can be performed. All approaches can be used to learn optimal policies in domains where prior knowledge exists. They can also be employed as part of other learning methods, e.g. to learn sub-policies in a given hierarchical policy. A related effort based on *Markov logic networks* was reported by Wang et al (2008b).

Initial investigations into automatically generating the abstractions have been reported (Song and Chen, 2007, 2008), and Morales (2004) employed *behavioral cloning* to learn *r-actions* from sub-optimal traces, generated by a human expert.

AMBIL (Walker et al, 2007) too learns abstract state-action pairs from traces; given an abstraction level, it estimates an approximate model (as in CARCASS) and uses it to generate new abstract state-action pairs. Each value learning iteration a new representation is generated.

method	**representation**	**algorithm**	**applications**
CARCASS (van Otterlo, 2003, 2004)	Cnj + negation	Q, PS	BW, Tic-Tac-Toe
LOMDP (Kersting and De Raedt, 2004)	Cnj	Q, TD	BW
RQ (Morales, 2003)	Cnj + negation	Q	Taxi, Chess, Grids
(Walker et al, 2004)	RF	Q	Robocup simulation
rTD (Asgharbeygi et al, 2006)	RF	TD	Tic-Tac-Toe, Mini-Chess

Table 8.2: Main model-free approaches with static generalization. Legend: BW=Blocks World, Cnj=conjunction of logical atoms, Q=Q-learning, PS=prioritized sweeping, TD=TD-learning, RF=relational features.

A variation on fixed abstractions is to use abstract states as *relational features* for value function approximation. Each abstract state can be seen as a binary feature; whether a state is subsumed by it or not. Walker et al (2004) first generates semi-randomly a set of relational features, and learns the weights of a linear combination of the features to represent the value function for each action. A related technique is *relational temporal difference* learning (rTD), by Asgharbeygi et al (2006). It uses a set of concept definitions (similar to abstract states) that must be supplied by a designer and assigns a utility to each of them. The value function is then represented as a linear combination of the utilities for those concepts, weighted by the number of times a concept occurs. Table 8.2 summarizes this section.

8.4.2 Value Functions with Adaptive Generalization

For *adaptive* generalization, i.e. changing the representation during learning (PIAGET-3), ILP methods are usually used on state(-action) samples derived from interaction with the RMDP (the I(nduction) steps in Equation 8.4). In this section we will describe three types of methods: **i)** those that build logical abstractions (e.g. of value functions), **ii)** those based on distances and kernels and **iii)** probabilistic techniques.

Learning Value Function Abstractions

As said, the Q-RRL method (Džeroski et al, 1998) was the first approach towards model-free RL in RMDPs. It is a straightforward combination of Q-learning and ILP for generalization of Q-functions, and was tested on small deterministic Blocks Worlds. Q-RRL collects experience in the form of state-action pairs with corre-

sponding Q-values. During an episode, actions are taken according to the current policy, based on the current Q-tree. After each episode a decision tree is induced from the example set (see Fig. 8.3(**right**)). Q-trees can employ *background knowledge* (such as about the amount of and heights of towers, the number of blocks).

One problem with Q-functions however, is that they implicitly encode a *distance* to the goal, and they are dependent on the domain size in *families* of RMDPs. A Q-function represents more information than needed for selecting an optimal action. *P-learning* can be used to learn policies from the current Q-function and a training set (Džeroski et al, 2001). For each state s occurring in the training set, all possible actions in that state are evaluated and a P-value is computed as $P(s,a) = 1$ if $a = \arg\max_{a'} Q(s,a')$ and 0 otherwise. The P-tree represents the best policy relative to that Q-tree. In general, it will be less complex and generalize (better) over domains with different numbers of blocks. Independently, Lecoeuche (2001) showed similar results. Cole et al (2003) use a similar setup as Q-RRL, but upgrade the representation language to *higher-order logic* (HOL) (Lloyd, 2003).

An *incremental* extension to Q-RRL is the TG-algorithm (Driessens et al, 2001). It can be seen as a relational extension of the *G*-algorithm (Chapman and Kaelbling, 1991), and it *incrementally* builds Q-trees. Each leaf in a tree is now augmented with statistics about Q-values and the number of positive matches of examples. A node is split when it has seen enough examples and a test on the node's statistics becomes significant with high confidence. This mechanism removes the need for storing, retrieving and updating individual examples. Generating new tests is much more complex than in the propositional case, because the amount of possible splits is essentially unlimited and the number of possibilities grows further down the tree with the number of variables introduced in earlier nodes.

In a subsequent upgrade TGR of TG, Ramon et al (2007) tackle the problem of the *irreversibility* of the splits by adding a *tree restructuring* operation. This includes leaf or subtree pruning, and internal node revision. To carry out these operations statistics are now stored in *all* nodes in the tree. Special care is to be taken with variables in the tree when building and restructuring the tree. Another method that has used restructuring operations from the start is the *relational* UTree (rUTree) algorithm by Dabney and McGovern (2007). rUTree is a relational extension of the UTree algorithm by McCallum (1995). Because rUTree is instance-based, tests can be regenerated when needed for a split such that statistics do not have to be kept for all nodes, as in TGR. Another interesting aspect of rUTree is that it uses *stochastic sampling* (similar to the approach by Walker et al (2004), to cope with the large number of possible tests when splitting a node. Combining these last two aspects shows an interesting distinction with TG (and TGR). Whereas TG must keep all statistics and consider all tests, rUTree considers only a limited, sampled set of possible tests. In return, rUTree must often recompute statistics.

Finally, *first-order* XCS (FOXCS) by Mellor (2008) is a *learning classifier system* (e.g. see Lanzi, 2002) with relational rules. FOXCS' policy representation is similar to that of CARCASS, but each rule is augmented with an *accuracy* and each time an action is required, *all* rules that cover the current state-action pair considered, are taken into account. The accuracy can be used for both action selection,

and adaptation (and creation) of rules by an *evolutionary learning* algorithm. Experimental results on Blocks World instances show that FOXCS can compete with other approaches such as TG.

Generalization using Distances and Kernels

Instead of building logical abstractions several methods use other means for generalization over states modeled as relational interpretations.

The *relational instance based regression* method (RIB) by Driessens and Ramon (2003) uses *instance-based learning* (Aha et al, 1991) on ground relational states. The Q-function is represented by a set of well-chosen experienced examples. To look-up the value of a newly encountered state-action pair, a *distance* is computed between this pair and the stored pairs, and the Q-value of the new pair is computed as an average of the Q-values of pairs that it resembles. Special care is needed to maintain the right set of examples, by throwing away, updating and adding examples. Instance-based regression for Q-learning has been employed for propositional representations before but the challenge in relational domains is defining a suitable *distance* between two interpretations. For RIB, a *domain-specific* distance has to be defined beforehand. For example, in Blocks World problems, the distance between two states is computed by first renaming variables, by comparing the stacks of blocks in the state and finally by the *edit distance* (e.g. how many actions are needed to get from one state to another). Other background knowledge or declarative bias is not used, as the representation consists solely of ground states and actions. García-Durán et al (2008) used a more general instance-based approach in a policy-based algorithm. Katz et al (2008) use a relational instance-based algorithm in a robotic manipulation task. Their similarity measure is defined in terms of isomorphic subgraphs induced by the relational representation.

Later, the methods TG and RIB were combined by Driessens and Džeroski (2005), making use of the strong points of both methods. TG builds an explicit, structural model of the value function and – in practice – can only build up a coarse approximation. RIB is not dependent on a language bias and the instance-based nature is better suited for regression, but it does suffer from large numbers of examples that have to be processed. The combined algorithm – TRENDI builds up a tree like TG but uses an instance-based representation in the leaves of the tree. Because of this, new splitting criteria are needed and both the (language) bias for TG and RIB are needed for TRENDI. However, on deterministic Blocks World examples the new algorithm performs better on some aspects (such as computation time) than its parent techniques. Note that the rUTree algorithm is also a combination of an instance-based representation combined with a logical abstraction level in the form of a tree. No comparison has been reported yet. Rodrigues et al (2008) recently investigated the online behavior of RIB and their results indicate that the number of instances in the system is decreased when per-sample updates are used.

Compared to RIB, Gärtner et al (2003) take a more principled approach in the KBR algorithm to distances between relational states and use *graph kernels* and

Gaussian processes for value function approximation in relational RL. Each state-action pair is represented as a *graph* and a product kernel is defined for this class of graphs. The kernel is wrapped into a Gaussian radial basis function, which can be tuned to regulate the amount of generalization.

Probabilistic Approaches

Two additional, structurally adaptive, algorithms learn and use probabilistic information about the environment to optimize behavior, yet for different purposes.

SVRRL (Sanner, 2005) targets undiscounted, finite-horizon domains in which there is a single terminal reward. This enables viewing the value function as a *probability of success*, such that it can be represented as a *relational naive Bayes network*. The structure and the parameters of this network are learned simultaneously. The parameters can be computed using standard techniques based on the maximum likelihood. Two structure learning approaches are described for SVRRL and in both relational features (ground relational atoms) can be combined into joint features if they are more informative than the independent features' estimates. An extension DM-SVRRL uses *datamining* to focus structure learning on only those parts of the state space that are frequently visited (Sanner, 2006) finds frequently co-occurring features and turns them into joint features, which can later be used to build even larger features.

SVRRL is not based on TD-learning, but on probabilistic reasoning. MARLIE (Croonenborghs et al, 2007b) too uses probabilistic techniques, but employs it for transition model learning. It is one of the very few relational *model-based* RL algorithms (next to CARCASS). It learns how ground relational atoms change from one state to another. For each a *probability tree* is learned incrementally using a modified TG algorithm. Such trees represent for each ground instance of a predicate the probability that it will be true in the next state, given the current state and action. Using the model amounts to look ahead some steps in the future using an existing technique called *sparse sampling*. The original TG algorithm is used to store the Q-value function.

Table 8.3 summarizes some of the main characteristics. Detailed (experimental) comparison between the methods is still a task to be accomplished. A crucial aspect of the methods is the combination of representation and behavior learning. Fixed abstraction levels can provide convergence guarantees but are not flexible. Incremental algorithms such as TG and restructuring approaches such as uTree and TGR provide increasingly flexible function approximators, at the cost of increased computational complexity and extensive bookkeeping.

An important aspect in all approaches is the *representation* used for examples, and how examples are *generalized into abstractions*. Those based on logical abstractions have a number of significant advantages: **i)** they are more easy to generalize over problems of different domain size through the use of variables and **ii)** abstractions are usually more *comprehensible* and *transferrable*. On the other hand, logical abstractions are less suitable for finegrained regression. Methods such as TG have

method	representation	algorithm	applications
Q-RRL (Džeroski et al, 1998)	rel. tree	Q-learning	BW
(Lecoeuche, 2001)	rel. decision list	Q-learning	dialogues
(Cole et al, 2003)	HOL	Q-learning	BW
TG (Driessens et al, 2001)	incremental tree	Q-learning	BW
TGR (Ramon et al, 2007)	adaptive tree	Q-learning	BW
rUTree (Dabney and McGovern, 2007).	adaptive tree	Q-learning	BW, Tsume-Go
RIB (Driessens and Ramon, 2003)	IB	Q-learning	BW
TRENDI (Driessens and Džeroski, 2005)	tree + IB	Q-learning	BW
KBR (Gärtner et al, 2003)	kernels	Q-learning	BW
MARLIE (Croonenborghs et al, 2007b)	prob. rules	model-based	BW
SVRRL (Sanner, 2005)	relational NB	Bayesian	Backgammon

Table 8.3: Main model-free approaches with adaptive generalization that were discussed. Legend: BW=Blocks World, IB=instance based, NB=naive Bayes, HOL=higher-order logic.

severe difficulties with some very simple Blocks World problems. Highly relational problems such as the Blocks World require complex patterns in their value functions and learning these in a typical RL learning process is difficult. Other methods that base their estimates (in part) on instance-based representations, kernels or first-order features are more suitable because they can provide more smooth approximations of the value function.

8.4.3 Policy-Based Solution Techniques

Policy-based approaches have a simple structure: one starts with a policy structure $\tilde{\Pi}^0$, generates samples (**S**) by interaction with the underlying *RMDP*, generates (**D**) a new abstract policy $\tilde{\Pi}^1$, and so on. The general structure is the following:

$$\tilde{\Pi}^0 \xrightarrow{\mathbf{S}} \{\langle s,a,q\rangle\} \xrightarrow{\mathbf{I}} \tilde{\Pi}^1 \xrightarrow{\mathbf{S}} \{\langle s,a,q\rangle\} \xrightarrow{\mathbf{I}} \tilde{\Pi}^2 \longrightarrow \ldots$$

The representation used is usually the decision-rule-like structure we have presented before. An important difference with value learning is that no explicit representations of $\tilde{Q}$ are required. At each iteration of these *approximate policy iteration* algorithms, the current policy is used to gather useful learning experience – which can be samples of state-action pairs, or the amount of reward gathered by that policy – which is then used to generate a new policy structure.

Evolutionary Relational Policy Search

The first type of policy-based approaches are *evolutionary* approaches, which have been used in propositional RL before (Moriarty et al, 1999). Distinct features of

these approaches are that they usually maintain a *population* (i.e. a set) of policy structures, and that they assign a single-valued *fitness* to each policy (or policy rule) based on how it performs on the problem. The fitness is used to combine or modify policies, thereby searching directly in *policy space*.

The Grey system (Muller and van Otterlo, 2005) employs simple relational decision list policies to evolve Blocks World policies. GAPI (van Otterlo and De Vuyst, 2009) is a similar approach based on genetic algorithms, and evolves *probabilistic* relational policies. Gearhart (2003) employs a related technique (*genetic programming* (GP)) in the real-time strategy FreeCraft domain used by Guestrin et al (2003a). Results show that it compares well to Guestrin et al's approach, though it has difficulties with rarely occurring states. Castilho et al (2004) focus on STRIPS planning problems, but unlike Grey for example, each chromosome encodes a full plan, meaning that the approach searches in *plan space*. Both Kochenderfer (2003) and Levine and Humphreys (2003) do search in policy space, and both use a GP algorithm. Levine and Humphreys learn decision list policies from optimal plans generated by a planner, which are then used in a *policy restricted* planner. Kochenderfer allows for *hierarchical* structure in the policies, by simultaneously evolving sub-policies that can call each other. Finally, Baum (1999) describes the Hayek machines that use evolutionary methods to learn policies for Blocks Worlds. An additional approach (FOXCS) was discussed in a previous section.

Policy Search as Classification

A second type of approach uses ILP algorithms to learn the structure of the policy from sampled state-action pairs. This essentially transforms the RL process into a sequence of supervised (classification) learning tasks in which an abstract policy is repeatedly induced from a (biased) set of state-action pairs sampled using the previous policy structure. The challenge is to get good samples either by getting them from optimal traces (e.g. generated by a human expert, or a planning algorithm), or by smart trajectory sampling from the current policy. Both types of approaches are PIAGET-3 , combining structure and parameter learning in a single algorithm.

The model-based approach by Yoon et al (2002) induces policies from optimal traces generated by a planner. The algorithm can be viewed upon as an extension of the work by Martin and Geffner (2000) and Khardon (1999) to stochastic domains. Khardon (1999) studied the induction of deterministic policies for undiscounted, goal-based planning domains, and proved general PAC-bounds on the number of samples needed to obtain policies of a certain quality.

The LRW-API approach by Fern et al (2006) unifies, and extends, the aforementioned approaches into one practical algorithm. LRW-API is based on a concept language (taxonomic syntax), similar to Martin and Geffner (2000)'s approach, and targeted at complex, probabilistic planning domains, as is Yoon et al (2002)'s approach. LWR-API shares its main idea of iteratively inducing policy structures (i.e. *approximate policy iteration*, API) and using the current policy to bias the generation of samples to induce an improved policy. Two main improvements of LRW-

API relative to earlier approaches lie in the sampling process of examples, and in the bootstrapping process. Concerning the first, LRW-API uses *policy rollout* (Boyan and Moore, 1995) to sample the current policy. That is, it estimates all action values for the current policy for a state s by drawing w trajectories of length h, where each trajectory is the result of starting in state s, doing a, and following the policy for $h-1$ more steps. Note that this requires a simulator that can be sampled from any state, at any moment in time. The *sampling width w* and *horizon h* are parameters that trade-off variance and computation time. A second main improvement of LRW-API is the bootstrapping process, which amounts here to *learning from random worlds* (LRW). The idea is to learn complex problems by first starting on simple problems and then iteratively solving more and more complex problem instances. Each problem instance is generated by a *random walk of length n* through the underlying RMDP, and by increasing n problems become more complex.

Policy gradient approaches

The last policy-based technique is based on *policy-gradient* approaches (Sutton et al, 2000). For this, one needs a *parameterized* relational policy to use the gradient of the policy parameters to optimize the policy.

Itoh and Nakamura (2004) describe a first approach for partially observable RMDPs in which policies are represented as a relational decision list. The hand-coded policies make use of a memory consisting of a limited number of memory bits. The algorithm is tested in a maze-like domain where planning is sometimes useful and the problem is to learn *when* it is useful. This is done by treating the policy as stochastic where the probabilities for the policy rules are used for exploration and learned via gradient descent techniques. Gretton (2007a) developed ROPG which learns *temporally extended* policies for domains with *non-Markovian* rewards. Policy gradients are used naturally, and two ways of generating policy rules are used: **i)** a sampling approach, and **ii**) computing rules from the problem specification. Despite slow convergence in general, the RPOG approach learned useful general policies in an elevator scheduling problem. Recently, in the *non-parametric policy gradient* (NPPG) approach, Kersting and Driessens (2008) applied the idea of *gradient boosting*. NPPG builds ensembles of regression models for the value function, thereby expanding the representation while learning (PIAGET-3). Since it just *lifts* the level at which gradients are computed, it works for propositional and relational domains in a similar way.

8.5 Models, Hierarchies, and Bias

The previous two parts of this chapter have surveyed relational upgrades of traditional model-free and model-based algorithms. In this section we take a look at relational techniques used for other aspects of the RMDP framework. We distin-

method	representation	algorithm	applications
Grey (Muller and van Otterlo, 2005)	DL	evo	BW
FOX-CS (Mellor, 2008)	rules	Q-evo	BW
(Gearhart, 2003)	PRM	GP	FreeCraft
(Yoon et al, 2002)	DL	planning	BW
LRW-API Fern et al (2007)	DL	rollout	planning
(Itoh and Nakamura, 2004)	prob. rules	PG/GD	maze
RPOG (Gretton, 2007a)	rules	PG	elevator planning
NPPG (Kersting and Driessens, 2008)	regression trees	PG	BW
GAPI (van Otterlo and De Vuyst, 2009)	prob. DL	evolution	BW, Goldfinder

Table 8.4: Main model-free policy-based approaches that were discussed. Legend: BW=Blocks World, PG=policy gradient, Q=Q-learning, PS=prioritized sweeping, TD=TD-learning, DL=decision list, GD=gradient descent,

guish three groups: **i)** those that learn (probabilistic) models of the RMDP, **ii)** those that impose structure on policies, and **iii)** those that generally bias the learner.

Learning Models of the World

Learning world models is one of the most useful things an agent can do. Transition models embody *knowledge* about the environment that can be exploited in various ways. Such models can be used for more efficient RL algorithms, for model-based DP algorithms, and furthermore, they can often be transferred to other, similar environments. There are several approaches that learn general operator models from interaction. All model-based approaches in Section 8.3 on the contrary, take for granted that a complete, logical model is available.

Usually, *model learning* amounts to learning general operator descriptions, such as the STRIPS rules earlier in this text. However, simpler models can already be very useful and we have seen examples such as the partial models used in MARLIE and the abstract models learned for CARCASS. Another example is the more specialized action model learning employed by Morales (2004) who learns *r*-actions using behavioral cloning. A recent approach by Halbritter and Geibel (2007) is based on graph kernels, which can be used to store transition models without the use of logical abstractions such as used in most action formalisms. A related approach in robotics by Mourão et al (2008) is based on *kernel perceptrons* but is restricted to learning (deterministic) *effects* of STRIPS-like actions.

Learning aspects of (STRIPS) operators from (planning) data is an old problem (e.g. see Vere, 1977) and several older works give partial solutions (e.g. only learning effects of actions). However, learning *full* models, with the added difficulty of *probabilistic* outcomes, is complex since it involves learning both logical structures and parameters from data. Remember the probabilistic STRIPS action we have defined earlier. This `move` action has two different, probabilistic outcomes. Learning such a description from data involves a number of aspects. First, the logical de-

scriptions of the pre- and post-conditions have to be learned from data. Second, the learning algorithm has to infer how many outcomes an action has. Third, probabilities must be estimated for each outcome of the action. For the relational case, early approaches by Gil (1994) and Wang (1995) learn *deterministic* operator descriptions from data, by interaction with simulated worlds. More recent work also targets *incomplete state information* (Wu et al, 2005; Zhuo et al, 2007), or considers *sample complexity* aspects (Walsh and Littman, 2008).

For general RMDPs though, we need *probabilistic* models. In the propositional setting, the earliest learning approach was described by Oates and Cohen (1996). For the relational models in the context of RMDPs the first approach was reported by Pasula et al (2004), using a three-step greedy search approach. First, a search is performed through the set of rule sets using standard ILP operators. Second, it finds the best set of outcomes, given a context and an action. Third, it learns a probability distribution over sets of outcomes. The learning process is *supervised* as it requires a dataset of state-action-state pairs taken from the domain. As a consequence, the rules are only valid on this set, and care has to be taken that it is representative for the domain. Experiments on Blocks Worlds and logistics domains show the robustness of the approach. The approach was later extended by Zettlemoyer et al (2005) who added *noise outcomes*, i.e. outcomes which are difficult to model exactly, but which do happen (like knocking over a blocks tower and scattering all blocks on the floor). Another approach was introduced by Safaei and Ghassem-Sani (2007), which works *incrementally* and combines planning and learning.

Hierarchies and Agents

Hierarchical approaches can be naturally incorporated into relational RL, yet not many techniques have been reported so far, although some cognitive architectures (and *sapient* agents, cf. van Otterlo et al, 2007)) share these aspects. An advantage of *relational* HRL is that parameterizations of sub-policies and goals naturally arise, through logical variables. For example, a Blocks World task such as `on(X,Y)`, where `X` and `Y` can be instantiated using any two blocks, can be *decomposed* into two tasks. First, all blocks must be removed from `X` and `Y` and then `X` and `Y` should be moved on top of each other. Note that the first task also consists of two subtasks, supporting even further decomposition. Now, by first learning policies for each of these subtasks, the individual learning problems for each of these subtasks are much simpler and the resulting skills might be reused. Furthermore, learning such subtasks can be done by any of the model-free algorithms in Section 8.4. Depending on the representation that is used, policies can be structured into hierarchies, facilitating learning in more complex problems.

Simple forms of *options* are straightforward to model relationally (see Croonenborghs et al, 2007a, for initial ideas). Driessens and Blockeel (2001) presented an approach based on the Q-RRL algorithm to learn two goals *simultaneously*, which can be considered a simple form of hierarchical decomposition. Aycenina (2002) uses the original Q-RRL system to build a complete system. A number of subgoals

is given and separate policies are learned to achieve them. When learning a more complex task, instantiated sub-policies can be used as new actions. A related system by Roncagliolo and Tadepalli (2004) uses batch learning on a set of examples to learn values for a given relational hierarchy. Andersen (2005) presents the most thorough investigation using MAXQ hierarchies adapted to relational representations. The work uses the Q-RRL framework to induce local Q-trees and P-trees, based on a manually constructed hierarchy of subgoals. A shortcoming of hierarchical relational systems is still that they assume the hierarchy is given beforehand. Two related systems combine planning and learning, which can be considered as generating hierarchical abstractions by planning, and using RL to learn concrete policies for behaviors. The approach by Ryan (2002) uses a planner to build a high-level task hierarchy, after which RL is uses to learn the subpolicies. The method by Grounds and Kudenko (2005) is based on similar ideas. Also in cognitive architectures, not much work has been reported yet, with notable exceptions of model-learning in Icarus (Shapiro and Langley, 2002) and RL in SOAR (Nason and Laird, 2004).

Hierarchical learning can be seen as first decomposing a policy, and then do learning. *Multi-agent* (Wooldridge, 2002) decompositions are also possible. Letia and Precup (2001) report on multiple agents, modeled as *independent reinforcement learners* who do not communicate, but act in the same environment. Programs specify initial plans and knowledge about the environment, and complex actions induce semi- MDPs, and learning is performed by model-free RL methods based on *options*. A similar approach by Hernandez et al (2004) is based on the *belief-desires-intentions* model. Finzi and Lukasiewicz (2004a) introduce GTGolog, a *game-theoretic* language, which integrates explicit agent programming with game-theoretic multi-agent planning in Markov Games. Along this direction Finzi and Lukasiewicz (2004b) introduce *relational Markov Games* which can be used to abstract over multi-agent RMDPs and to compute *Nash policy pairs*.

Bias

Solution algorithms for RMDPs can be helped in many ways. A basic distinction is between helping the solution of the current problem (*bias*, *guidance*), and helping the solution of related problems (*transfer*).

Concerning the first, one idea is to supply a policy to the learner that can *guide* it towards useful areas in the state space (Driessens and Džeroski, 2002) by generating semi-optimal traces of behavior that can be used as experience by the learner. Related to guidance is the usage of *behavioral cloning* based on human generated traces, for example by (Morales, 2004) and Cocora et al (2006), the use of optimal plans by Yoon et al (2002), or the random walk approach by Fern et al (2006), which uses a guided exploration of *domain instantiations*. In the latter, first only easy instantiations are generated and the difficulty of the problems is increased in accordance with the current policy's quality. Domain sampling was also used by Guestrin et al (2003a). Another way to help the learner is by structuring the domain itself, for example by imposing a strong *topological structure* (Lane and Wilson, 2005). Since

many of these guidance techniques are essentially representation-independent, there are many more existing (propositional) algorithms to be used for RMDPs.

A second way to help the learner, is by *transferring* knowledge from other tasks. *Transfer learning* – in a nutshell – is leveraging learned knowledge on a *source task* to improve learning on a related, but different *target task*. It is particularly appropriate for learning agents that are meant to persist over time, changing flexibly among tasks and environments. Rather than having to learn each task from scratch, the goal is to take advantage of its past experience to speed up learning (see Stone, 2007). Transfer is much representation-dependent, and the use of (declarative) FOL formalisms in relational RL offers good opportunities.

In the more restricted setting of (relational) RL there are several possibilities. For example, the source and target problems can differ in the goal that must be reached, but possibly the transition model and reward model can be transferred. Sometimes a specific state abstraction can be transferred between problems (e.g. Walsh et al, 2006). Or, possibly some knowledge about actions can be transferred, although they can have slightly different effects in the source and target tasks. Sometimes a complete policy can be transferred, for example a *stacking* policy for a Blocks World can be transferred between worlds of varying size. Another possibility is to transfer solutions to *subproblems*, e.g. a sub-policy. Very often in relational domains, transfer is possible by the intrinsic nature of relational representations alone. That is, in Blocks Worlds, in many cases a policy that is learned for a task with n blocks will work for $m > n$ blocks.

So far, a few approaches have approached slightly more general notions of transfer in relational RL. Several approaches are based on transfer of learned skills in hierarchical decompositions, e.g. (Croonenborghs et al, 2007a) and (Torrey et al, 2007), and an extension of the latter using Markov logic networks (Torrey et al, 2008). Explicit investigations into transfer learning were based on methods we have discussed such as in rTD (Stracuzzi and Asgharbeygi, 2006) and the work by García-Durán et al (2008) on transferring instance-based policies in deterministic planning domains.

8.6 Current Developments

In this chapter so far, we have discussed an important selection of the methods described in (van Otterlo, 2009b). In this last section we briefly discuss some recent approaches and trends in solution techniques for RMDPs that appeared very recently. These topics also mark at least two areas with lots of open problems and potential; general logical-probabilistic engines, and partially observable problems.

Probabilistic Inference Frameworks

In many solution techniques we have discussed, RMDPs are tackled by upgrading MDP-specific representations, solution algorithms and methodology to the relational domain. Another route that has catched considerable interest nowadays comes from seeing decision-theoretic problems as problems that can be solved by general probabilistic and decision-theoretic reasoning engines. This direction starts with first viewing *planning* as a specific instance of *probabilistic inference* (Toussaint, 2009). For any MDP one can find a formulation where basically the task is to find that policy π that will have the *highest probability* of reaching the goal. This connection between planning and probabilistic inference opens up many possibilities to setup a probabilistic planning problem as a general probabilistic inference problem and use optimized strategies for finding optimal plans.

Lang and Toussaint (2009) directly implement these ideas in the context of learned rules based on the technique by Pasula et al (2004). The relational probabilistic transition model is then transformed into a Bayesian network and standard inference techniques are used for (re-)planning. Later it was extended by incorporating both *forward* and *backward* planning Lang and Toussaint (2010). The transformation into propositional Bayesian networks makes it also easy to connect to lower-level probabilistic reasoning patterns, resulting in an integrated framework for high-level relational learning and planning with low-level behaviors on a real robotic arm (Toussaint et al, 2010). A downside of translating to Bayesian networks though, is that these networks typically blow up in size, and planning is still essentially done on a propositional level. A more logically-oriented approach was described by Thon et al (2009) who used SRL to learn probabilistic action models which can be directly translated into standard relational planning languages.

A more generic solution which is developing currently, is to extend SRL systems into full probabilistic *programming languages* with decision-theoretic capabilities. Many ideas were developed early on, mainly in the *independent choice logic* (Poole, 1997), and recently approaches appear that employ state-of-the-art reasoning and/or learning techniques. The DT-ProbLog language is the first probabilistic decision-theoretic programming capable of efficiently computing optimal (and approximate) plans (Van den Broeck et al, 2010). It is based on the ProbLog language for probabilistic reasoning, but it extends it with *reward facts*. Based on a set of *decision facts*, DT-ProbLog computes possible plans and stores their values compactly in algebraic decision diagrams (ADD). Optimization of the ADD gives the optimal (or approximated) set of decisions. So far, the language deals efficiently with *simultaneous* actions, i.e. multiple decisions have to be made in a problem, but the sequential aspect is not yet taken into account (such problems can be modeled and solved, but solving them efficiently is a step to be taken). It successfully makes *direct marketing* decisions for a huge social network. A related approach based on Markov logic networks (Nath and Domingos, 2009) can model similar problems, but only supports approximate reasoning. Another approach by Chen and Muggleton (2010) is based on stochastic logic programming but lacks an efficient implementation. Yet another approach along the same direction is that by Saad (2008) which is based on *an-*

swer set programming. The central hypothesis in all these recent approaches is that current progress in SRL systems will become available to solve decision-theoretic problems such as RMDPs too. A related direction of employing RL in programming languages (Simpkins et al, 2008) is expected to become important in that respect as well.

Relational POMDPs

Virtually all methods presented so far assume fully-observable first-order MDPs. Extensions towards more complex models beyond this Markov assumption are almost unexplored so far, and in (van Otterlo, 2009b) we mentioned some approaches that have went up this road.

Wingate et al (2007) present the first steps towards relational KR in *predictive representations of states* (Littman et al, 2001). Although the representation is still essentially propositional, they do capture some of Blocks World structure in a much richer framework than MDPs. Zhao and Doshi (2007) introduce a semi-Markov extension to the situation calculus approach in SDP in the Haley system for *web-based services*. Although no algorithm for solving the induced first-order SMDP is given, the approach clearly shows that the formalization can capture useful temporal structures. Gretton (2007a,b) compute *temporally extended policies* for domains with non-Markovian rewards, using a *policy gradient* approach and we have discussed it in the previous chapter.

The first contribution to the solution of *first-order* POMDPs is given by Wang (2007). Although modeling POMDPs using FOL formalisms has been done before (e.g. see Geffner and Bonet, 1998; Wang and Schmolze, 2005), Wang is the first to upgrade an existing POMDP solution algorithm to the first-order case. It takes the FODD formalism we have discussed earlier and extends it to model *observations*, conditioned on states. Based on the clear connections between regression-based backups in IDP algorithms and value backups over belief states, Wang upgrades the *incremental pruning* algorithm (see Kaelbling et al, 1998) to the first-order case.

Recently some additional efforts into relational POMDPs have been described, for example by Lison (2010) in dialogue systems, and by both Wang and Khardon (2010) and Sanner and Kersting (2010) who describe basic algorithms for such POMDPs along the lines of IDP.

New Methods and Applications

In addition to inference engines and POMDPs, other methods have appeared recently. In the previous we have already mentioned the evolutionary technique GAPI (van Otterlo and De Vuyst, 2009), and furthermore Neruda and Slusny (2009) provide a performance comparison between relational RL and evolutionary techniques.

New techniques for learning models include *instance based* techniques for deterministic action models in the SOAR cognitive architecture (Xu and Laird, 2010),

and *incremental* learning of action models in the context of noise by Rodrigues et al (2010). Both represent new steps towards learning transition models that can be used for planning and FODTR. A related technique by Vargas-Govea and Morales (2009) learns *grammars* for sequences of actions, based on sequences of low-level sensor readings in a robotic context.

In fact, domains such as robotics are interesting application areas for relational RL. Both Vargas and Morales (2008) and Hernández and Morales (2010) apply relational techniques for navigation purposes in a robotic domain. The first approach learns *teleo-reactive programs* from traces using behavioral cloning. The second combines relational RL and continuous actions. Another interesting area for relational RL is *computer vision*, and initial work in this direction is reported by Hming and Peters (2009) who apply it for object recognition.

Just like in (van Otterlo, 2009b) we also pay attention to recent PhD theses that were written on topics in relational RL. Five more theses have appeared, on model-assisted approaches by Croonenborghs (2009), on transfer learning by Torrey (2009), on object-oriented representations in RL by Diuk (2010), on efficient model learning by Walsh (2010) and on FODTR by Joshi (2010).

8.7 Conclusions and Outlook

In this chapter we have surveyed the field of relational RL, summarizing the main techniques discussed in (van Otterlo, 2009b). We have discussed large subfields such as model-based and model-free algorithms, and in addition hierarchies, models, POMDPs and much more.

There are many future directions for relational RL. For one part, it can be noticed that not many techniques make use of efficient data structures. Some model-based techniques use binary (or algebraic) data structures, but many other methods may benefit since relational RL is a computationally complex task. The use of more efficient inference engines has started with the development of languages such as DT-Problog, but it is expected that the next few years more efforts along these lines will be developed in the field of SRL. In the same direction, POMDP techniques and probabilistic model learning can be explored too from the viewpoint of SRL.

In terms of application areas, robotics, computer vision, and web and social networks may yield interesting problems. The connection to *manipulation* and *navigation* in robotics is easily made and not many relational RL techniques have been used so far. Especially the grounding and anchoring of sensor data to relational representations is a hard problem and largely unsolved.

References

Aha D, Kibler D, Albert M (1991) Instance-based learning algorithms. Machine Learning 6(1):37–66

Alpaydin E (2004) Introduction to Machine Learning. The MIT Press, Cambridge, Massachusetts

Andersen CCS (2005) Hierarchical relational reinforcement learning. Master's thesis, Aalborg University, Denmark

Asgharbeygi N, Stracuzzi DJ, Langley P (2006) Relational temporal difference learning. In: Proceedings of the International Conference on Machine Learning (ICML), pp 49–56

Aycenina M (2002) Hierarchical relational reinforcement learning. Stanford Doctoral Symposium, unpublished

Baum EB (1999) Toward a model of intelligence as an economy of agents. Machine Learning 35(2):155–185

Baum EB (2004) What is Thought? The MIT Press, Cambridge, Massachusetts

Bergadano F, Gunetti D (1995) Inductive Logic Programming: From Machine Learning to Software Engineering. The MIT Press, Cambridge, Massachusetts

Bertsekas DP, Tsitsiklis J (1996) Neuro-Dynamic Programming. Athena Scientific, Belmont, MA

Boutilier C, Poole D (1996) Computing optimal policies for partially observable markov decision processes using compact representations. In: Proceedings of the National Conference on Artificial Intelligence (AAAI), pp 1168–1175

Boutilier C, Dean T, Hanks S (1999) Decision theoretic planning: Structural assumptions and computational leverage. Journal of Artificial Intelligence Research 11:1–94

Boutilier C, Dearden RW, Goldszmidt M (2000) Stochastic dynamic programming with factored representations. Artificial Intelligence 121(1–2):49–107

Boutilier C, Reiter R, Price B (2001) Symbolic dynamic programming for first-order MDP's. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp 690–697

Boyan JA, Moore AW (1995) Generalization in reinforcement learning: Safely approximating the value function. In: Proceedings of the Neural Information Processing Conference (NIPS), pp 369–376

Brachman RJ, Levesque HJ (2004) Knowledge Representation and Reasoning. Morgan Kaufmann Publishers, San Francisco, CA

Castilho M, Kunzle LA, Lecheta E, Palodeto V, Silva F (2004) An investigation on genetic algorithms for generic STRIPS planning. In: IBERAMIA 2004, Lecture Notes in Computer Science, vol 3315, pp 185–194

Chapman D, Kaelbling LP (1991) Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp 726–731

Chen J, Muggleton S (2010) Decision-theoretic logic programs. In: Proceedings of the International Conference on Inductive Logic Programming (ILP)

Cocora A, Kersting K, Plagemann C, Burgard W, De Raedt L (2006) Learning relational navigation policies. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

Cole J, Lloyd JW, Ng KS (2003) Symbolic learning for adaptive agents. In: Proceedings of the Annual Partner Conference, Smart Internet Technology Cooperative Research Centre, http://csl.anu.edu.au/jwl/crc_paper.pdf

Croonenborghs T (2009) Model-assisted approaches for relational reinforcement learning. PhD thesis, Department of Computer Science, Catholic University of Leuven, Belgium

Croonenborghs T, Driessens K, Bruynooghe M (2007a) Learning relational options for inductive transfer in relational reinforcement learning. In: Proceedings of the International Conference on Inductive Logic Programming (ILP)

Croonenborghs T, Ramon J, Blockeel H, Bruynooghe M (2007b) Online learning and exploiting relational models in reinforcement learning. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp 726–731

Dabney W, McGovern A (2007) Utile distinctions for relational reinforcement learning. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp 738–743

de la Rosa T, Jimenez S, Borrajo D (2008) Learning relational decision trees for guiding heuristic planning. In: Proceedings of the International Conference on Artificial Intelligence Planning Systems (ICAPS)

De Raedt L (2008) Logical and Relational Learning. Springer

Dietterich TG, Flann NS (1997) Explanation-based learning and reinforcement learning: A unified view. Machine Learning 28(503):169–210

Diuk C (2010) An object-oriented representation for efficient reinforcement learning. PhD thesis, Rutgers University, Computer Science Department

Diuk C, Cohen A, Littman ML (2008) An object-oriented representation for efficient reinforcement learning. In: Proceedings of the International Conference on Machine Learning (ICML)

Driessens K, Blockeel H (2001) Learning Digger using hierarchical reinforcement learning for concurrent goals. In: Proceedings of the European Workshop on Reinforcement Learning (EWRL)

Driessens K, Džeroski S (2002) Integrating experimentation and guidance in relational reinforcement learning. In: Proceedings of the Nineteenth International Conference on Machine Learning, pp 115–122

Driessens K, Džeroski S (2005) Combining model-based and instance-based learning for first order regression. In: Proceedings of the International Conference on Machine Learning (ICML), pp 193–200

Driessens K, Ramon J (2003) Relational instance based regression for relational reinforcement learning. In: Proceedings of the International Conference on Machine Learning (ICML), pp 123–130

Driessens K, Ramon J, Blockeel H (2001) Speeding up relational reinforcement learning through the use of an incremental first order decision tree algorithm. In: Proceedings of ECML - European Conference on Machine Learning, Springer-Verlag, Lecture Notes in Artificial Intelligence, vol 2167, pp 97–108

Džeroski S, De Raedt L, Blockeel H (1998) Relational reinforcement learning. In: Shavlik J (ed) Proceedings of the International Conference on Machine Learning (ICML), pp 136–143

Džeroski S, De Raedt L, Driessens K (2001) Relational reinforcement learning. Machine Learning 43:7–52

Feng Z, Dearden RW, Meuleau N, Washington R (2004) Dynamic programming for structured continuous Markov decision problems. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pp 154–161

Fern A, Yoon SW, Givan R (2006) Approximate policy iteration with a policy language bias: Solving relational markov decision processes. Journal of Artificial Intelligence Research (JAIR) 25:75–118, special issue on the International Planning Competition 2004

Fern A, Yoon SW, Givan R (2007) Reinforcement learning in relational domains: A policy-language approach. The MIT Press, Cambridge, Massachusetts

Fikes RE, Nilsson NJ (1971) STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 2(2)

Finney S, Gardiol NH, Kaelbling LP, Oates T (2002) The thing that we tried Didn't work very well: Deictic representations in reinforcement learning. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pp 154–161

Finzi A, Lukasiewicz T (2004a) Game-theoretic agent programming in Golog. In: Proceedings of the European Conference on Artificial Intelligence (ECAI)

Finzi A, Lukasiewicz T (2004b) Relational Markov games. In: European Conference on Logics in Artificial Intelligence (JELIA), Lecture Notes in Computer Science, vol 3229, pp 320–333

García-Durán R, Fernández F, Borrajo D (2008) Learning and transferring relational instance-based policies. In: Proceedings of the AAAI-2008 Workshop on Transfer Learning for Complex Tasks

Gardiol NH, Kaelbling LP (2003) Envelope-based planning in relational MDPs. In: Proceedings of the Neural Information Processing Conference (NIPS)

Gardiol NH, Kaelbling LP (2008) Adaptive envelope MDPs for relational equivalence-based planning. Tech. Rep. MIT-CSAIL-TR-2008-050, MIT CS & AI Lab, Cambridge, MA

Gärtner T, Driessens K, Ramon J (2003) Graph kernels and Gaussian processes for relational reinforcement learning. In: Proceedings of the International Conference on Inductive Logic Programming (ILP)

Gearhart C (2003) Genetic programming as policy search in Markov decision processes. In: Genetic Algorithms and Genetic Programming at Stanford, pp 61–67

Geffner H, Bonet B (1998) High-level planning and control with incomplete information using pomdps. In: Proceedings Fall AAAI Symposium on Cognitive Robotics

Gil Y (1994) Learning by experimentation: Incremental refinement of incomplete planning domains. In: Proceedings of the International Conference on Machine Learning (ICML)

Gordon GJ (1995) Stable function approximation in dynamic programming. In: Proceedings of the International Conference on Machine Learning (ICML), pp 261–268

Gretton C (2007a) Gradient-based relational reinforcement-learning of temporally extended policies. In: Proceedings of the International Conference on Artificial Intelligence Planning Systems (ICAPS)

Gretton C (2007b) Gradient-based relational reinforcement learning of temporally extended policies. In: Workshop on Artificial Intelligence Planning and Learning at the International Conference on Automated Planning Systems

Gretton C, Thiébaux S (2004a) Exploiting first-order regression in inductive policy selection. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pp 217–225

Gretton C, Thiébaux S (2004b) Exploiting first-order regression in inductive policy selection (extended abstract). In: Proceedings of the Workshop on Relational Reinforcement Learning at ICML'04

Groote JF, Tveretina O (2003) Binary decision diagrams for first-order predicate logic. The Journal of Logic and Algebraic Programming 57:1–22

Grounds M, Kudenko D (2005) Combining reinforcement learning with symbolic planning. In: Proceedings of the Fifth European Workshop on Adaptive Agents and Multi-Agent Systems

Guestrin C (2003) Planning under uncertainty in complex structured environments. PhD thesis, Computer Science Department, Stanford University

Guestrin C, Koller D, Gearhart C, Kanodia N (2003a) Generalizing plans to new environments in relational MDPs. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp 1003–1010

Guestrin C, Koller D, Parr R, Venkataraman S (2003b) Efficient solution algorithms for factored MDPs. Journal of Artificial Intelligence Research (JAIR) 19:399–468

Halbritter F, Geibel P (2007) Learning models of relational MDPs using graph kernels. In: Proceedings of the Mexican Conference on Artificial Intelligence (MICAI), pp 409–419

Hanks S, McDermott DV (1994) Modeling a dynamic and uncertain world I: Symbolic and probabilistic reasoning about change. Artificial Intelligence 66(1):1–55

Hernandez AG, El Fallah-Seghrouchni A, Soldano H (2004) Learning in BDI multi-agent systems. In: Proceedings of CLIMA IV - Computational Logic in Multi-Agent Systems

Hernández J, Morales EF (2010) Relational reinforcement learning with continuous actions by combining behavioral cloning and locally weighted regression. Journal of Intelligent Systems and Applications 2:69–79

Hming K, Peters G (2009) Relational reinforcement learning applied to appearance-based object recognition. In: Palmer-Brown D, Draganova C, Pimenidis E, Mouratidis H (eds) Engineering Applications of Neural Networks, Communications in Computer and Information Science, vol 43, Springer Berlin Heidelberg, pp 301–312

Hölldobler S, Skvortsova O (2004) A logic-based approach to dynamic programming. In: Proceedings of the AAAI Workshop on Learning and Planning in Markov Processes - Advances and Challenges

Itoh H, Nakamura K (2004) Towards learning to learn and plan by relational reinforcement learning. In: Proceedings of the ICML Workshop on Relational Reinforcement Learning

Joshi S (2010) First-order decision diagrams for decision-theoretic planning. PhD thesis, Tufts University, Computer Science Department

Kaelbling LP, Littman ML, Cassandra AR (1998) Planning and acting in partially observable stochastic domains. Artificial Intelligence 101:99–134

Kaelbling LP, Oates T, Gardiol NH, Finney S (2001) Learning in worlds with objects. In: The AAAI Spring Symposium

Karabaev E, Skvortsova O (2005) A heuristic search algorithm for solving first-order MDPs. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)

Karabaev E, Rammé G, Skvortsova O (2006) Efficient symbolic reasoning for first-order MDPs. In: ECAI Workshop on Planning, Learning and Monitoring with Uncertainty and Dynamic Worlds

Katz D, Pyuro Y, Brock O (2008) Learning to manipulate articulated objects in unstructured environments using a grounded relational representation. In: Proceedings of Robotics: Science and Systems IV

Kersting K, De Raedt L (2004) Logical Markov decision programs and the convergence of TD(λ). In: Proceedings of the International Conference on Inductive Logic Programming (ILP)

Kersting K, Driessens K (2008) Non-parametric gradients: A unified treatment of propositional and relational domains. In: Proceedings of the International Conference on Machine Learning (ICML)

Kersting K, van Otterlo M, De Raedt L (2004) Bellman goes relational. In: Proceedings of the International Conference on Machine Learning (ICML)

Khardon R (1999) Learning to take actions. Machine Learning 35(1):57–90

Kochenderfer MJ (2003) Evolving hierarchical and recursive teleo-reactive programs through genetic programming. In: EuroGP 2003, Lecture Notes in Computer Science, vol 2610, pp 83–92

Lane T, Wilson A (2005) Toward a topological theory of relational reinforcement learning for navigation tasks. In: Proceedings of the International Florida Artificial Intelligence Research Society Conference (FLAIRS)

Lang T, Toussaint M (2009) Approximate inference for planning in stochastic relational worlds. In: Proceedings of the International Conference on Machine Learning (ICML)

Lang T, Toussaint M (2010) Probabilistic backward and forward reasoning in stochastic relational worlds. In: Proceedings of the International Conference on Machine Learning (ICML)

Langley P (2006) Cognitive architectures and general intelligent systems. AI Magazine 27:33–44

Lanzi PL (2002) Learning classifier systems from a reinforcement learning perspective. Soft Computing 6:162–170

Lecoeuche R (2001) Learning optimal dialogue management rules by using reinforcement learning and inductive logic programming. In: Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)

Letia I, Precup D (2001) Developing collaborative Golog agents by reinforcement learning. In: Proceedings of the 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01), IEEE Computer Society

Levine J, Humphreys D (2003) Learning action strategies for planning domains using genetic programming. In: EvoWorkshops 2003, Lecture Notes in Computer Science, vol 2611, pp 684–695

Lison P (2010) Towards relational POMDPs for adaptive dialogue management. In: ACL '10: Proceedings of the ACL 2010 Student Research Workshop, Association for Computational Linguistics, Morristown, NJ, USA, pp 7–12

Littman ML, Sutton RS, Singh S (2001) Predictive representations of state. In: Proceedings of the Neural Information Processing Conference (NIPS)

Lloyd JW (2003) Logic for Learning: Learning Comprehensible Theories From Structured Data. Springer-Verlag

Martin M, Geffner H (2000) Learning generalized policies in planning using concept languages. In: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)

Mausam, Weld DS (2003) Solving relational MDPs with first-order machine learning. In: Workshop on Planning under Uncertainty and Incomplete Information at ICAPS'03

McCallum RA (1995) Instance-based utile distinctions for reinforcement learning with hidden state. In: Proceedings of the International Conference on Machine Learning (ICML), pp 387–395

Mellor D (2008) A learning classifier system approach to relational reinforcement learning. In: 10th and 11th International Workshops on Learning Classifier Systems (IWLCS 2006–2007), Revised Selected Papers, Springer, Lecture Notes in Computer Science, vol 4998, pp 169–188

Minker J (2000) Logic-Based Artificial Intelligence. Kluwer Academic Publishers Group, Dordrecht, The Netherlands

Minton S, Carbonell J, Knoblock CA, Kuokka DR, Etzioni O, Gil Y (1989) Explanation-based learning: A problem solving perspective. Artificial Intelligence 40(1–3):63–118

Mooney RJ, Califf ME (1995) Induction of first-order decision lists: Results on learning the past tense of english verbs. Journal of Artificial Intelligence Research (JAIR) 3:1–24

Moore AW, Atkeson CG (1993) Prioritized sweeping: Reinforcement learning with less data and less time. Machine Learning 13(1):103–130

Morales EF (2003) Scaling up reinforcement learning with a relational representation. In: Proceedings of the Workshop on Adaptability in Multi-Agent Systems at AORC'03, Sydney

Morales EF (2004) Learning to fly by combining reinforcement learning with behavioral cloning. In: Proceedings of the International Conference on Machine Learning (ICML), pp 598–605

Moriarty DE, Schultz AC, Grefenstette JJ (1999) Evolutionary algorithms for reinforcement learning. Journal of Artificial Intelligence Research (JAIR) 11:241–276

Mourão K, Petrick RPA, Steedman M (2008) Using kernel perceptrons to learn action effects for planning. In: Proceedings of the International Conference on Cognitive Systems (CogSys), pp 45–50

Muller TJ, van Otterlo M (2005) Evolutionary reinforcement learning in relational domains. In: Proceedings of the 7th European Workshop on Reinforcement Learning

Nason S, Laird JE (2004) Soar-RL: Integrating reinforcement learning with soar. In: Proceedings of the Workshop on Relational Reinforcement Learning at ICML'04

Nath A, Domingos P (2009) A language for relational decision theory. In: International Workshop on Statistical Relational Learning (SRL)

Neruda R, Slusny S (2009) Performance comparison of two reinforcement learning algorithms for small mobile robots. International Journal of Control and Automation 2(1):59–68

Oates T, Cohen PR (1996) Learning planning operators with conditional and probabilistic effects. In: Planning with Incomplete Information for Robot Problems: Papers from the 1996 AAAI Spring Symposium, pp 86–94

Pasula HM, Zettlemoyer LS, Kaelbling LP (2004) Learning probabilistic planning rules. In: Proceedings of the International Conference on Artificial Intelligence Planning Systems (ICAPS)

Poole D (1997) The independent choice logic for modeling multiple agents under uncertainty. Artificial Intelligence 94:7–56

Ramon J, Driessens K, Croonenborghs T (2007) Transfer learning in reinforcement learning problems through partial policy recycling. In: Proceedings of the European Conference on Machine Learning (ECML)

Reiter R (2001) Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. The MIT Press, Cambridge, Massachusetts

Rodrigues C, Gerard P, Rouveirol C (2008) On and off-policy relational reinforcement learning. In: Late-Breaking Papers of the International Conference on Inductive Logic Programming

Rodrigues C, Gerard P, Rouveirol C (2010) Incremental learning of relational action models in noisy environments. In: Proceedings of the International Conference on Inductive Logic Programming (ILP)

Roncagliolo S, Tadepalli P (2004) Function approximation in hierarchical relational reinforcement learning. In: Proceedings of the Workshop on Relational Reinforcement Learning at ICML'04

Russell SJ, Norvig P (2003) Artificial Intelligence: a Modern Approach. Prentice Hall, New Jersey, 2nd edition

Ryan MRK (2002) Using abstract models of behaviors to automatically generate reinforcement learning hierarchies. In: Proceedings of the International Conference on Machine Learning (ICML), pp 522–529

Saad E (2008) A logical framework to reinforcement learning using hybrid probabilistic logic programs. In: Proceedings of the International Conference on Scalable Uncertainty Management (SUM), Lecture Notes in Artificial Intelligence, vol 5291, pp 341–355

Safaei J, Ghassem-Sani G (2007) Incremental learning of planning operators in stochastic domains. In: Proceedings of the International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM), pp 644–655

Sanner S (2005) Simultaneous learning of structure and value in relational reinforcement learning. In: Driessens K, Fern A, van Otterlo M (eds) Proceedings of the ICML-2005 Workshop on Rich Representations for Reinforcement Learning

Sanner S (2006) Online feature discovery in relational reinforcement learning. In: Proceedings of the ICML-06 Workshop on Open Problems in Statistical Relational Learning

Sanner S, Boutilier C (2005) Approximate linear programming for first-order MDPs. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)

Sanner S, Boutilier C (2006) Practical linear value-approximation techniques for first-order MDPs. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)

Sanner S, Boutilier C (2007) Approximate solution techniques for factored first-order mdps. In: Proceedings of the International Conference on Artificial Intelligence Planning Systems (ICAPS)

Sanner S, Kersting K (2010) Symbolic dynamic programming for first-order pomdps. In: Proceedings of the National Conference on Artificial Intelligence (AAAI)

Schmid U (2001) Inductive synthesis of functional programs: Learning domain-specific control rules and abstraction schemes. Habilitationsschrift, Fakultät IV, Elektrotechnik und Informatik, Technische Universität Berlin, Germany

Schuurmans D, Patrascu R (2001) Direct value approximation for factored MDPs. In: Proceedings of the Neural Information Processing Conference (NIPS)

Shapiro D, Langley P (2002) Separating skills from preference. In: Proceedings of the International Conference on Machine Learning (ICML), pp 570–577

Simpkins C, Bhat S, Isbell CL, Mateas M (2008) Adaptive Programming: Integrating Reinforcement Learning into a Programming Language. In: Proceedings of the Twenty-Third ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)

Slaney J, Thiébaux S (2001) Blocks world revisited. Artificial Intelligence 125:119–153

Song ZW, Chen XP (2007) States evolution in $\Theta(\lambda)$-learning based on logical mdps with negation. In: IEEE International Conference on Systems, Man and Cybernetics, pp 1624–1629

Song ZW, Chen XP (2008) Agent learning in relational domains based on logical mdps with negation. Journal of Computers 3(9):29–38

Stone P (2007) Learning and multiagent reasoning for autonomous agents. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Computers and Thought Award Paper

Stracuzzi DJ, Asgharbeygi N (2006) Transfer of knowledge structures with relational temporal difference learning. In: Proceedings of the ICML'06 Workshop on Structural Knowledge Transfer for Machine Learning

Sutton RS, Barto AG (1998) Reinforcement Learning: an Introduction. The MIT Press, Cambridge, Massachusetts

Sutton RS, McAllester DA, Singh S, Mansour Y (2000) Policy gradient methods for reinforcement learning with function approximation. In: Proceedings of the Neural Information Processing Conference (NIPS), pp 1057–1063

Thielscher M (1998) Introduction to the Fluent Calculus. Electronic Transactions on Artificial Intelligence 2(3–4):179–192

Thon I, Guttman B, van Otterlo M, Landwehr N, De Raedt L (2009) From non-deterministic to probabilistic planning with the help of statistical relational learning. In: Workshop on Planning and Learning at ICAPS

Torrey L (2009) Relational transfer in reinforcement learning. PhD thesis, University of Wisconsin-Madison, Computer Science Department

Torrey L, Shavlik J, Walker T, Maclin R (2007) Relational macros for transfer in reinforcement learning. In: Proceedings of the International Conference on Inductive Logic Programming (ILP)

Torrey L, Shavlik J, Natarajan S, Kuppili P, Walker T (2008) Transfer in reinforcement learning via markov logic networks. In: Proceedings of the AAAI-2008 Workshop on Transfer Learning for Complex Tasks

Toussaint M (2009) Probabilistic inference as a model of planned behavior. Knstliche Intelligenz (German Artificial Intelligence Journal) 3

Toussaint M, Plath N, Lang T, Jetchev N (2010) Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference. In: IEEE International Conference on Robotics and Automation (ICRA)

Van den Broeck G, Thon I, van Otterlo M, De Raedt L (2010) DTProbLog: A decision-theoretic probabilistic prolog. In: Proceedings of the National Conference on Artificial Intelligence (AAAI)

van Otterlo M (2003) Efficient reinforcement learning using relational aggregation. In: Proceedings of the Sixth European Workshop on Reinforcement Learning, Nancy, France (EWRL-6)

van Otterlo M (2004) Reinforcement learning for relational MDPs. In: Nowé A, Lenaerts T, Steenhaut K (eds) Machine Learning Conference of Belgium and the Netherlands (BeNeLearn'04), pp 138–145

van Otterlo M (2009a) Intensional dynamic programming: A rosetta stone for structured dynamic programming. Journal of Algorithms 64:169–191

van Otterlo M (2009b) The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for Adaptive Sequential Decision Making under Uncertainty in First-Order and Relational Domains. IOS Press, Amsterdam, The Netherlands

van Otterlo M, De Vuyst T (2009) Evolving and transferring probabilistic policies for relational reinforcement learning. In: Proceedings of the Belgium-Netherlands Artificial Intelligence Conference (BNAIC), pp 201–208

van Otterlo M, Wiering MA, Dastani M, Meyer JJ (2007) A characterization of sapient agents. In: Mayorga RV, Perlovsky LI (eds) Toward Computational Sapience: Principles and Systems, Springer, chap 9

Vargas B, Morales E (2008) Solving navigation tasks with learned teleo-reactive programs. pp 4185 –4185, DOI 10.1109/IROS.2008.4651240

Vargas-Govea B, Morales E (2009) Learning relational grammars from sequences of actions. In: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, Lecture Notes in Computer Science, vol 5856, pp 892–900

Vere SA (1977) Induction of relational productions in the presence of background information. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp 349–355

Walker T, Shavlik J, Maclin R (2004) Relational reinforcement learning via sampling the space of first-order conjunctive features. In: Proceedings of the Workshop on Relational Reinforcement Learning at ICML'04

Walker T, Torrey L, Shavlik J, Maclin R (2007) Building relational world models for reinforcement learning. In: Proceedings of the International Conference on Inductive Logic Programming (ILP)

Walsh TJ (2010) Efficient learning of relational models for sequential decision making. PhD thesis, Rutgers University, Computer Science Department

Walsh TJ, Littman ML (2008) Efficient learning of action schemas and web-service descriptions. In: Proceedings of the National Conference on Artificial Intelligence (AAAI)

Walsh TJ, Li L, Littman ML (2006) Transferring state abstractions between mdps. In: ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning

Wang C (2007) First-order markov decision processes. PhD thesis, Department of Computer Science, Tufts University, U.S.A.

Wang C, Khardon R (2007) Policy iteration for relational mdps. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)

Wang C, Khardon R (2010) Relational partially observable mdps. In: Proceedings of the National Conference on Artificial Intelligence (AAAI)

Wang C, Schmolze J (2005) Planning with pomdps using a compact, logic-based representation. In: Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI)

Wang C, Joshi S, Khardon R (2007) First order decision diagrams for relational MDPs. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)

Wang C, Joshi S, Khardon R (2008a) First order decision diagrams for relational MDPs. Journal of Artificial Intelligence Research (JAIR) 31:431–472

Wang W, Gao Y, Chen X, Ge S (2008b) Reinforcement learning with markov logic networks. In: Proceedings of the Mexican Conference on Artificial Intelligence, Lecture Notes in Computer Science, vol 5317, pp 230–242

Wang X (1995) Learning by observation and practice: An incremental approach for planning operator acquisition. In: Proceedings of the International Conference on Machine Learning (ICML), pp 549–557

Wingate D, Soni V, Wolfe B, Singh S (2007) Relational knowledge with predictive state representations. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)

Wooldridge M (2002) An introduction to MultiAgent Systems. John Wiley & Sons Ltd., West Sussex, England

Wu JH, Givan R (2007) Discovering relational domain features for probabilistic planning. In: Proceedings of the International Conference on Artificial Intelligence Planning Systems (ICAPS)

Wu K, Yang Q, Jiang Y (2005) ARMS: Action-relation modelling system for learning action models. In: Proceedings of the National Conference on Artificial Intelligence (AAAI)

Xu JZ, Laird JE (2010) Instance-based online learning of deterministic relational action models. In: Proceedings of the International Conference on Machine Learning (ICML)

Yoon SW, Fern A, Givan R (2002) Inductive policy selection for first-order MDPs. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)

Zettlemoyer LS, Pasula HM, Kaelbling LP (2005) Learning planning rules in noisy stochastic worlds. In: Proceedings of the National Conference on Artificial Intelligence (AAAI)

Zhao H, Doshi P (2007) Haley: A hierarchical framework for logical composition of web services. In: Proceedings of the International Conference on Web Services (ICWS), pp 312–319

Zhuo H, Li L, Bian R, Wan H (2007) Requirement specification based on action model learning. In: Proceedings of the International Conference on Intelligent Computing (ICIC), Springer, Lecture Notes in Computer Science, vol 4681, pp 565–574