A Survey of Knowledge-based Sequential Decision Making under Uncertainty

Shiqi Zhang[1] , **Mohan Sridharan**[2]

[1]SUNY Binghamton, USA
[2]University of Birmingham, UK
`zhangs@binghamton.edu; m.sridharan@bham.ac.uk`

Abstract

Reasoning with declarative knowledge (RDK) and sequential decision-making (SDM) are two key research areas in artificial intelligence. RDK methods reason with declarative domain knowledge, including commonsense knowledge, that is either provided a priori or acquired over time, while SDM methods (probabilistic planning and reinforcement learning) seek to compute action policies that maximize the expected cumulative utility over a time horizon; both classes of methods reason in the presence of uncertainty. Despite the rich literature in these two areas, researchers have not fully explored their complementary strengths. In this paper, we survey algorithms that leverage RDK methods while making sequential decisions under uncertainty. We discuss significant developments, open problems, and directions for future work.

1 Introduction

Agents operating in complex domains often have to execute a sequence of actions to complete complex tasks. These domains are characterized by non-deterministic action outcomes and partial observability, with sensing, reasoning, and actuation associated with varying levels of uncertainty. For instance, state of the art manipulation and grasping algorithms still cannot guarantee that a robot will grasp a desired object (say a coffee mug). In this paper, we use sequential decision making (**SDM**) to refer to algorithms that enable agents in such domains to compute action policies that map the current state (or the agent's estimate of it) to an action. More specifically, we consider SDM methods that model uncertainty probabilistically, i.e., **probabilistic planning** and **reinforcement learning** methods that enable the agents to choose actions toward maximizing long-term utilities.

SDM methods, by themselves, find it difficult to make best use of *commonsense* knowledge that is often available in any given domain. This knowledge includes *default* statements that hold in all but a few exceptional circumstances, e.g., "books are usually in the library but cookbooks are in the kitchen", but may not necessarily be natural or easy to represent quantitatively (e.g., probabilistically). It also includes information about domain objects and their attributes, agent

attributes and actions, and rules governing domain dynamics. In this paper, we use **declarative knowledge** to refer to such knowledge represented as relational statements. Many methods have been developed for reasoning with declarative knowledge (**RDK**), often using logics. These methods, by themselves, do not support probabilistic models of uncertainty toward achieving long-term goals, whereas a lot of information available to agents in dynamic domains is represented quantitatively to model the associated uncertainty.

For many decades, the development of RDK and SDM methods occurred in different communities that did not have a close interaction with each other. Sophisticated algorithms have been developed, more so in the last couple of decades, to combine the principles of RDK and SDM. However, even these developments have occurred in different communities, e.g., statistical relational AI, logic programming, reinforcement learning, and robotics. Also, these algorithms have not always considered the needs of agents in dynamic domains, e.g., reliability and computational efficiency while reasoning with incomplete knowledge. As a result, the complementary strengths of RDK and SDM methods have not been fully exploited. This paper aims to provide a survey of existing algorithms that facilitate SDM using RDK methods, answering the following question:

> *How best to reason with declarative knowledge for sequential decision making under uncertainty?*

Many algorithms have been developed to answer this question; we limit our attention in this paper to algorithms developed for an agent making sequential decisions under uncertainty in dynamic domains. To explain and illustrate some of the key concepts, we will draw on our own expertise in developing such methods for robotics domains. Figure 1 provides an overview of the theme of this survey[1].

2 Background

We begin by briefly introducing key concepts related to the RDK and SDM methods that we consider in this paper.

---

[1]This survey is based on a tutorial, titled "*Knowledge-based Sequential Decision-Making under Uncertainty*", presented by the authors at the AAAI Conference in 2019.
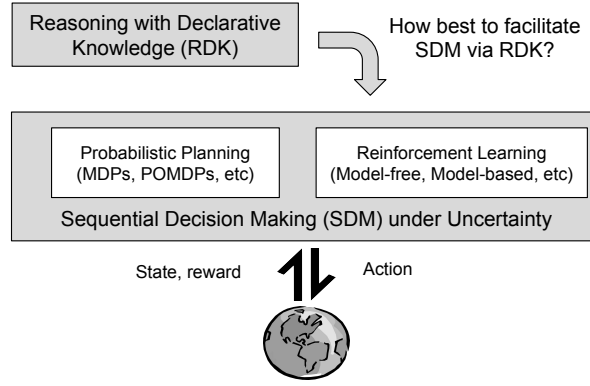
Figure 1: An overview of this survey: *reasoning with declarative knowledge* (RDK) for *sequential decision making* (SDM).

2.1 Reasoning with Declarative Knowledge

We consider a representation of commonsense knowledge in the form of statements describing relations between domain objects, domain attributes, actions, and axioms (i.e., rules). Historically, declarative paradigms based on logics have been used to represent and reason with such knowledge. This knowledge can also be represented quantitatively, e.g., using probabilities, but this is not always meaningful, especially in the context of statements of default knowledge such as "people typically drink a hot beverage in the morning" and "office doors usually closed over weekends". In this survey, *any mention of RDK refers to the use of logics for representing and using such domain knowledge for inference, planning, and diagnostics*. Planning and diagnostics in the context of RDK refer to *classical planning*, i.e., computing a sequence of actions to achieve any given goal, monitoring the execution of actions, and replanning if needed. This is different from probabilistic planning that computes and uses policies to choose actions in any given state or belief state (Section 2.2).

Prolog was one of the first logic programming languages [Colmerauer and Roussel, 1996], encoding domain knowledge using "rules" in terms of relations and axioms. Inferences are drawn by running a *query* over the relations. An axiom in Prolog is of the form:

```
Head :- Body
```

and is read as "Head is true if Body is true". For instance, the following rule states that all birds can fly.

```
fly(B) :- bird(B)
```

Rules with empty bodies are called *facts*. For instance, we can use "`bird(tweety)`" to state that tweety is a bird. Reasoning with this fact and the rule given above, we can infer that "`fly(tweety)`", i.e., tweety can fly. Research on RDK dates back to the 1950's, and has produced many knowledge representation and reasoning paradigms, such as First Order Logic, Lambda Calculus [Barendregt and others, 1984], Web Ontology Language [McGuinness *et al.*, 2004], and LISP [McCarthy, 1978].

Incomplete Knowledge   In most practical domains, it is infeasible to provide *comprehensive* domain knowledge. As a consequence, reasoning with the incomplete knowledge can result in incorrect or sub-optimal outcomes. Many logics

have been developed for reasoning with incomplete declarative knowledge. One representative example is Answer set programming (ASP), a declarative paradigm [Gebser *et al.*, 2012; Gelfond and Kahl, 2014]. ASP supports *default negation* and *epistemic disjunction* to provide non-monotonic logical reasoning, i.e., unlike classical logic, it allows an agent to revise previously held conclusions. An ASP program consists of a set of rules of the form:

```
a :- b, ..., c, not d, ..., not e.
```

where `a...e` are called literals, and `not` represents default negation, i.e., `not d` implies that d is not believed to be true, which is different from saying that d is false. Each literal can thus be true, false or unknown, and an agent associated with a program comprising such rules only believes things that it is forced to believe.

Action Knowledge   Action languages are formal models of part of natural language used for describing transition diagrams, and many action languages have been developed and used in robotics and AI. This includes STRIPS [Fikes and Nilsson, 1971], PDDL [McDermott *et al.*, 1998], and those with a distributed representation such as $\mathcal{AL}_d$ [Gelfond and Inclezan, 2013]. The following shows an example of using STRIPS to model an action `stack` whose preconditions require that the robot be holding object `X` and that object `Y` be clear. After executing this action, object `Y` is no longer clear and the robot is no longer holding `X`.

```
operator(stack(X,Y),
        Precond [holding(X),clear(Y)],
        Add [on(X,Y),clear(X)],
        Delete [holding(X),clear(Y)])
```

Given a goal, e.g., `on(`b_1`, `b_2`)`, which requires block b_1 to be on b_2, the action language description, along with a description of the initial/current state, can be used for planning a sequence of actions that achieve this goal. Action languages and corresponding systems have been widely used for classical planning [Ghallab *et al.*, 2004], aiming at computing action sequences toward accomplishing complex tasks that require more than one action.

Hybrid Representations   Logic-based knowledge representation paradigms typically support Prolog-style statements that are either true or false. By themselves, they do not

support reasoning about quantitative measures of uncertainty, which is often necessary for the interactions with SDM paradigms. As a result, many RDK-for-SDM methods utilize hybrid knowledge representation paradigms that jointly support both logic-based and probabilistic representations of knowledge; they do so by associating probabilities with specific facts and/or rules. Over the years, many such paradigms have been developed; these include Markov Logic Network (MLN) [Richardson and Domingos, 2006], Bayesian Logic [Milch *et al.*, 2006], probabilistic first-order logic [Halpern, 2003], PRISM [Gorlin *et al.*, 2012], independent choice logic [Poole, 2000], ProbLog [Fierens *et al.*, 2015; Raedt and Kimmig, 2015], KBANN [Towell and Shavlik, 1994], and P-log, an extension of ASP [Baral *et al.*, 2009]. We will discuss some of these later in this paper.

2.2 Sequential Decision Making

We consider two classes of SDM methods: probabilistic planning (**PP**) [Puterman, 2014] and reinforcement learning (**RL**) [Sutton and Barto, 2018], depending on the availability of world models. A common assumption in these methods is the first-order Markov property, i.e., the next state is assumed to be conditionally independent of all previous states given the current state. Also, actions are assumed to be non-deterministic, i.e., they do not always provide the expected outcomes, and the state is assumed to be fully or partially observable. Unlike classical planning (see Section 2.1), these methods compute and use a *policy* that maps each possible (belief) state to an action to be executed in that (belief) state.

Probabilistic Planning If the state is fully observable, PP problems are often formulated as a Markov decision process (**MDP**) described by a four-tuple $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$ whose elements define the set of states, set of actions, the probabilistic state transition function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$, and the reward specification $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S}' \to \Re$. Each state can be specified by assigning values to a (sub)set of domain attributes. The MDP is solved to maximize the expected cumulative reward over a time horizon, resulting in a *policy* $\pi : s \mapsto a$ that maps each state $s \in \mathcal{S}$ to an action $a \in \mathcal{A}$. Action execution corresponds to repeatedly invoking the policy and executing the corresponding action.

If the current world state is not fully observable, PP problems can be modeled as a partially observable MDP (**POMDP**) [Kaelbling *et al.*, 1998] that is described by a six-tuple $\langle \mathcal{S}, \mathcal{A}, Z, T, O, R \rangle$, where Z is a set of observations, and $O : \mathcal{S} \times \mathcal{A} \times Z \to [0,1]$ is the observation function; other elements are defined as in the case of MDPs. The agent maintains a *belief state*, a probability distribution over the underlying states. It repeatedly executes actions, obtains observations, and revises the belief state through Bayesian updates:

$$b'(s') = \frac{O(s', a, o) \sum_{s \in \mathcal{S}} T(s, a, s') b(s)}{pr(o|a, b)}$$

where b, s, a, and o represent belief state, state, action, and observation respectively; and $pr(o|a, b)$ is a normalizer. The POMDP is also solved to maximize the expected cumulative reward over a time horizon; in this case, the output is a *policy* $\pi : b \mapsto a$ that maps beliefs to actions.

Reinforcement Learning Agents frequently have to make sequential decisions with an incomplete model of domain dynamics (e.g., without R, T, or both), making it infeasible to use classical PP methods. Under such circumstances, RL algorithms can be used by the agent to explore the effects of executing different actions, learning a policy (mapping states to actions) that maximizes the expected cumulative reward as the agent tries to achieve a goal [Sutton and Barto, 2018]. The underlying formulation is that of an MDP or a formulation that reduces to an MDP under certain constraints.

There are at least two broad classes of RL methods: **model-based** and **model-free**. Model-based RL methods enable an agent to learn a model of the domain, e.g., $R(s, a)$ and $T(s, a, s')$ in an MDP, from the experiences obtained by the agent by trying out different actions in different states. Once a model of the domain is learned, the agent can use PP methods to compute an action policy. Model-free RL methods, on the other hand, do not learn an explicit model of the domain; the policy is instead directly computed from the experiences gathered by the agent. The standard approach to incrementally update the value of each state is the Bellman equation:

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} pr(s', r|s, a)[r + \gamma v_k(s')], \forall s \in \mathcal{S}$$

where $v(s)$ is the value of state s, and γ is a discount factor. It is also possible to compute the values of state-action pairs, i.e., $Q(s, a)$, from which a policy can be computed.

Many algorithms have been developed for model-based and model-free RL; for more details, please see [Sutton and Barto, 2018]. More recent work has also explored the integration of deep neural networks (DNNs) with RL, e.g., to approximate the value function [Mnih *et al.*, 2015], and deep policy-based methods, e.g., [Schulman *et al.*, 2015; Schulman *et al.*, 2017]. This survey focuses on the interplay between SDM (including RL) and RDK methods; the properties of individual RL methods (or SDM methods) are out of scope.

3 Characteristic Factors

Before we discuss methods that leverage RDK for SDM, we describe the factors that we use to characterize these methods. The first two factors are related to the representation of knowledge and uncertainty, and the next three factors are related to reasoning with this knowledge and the underlying assumptions about domain dynamics and observability. The final three factors are related to the acquisition of domain knowledge. Unless stated otherwise, the factors are orthogonal to each other; the choice of a particular value for one factor can (for the most part) be made independent of the choice of value for other factors.

3.1 Representational Factors

We introduce two characteristic factors related to the representation of knowledge and uncertainty in RDK-for-SDM methods. The first factor is based on the relationships between the different descriptions of knowledge and uncertainty considered in these methods, and the second factor is based on the abstractions considered in this representation.
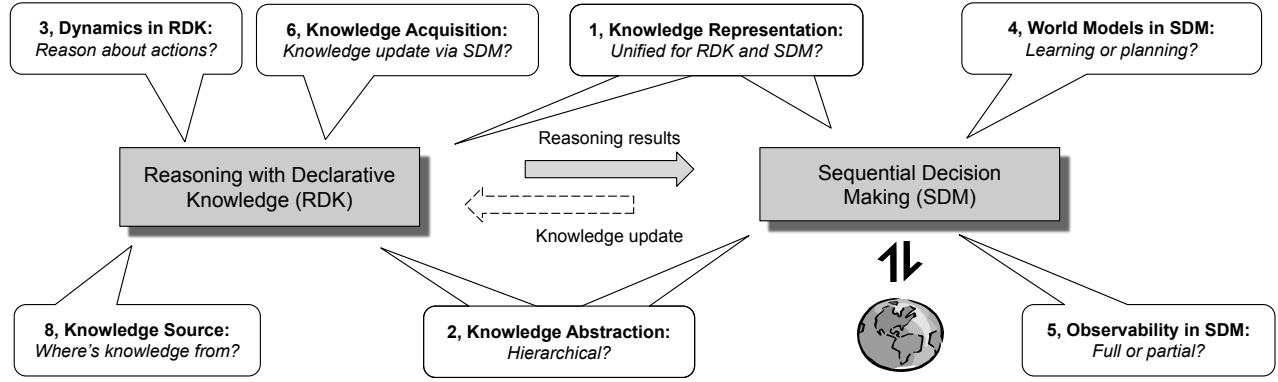
Figure 2: Characteristic factors in the development of RDK-for-SDM methods. The individual factors are discussed in details in Section 3, and are also used for the discussions of representative algorithms in Section 4.

Factor 1: Representation of Descriptions The *first factor* categorizes the methods that leverage RDK for SDM into two broad classes based on how they represent logical and probabilistic descriptions of knowledge and uncertainty.

Methods in the first group use a **unified representation** that is expressive enough to serve as the shared representation paradigm of both RDK and SDM. For instance, one can use a joint probabilistic-logical representation of knowledge and the associated uncertainty with probabilistic relational statements to describe both the beliefs of RDK, and the rules of SDM governing domain dynamics. These approaches provide significant expressive power, but manipulating such a representation (for reasoning or learning) imposes a significant computational burden.

Methods in the second group use a **linked representation** to model the components of RDK and SDM. For instance, information closer to the sensorimotor level can be represented quantitatively to model the uncertainty in beliefs, and logics can be used for representing and reasoning with a more high-level representation of commonsense domain knowledge and uncertainty. These methods trade expressivity, correctness guarantees, or both for computational speed. For instance, to save computation, sometimes probabilistic statements with residual uncertainty are committed as true statements in the logical representation, potentially leading to incorrect inferences. Methods in this group can vary substantially based on if/how information and control are transferred between the different representations. For instance, many methods based on a linked representation switch between a logical representation and a probabilistic representation depending on the task to be performed; other methods address the challenging problem of establishing links between the corresponding transition diagrams to provide a *tighter coupling*.

Factor 2: Knowledge Abstraction RDK-for-SDM methods often reason about knowledge at different granularities. Consider a mobile delivery robot. It can reason about rooms and cups to compute a high-level plan for preparing and delivering a beverage, and reason about geometric locations at finer granularities to grasp and manipulate a given cup.

The *second factor* used to characterize these methods is the use of different **abstractions** within each component or in different components, e.g., a hierarchy of state-action spaces for SDM, or a combination of an abstract representation for logic-based task planning and a fine-resolution metric representation for probabilistic motion planning. The use of different abstractions makes it difficult to identify and use all the relevant knowledge to make decisions; note that this challenge is present in the "linked representation" methods discussed in the context of Factor 1.

Note that methods that explore different abstractions of knowledge often encode rich domain knowledge (including cognitive models) and perform RDK at an abstract level, using SDM for selecting and executing more primitive (but more precise) actions at a finer granularity. Despite the variety of knowledge representation paradigms, we still maintain the constraint that we only consider algorithms with knowledge being represented declaratively.

3.2 Reasoning Factors

Given a representation of knowledge, an agent needs to reason with this knowledge to achieve desired goals. Here, we are particularly interested in if and how knowledge of domain dynamics is used in the RDK and SDM components, and the effect of state representation on the choice of methods. These issues are captured by the following three factors.

Factor 3: Dynamics in RDK RDK algorithms manipulate the underlying representations for different classes of tasks; in this paper, we consider inference, classical planning, and diagnostics. Among these tasks, inference requires the agent to draw conclusions based on its current beliefs and domain knowledge, while planning and diagnostics require RDK algorithms to reason about changes caused by actions executed over a period of time.

We introduce the *third factor* to categorize the RDK-for-SDM methods into two groups based on whether the RDK component reasons about **actions and change**. Methods that perform inference based on a particular snapshot of the world are in one category, whereas methods for classical planning and diagnostics that requiring decisions to be made over a sequence of time steps are in the other category. These choices may be influenced by the specific application or use of SDM methods, e.g, RDK-for-SDM methods that leverage domain

knowledge to improve the exploration behaviors of RL agents require the ability to reason about actions and change.

Factor 4: World Models in SDM The *fourth factor* categorizes SDM methods, i.e., PP and RL methods, depending on the availability of world models. When world models are available, we can construct Dynamic Bayesian Networks (DBNs), and compute action policies. When world models are not available, the agent can interact with its environment and learn action policies through trial and error, and this can be formulated as an RL problem. Among the RL methods, model-based methods explicitly learn the domain dynamics (e.g., T and R) and use PP methods to compute the action policy. Model-free RL methods, on the other hand, enable agents to directly use their experiences of executing different actions to compute the policies mapping states to actions.

Factor 5: State or Belief State in SDM SDM methods involve an agent making decisions based on observing and estimating the state of the world. A key distinction here is whether this state is fully observable or partially observable, or equivalently, whether the observations are assumed to be complete and correct. The *fifth factor* categorizes the SDM methods based on whether they reason assuming full knowledge of state after action execution, or assume that the true state is unknown and reason with *belief states*, i.e., probability distributions over the underlying states. Among the SDM formulations considered in this paper, MDPs map to the former category, whereas POMDPs map to the latter category.

Note that there are other distinguishing characteristics of reasoning systems (of RDK-for-SDM systems) that we do not explore in this paper. For instance, reasoning in such systems often includes a combination of active and reactive processes, e.g., actively planning and executing a sequence of actions to achieve a particular goal in the RDK component, and computing a probabilistic policy that is then used reactively for action selection in the SDM component.

3.3 Knowledge Acquisition Factors

Since comprehensive domain knowledge is often not available, RDK-for-SDM methods may include an approach for knowledge acquisition and revision. We introduce three characteristic factors related to how knowledge is acquired and the source of this knowledge.

Factor 6: Online vs. Offline Acquisition Our *sixth factor* categorizes methods based on whether they acquire knowledge **online** or **offline**. Methods in the first category interleave knowledge acquisition and task completion, with the agent revising its knowledge while performing the task based on the corresponding observations. In comparison, methods in the second category decouple knowledge acquisition and task completion, with the agent extracting knowledge from a batch of observations in a separate phase; this phase occurs either before or after the assigned task is completed. For the purposes of this survey, RDK-for-SDM methods that do not support knowledge acquisition are grouped in the "offline" category for this characteristic factor.

Factor 7: Active vs. Reactive Acquisition RDK-for-SDM methods are categorized by our *seventh factor* based on

whether they explicitly execute actions for acquiring knowledge. Some methods use an **active acquisition** approach, which has the agent explicitly plan and execute actions with the objective of acquiring previously unknown knowledge and revising existing knowledge. These actions take the form of exploring the outcomes of actions and extracting information from the observations, or soliciting input from humans. Active acquisition is often coupled with active or reactive reasoning, e.g., for computing exploration plans. Other RDK-for-SDM methods use a **reactive acquisition** approach in which knowledge acquisition is a secondary outcome. As the agent is executing actions to perform the target task(s), it ends up acquiring knowledge from the corresponding observations; this may, in turn, trigger active acquisition.

Factor 8: Knowledge Source The *eighth factor* categorizes RDK-for-SDM methods based on the source of the declarative domain knowledge. In some methods, this knowledge is obtained by direct **human encoding**, e.g., in the form of logical statements written by humans to represent facts and axioms. This is a common source of domain knowledge, especially that which is encoded initially. The knowledge can also be acquired through **agent interaction**, e.g., agents can directly perceive their working environments through cameras and extract information using computer vision methods to populate the knowledge base. Note that some methods use a combination of sources, e.g., agents extract information from some Web sources provided by humans, or agents solicit information through dialog with humans. Knowledge directly encoded by domain experts is often more reliable but it may require considerable time and effort from these experts. In comparison, the human effort required to enable agents to acquire knowledge is typically much less, but this knowledge is often less reliable.

3.4 Summary of Characteristic Factors

Methods that use RDK for SDM can be mapped to the space whose axes are the factors described above; these factors are also summarized in Figure 2. Some methods can include combinations of the factors related to representation, reasoning, and/or knowledge acquisition. For instance, a given system could support active online knowledge acquisition while reasoning with domain dynamics and belief states, which another system could support interactive knowledge acquisition from humans while reasoning about actions and change based on a linked representation. Methods that couple representation, reasoning, and learning provide key benefits, e.g., reasoning can be used to trigger, inform, and guide efficient knowledge acquisition. However, they also present some challenges, e.g., in suitably reconciling differences between existing knowledge and learned knowledge. These advantages and challenges are discussed below in the context of representative RDK-for-SDM methods.

4 RDK-for-SDM Methods

In this section, we review some representative RDK-for-SDM systems by grouping them based on their primary contributions. First, Section 4.1 discusses some systems that primarily

focus on the knowledge representation challenges in RDK-for-SDM. Sections 4.2- 4.3 then describe RDK-for-SDM systems in which the key focus is on the underlying reasoning and knowledge acquisition challenges respectively. Note that this grouping is based on *our* understanding of the key contributions of each system; many of these systems include contributions across the three groups as summarized in Table 1.

4.1 Representation-focused Systems

As stated in Section 2.1, many generic hybrid representations have been developed to support both logic-based and probabilistic reasoning with knowledge and uncertainty.

Statistical relational AI Some of the foundational work in this area has built on work in statistical relational learning/AI. These RDK-for-SDM methods typically use unified representations and differ based on the underlying design choices. For instance, Markov Logic Networks (MLNs) combine probabilistic graphical models and first order logic, assigning weights to logic formulas [Richardson and Domingos, 2006]; these have been extended to Markov logic decision networks by associating logic formulas with utilities in addition to weights [Nath and Domingos, 2009]. In a similar manner, Probabilistic Logic (ProbLog) programming annotates facts in logic programs with probabilities and supports efficient inference and learning using weighted Boolean formulas [Raedt and Kimmig, 2015]. This includes an extension of the basic ProbLog system, called Decision-Theoretic (DT)ProbLog, in which the utility of a particular choice of actions is defined as the expected reward for its execution in the presence of probabilistic effects [den Broeck *et al.*, 2010]. Another example of an elegant (unified) formalism for dealing with degrees of belief and their evolution in the presence of noisy sensing and acting, extends situation calculus by assigning weights to possible worlds and embedding a theory of action and sensing [Bacchus *et al.*, 1999]. This formalism has been extended to deal with decision making in the continuous domains seen in many robotics applications [Belle and Levesque, 2018]. Others have developed frameworks based on unified representations specifically for decision theoretic reasoning, e.g., first-order relational POMDPs that leverage symbolic programming for the specification of POMDPs with first-order abstractions [Juba, 2016; Sanner and Kersting, 2010].

Classical planning RDK-for-SDM systems based on unified representations have also built on tools and methods in classical planning. Examples include PPDDL, a probabilistic extension of the action language PDDL, which retains the capabilities of PDDL and provides a semantics for planning problems as MDPs [Younes and Littman, 2004], and Relational Dynamic Influence Diagram Language (RDDL) that was developed to formulate factored MDPs and POMDPs [Sanner, 2010]. In comparison with PPDDL, RDDL provides better support for modeling concurrent actions and for representing rewards and uncertainty quantitatively.

Logic programming RDK-for-SDM systems with a unified representation have also been built based on logic programming frameworks. One example is P-log, a probabilistic extension of ASP that encodes probabilistic facts and rules to compute probabilities of different possible worlds represented as answer sets [Baral *et al.*, 2009]. P-log has been used to specify MDPs for SDM tasks, e.g., for robot grasping [Zhu, 2012]. More recent work has introduced a coherence condition that facilitates the construction of P-log programs and proofs of correctness [Balai *et al.*, 2019]. One limitation of P-log, from the SDM perspective, is that it requires the horizon to be provided as part of the input. The use of P-log for probabilistic planning with infinite horizons requires a significant engineering effort.

As stated earlier in the context of Factor 1 in Section 3.1, RDK-for-SDM systems with linked (hybrid) representations trade expressivity or correctness guarantees for computational speed, an important consideration if an agent has to respond to dynamic changes in complex domains. These methods often also use different levels of abstraction and link rather than unify the corresponding descriptions of knowledge and uncertainty. This raises interesting questions about the choice of domain variables in each representation, and the transfer of knowledge and control between the different reasoning mechanisms. For instance, a robot delivering objects in an office building may plan at an abstract level, reasoning logically with rich commonsense domain knowledge (e.g., about rooms, objects, and exogenous agents) and cognitive theories. The abstract actions can be implemented by reasoning probabilistically at a finer resolution about relevant domain variables (e.g., regions in specific rooms, parts of objects, and agent actions).

Switching systems The simplest option for methods based on linked representations is to switch between reasoning mechanisms based on different representations for different tasks. One example is the *switching planner* that uses either a classical first-order logic planner or a probabilistic (decision-theoretic) planner for action selection [Göbelbecker *et al.*, 2011]. This method used a combination of the Fast-Downward [Helmert, 2006] and PPDDL [Younes and Littman, 2004] representations. Another approach uses ASP for planning and diagnostics at a coarser level of abstraction, switches to using probabilistic algorithms for executing each abstract action, and adds statements to the ASP program's history to denote success or failure of action execution; this approach has been used for multiple robots in scenarios that mimic manufacturing in toy factories [Saribatur *et al.*, 2019].

Tightly-Coupled Systems There has been some work on generic RDK-for-SDM frameworks that represent and reason with knowledge and beliefs at different abstractions, and "*tightly couple*" the different representations and reasoning mechanisms by formally establishing the links between and the attributes of the different representations. These methods are often based on the *principle of refinement*. This principle has also been explored in fields such as software engineering and programming languages, but without any theories of actions and change that are important in robotics and AI. One recent approach examined the refinement of agent action theories represented using situation calculus at two different levels. This approach makes a strong assumption of the existence of a bisimulation relation between the action theories for a given refinement mapping between these theories at the high-level and the low-level [Banihashemi *et*

al., 2018]. Our recent work on a refinement-based architecture (REBA) in robotics considers transition diagrams of any given domain at two different resolutions, with the fine-resolution diagrams defined formally as a refinement of the coarse-resolution diagram [Sridharan *et al.*, 2019]. Non-monotonic logical reasoning at the coarse-resolution with incomplete commonsense domain knowledge provides a sequence of abstract actions to achieve any given goal. Each abstract action is implemented as a sequence of concrete actions by automatically zooming to and reasoning probabilistically with automatically-constructed models (e.g., POMDPs) of the relevant part of the fine-resolution diagram, adding relevant observations and outcomes to the coarse-resolution history [Gomez *et al.*, 2020]. The formal definition of refinement, zooming, and the connections between the transition diagrams enables smooth transfer of relevant information and control, improving scalability to complex domains.

Cognitive architectures Systems such as ACT-R [Anderson and Lebiere, 2014], SOAR [Laird, 2012], ICARUS [Langley and Choi, 2006] and DIRAC [Scheutz *et al.*, 2007] can represent and draw inferences based on declarative knowledge, often using first-order logic. These architectures typically support SDM through a linked representation, but some architectures have pursued a unified representation for use in robotics by attaching a quantitative measure of uncertainty to logic statements [Sarathy and Scheutz, 2018].

There are many other RDK-for-SDM systems based on hybrid representations. In these systems, the focus is not on developing new representations; they instead adapt or combine existing representations to support interesting reasoning and learning capabilities, as described below.

4.2 Reasoning-focused Systems

Next, we discuss some other representative RDK-for-SDM systems in which the primary focus is on addressing related reasoning challenges.

RDK Guiding SDM Many RDK-for-SDM systems use RDK for planning a sequence of tasks to be completed, and implement each task by performing SDM based on the (encoded) relevant information to execute a sequence of primitive actions. This includes many of the systems described in the previous section and especially those developed for robotics domains. For instance, an extension of the switching planner uses a three-layered organization of knowledge (instance, default and diagnostic), with knowledge at the higher level modifying that at the lower levels, and reasons with first-order logic to guide probabilistic planning [Hanheide *et al.*, 2017]. Another example is the CORPP system that uses P-log to generate informative priors for POMDP planning [Zhang and Stone, 2015]. Other researchers have exploited factored state spaces to develop algorithms that use manually-encoded probabilistic declarative knowledge to efficiently compute informative priors for POMDPs [Chitnis *et al.*, 2018]. These methods separate the variables modeled at different levels and (manually) link relevant variables across the levels, improving scalability and dynamic response. All these systems reason about actions and change in the RDK component, and reason with state/belief states and world models for SDM.

They link the flow of information between the different reasoning mechanisms, often at different abstractions, but they typically do not focus on developing (or extending) the underlying representations or on formally establishing properties of the connections between the representations.

Dynamics Models for SDM In some RDK-for-SDM systems, the focus is on RDK guiding the construction or adaptation of the world models used for SDM. One example is the extension of [Chitnis *et al.*, 2018] that seeks to automatically determine the variables to be modeled in the different representations [Chitnis and Lozano-Pérez, 2020]. Another example is the use of logical smoothing to refine past beliefs in light of new observations; the refined beliefs can then be used for diagnostics and to reduce the state space for planning [Mombourquette *et al.*, 2017]. There is also recent work on an action language called pBC+, which supports the definition of MDPs and POMDPs over finite and infinite horizons [Wang *et al.*, 2019].

In some RDK-for-SDM systems, RDK and prior experiences of executing actions in the domain are used to construct domain models and guide SDM. For instance, symbolic planning has been combined with hierarchical RL to guide the agent's interactions with the world, resulting in reliable world models and SDM [Illanes *et al.*, 2020]. In other work, each symbolic transition is mapped (manually) to options, i.e., temporally-extended MDP actions; RDK helps compute the MDP models and policies, and the outcomes of executing the corresponding primitive actions help revise the values of state action combinations in the symbolic reasoner [Yang *et al.*, 2018]. These systems use a linked representation, and reason about dynamics in RDK and states and world models in SDM. Other systems reason without explicit world models in SDM, e.g., the use of deep RL methods to compute the policies in the options corresponding to each symbolic transition in the context of game domains [Lyu *et al.*, 2019].

Credit Assignment and Reward Shaping When MDPs or POMDPs are used for SDM in complex domains, rewards are sparse and typically obtained only on task completion, e.g., after executing a plan or at the end of a board game. As a special case of learning and using world models in SDM, researchers have leveraged RDK methods to model and shape the rewards to improve the agent's decision-making. For instance, declarative action knowledge has been used to compute action sequences, using the action sequences to compute a potential function and for reward shaping in game domains [Grounds and Kudenko, 2005; Grzes and Kudenko, 2008; Efthymiadis and Kudenko, 2013]. In this work, RL methods such as Q-learning, SARSA, and Dyna-Q were combined with a STRIPS planner, with the planner shaping the reward function used by the agents to compute the optimal policy. These systems perform RDK with domain dynamics, and reason about states but no explicit world models in SDM.

In some cases, the reward specification is obtained from statistics and/or contextual knowledge provided by humans. For example, the iCORPP algorithm enables a robot to reason with contextual knowledge using P-log to automatically determine the rewards (and transition functions) of a POMDP used for planning [Zhang *et al.*, 2017]. Another system,

Table 1: A subset of the surveyed RDK-for-SDM algorithms from the literature mapped to the space defined by the characteristic factors discussed in Section 3. Each column corresponds to one characteristic factor (except for the last one); if a factor's range includes multiple values, this table shows the most typical value. **Uni. Rep.:** unified representation for both RDK and SDM (Factor 1). **Abs. Rep.:** abstract representations for RDK and SDM that are linked together (Factor 2). **Dyn. RDK:** declarative knowledge includes action knowledge and can be used for task planning (Factor 3). **RL SDM:** world models are not provided to SDM, rendering RL necessary (Factor 4). **Par. Obs.:** current world states are partially observable (Factor 5). **On. Acq.:** online knowledge acquisition is enabled (Factor 6). **ML RDK:** at least part of the knowledge base is learned by the agents, where the opposite is human developing the entire knowledge base (Factor 7). **Rew. RDK:** RDK is used for reward shaping.

		Uni. Rep.	Abs. Rep.	Dyn. RDK	RL SDM	Par. Obs.	On. Acq.	ML RDK	Rew. RDK
Representation	[Younes and Littman, 2004]	●	○	/	○	○	○	○	○
	[Sanner, 2010]	●	○	/	○	◐	○	○	●
	[Baral et al., 2009]	●	○	/	○	◐	○	○	○
	[Wang et al., 2019]	●	○	/	○	◐	●	●	●
	[Zhang et al., 2017]	●	○	○	○	◐	●	○	●
Reasoning	[Sridharan et al., 2019]	○	●	●	○	●	●	●	○
	[Illanes et al., 2020]	○	●	●	●	○	○	○	○
	[Yang et al., 2018; Lyu et al., 2019]	○	●	●	●	○	●	○	○
	[Furelos-Blanco et al., 2020]	○	●	●	●	○	●	●	●
	[Göbelbecker et al., 2011]	○	○	●	○	●	○	○	○
	[Garnelo et al., 2016]	○	○	○	●	○	●	●	○
	[Chitnis et al., 2018]	○	○	○	○	●	●	○	○
	[Zhang et al., 2015; Zhang and Stone, 2015]	○	○	○	○	●	○	○	○
	[Amiri et al., 2020]	○	○	○	○	●	●	●	○
	[Grounds and Kudenko, 2005]	○	●	●	●	○	○	○	●
	[Hoelscher et al., 2018]	○	○	○	○	●	●	○	●
	[Camacho et al., 2019]	○	○	●	●	○	○	○	●
	[Zhang et al., 2019]	○	○	○	●	○	○	○	○
	[Leonetti et al., 2016]	○	○	●	●	○	●	○	○
	[Eysenbach et al., 2019]	○	●	●	●	○	●	●	○
Acquisition	[Konidaris et al., 2018; Gopalan et al., 2020]	○	●	●	○	○	●	●	○
	[Thomason et al., 2015; Amiri et al., 2019]	○	○	○	○	●	●	○	○
	[She and Chai, 2017]	○	○	●	●	○	●	●	○
	[Mericli et al., 2014]	○	○	●	○	○	●	○	○
	[Samadi et al., 2012]	○	○	○	○	○	●	●	○

called LPPGI, enables robots to leverage human expertise for POMDP-based planning under uncertainty in the context of task specification and execution [Hoelscher et al., 2018]. RDK in this system is rather limited; domain dynamics are not considered and the system is limited to maximizing the expected probability of satisfying logic objectives in the context fo a robot arm stacking boxes. There has also been work on using Linear Temporal Logic to represent and reason with declarative knowledge, especially temporal constraints implied by phrases such as "until" and "eventually", to automatically generate reward specification for RL in a simulated game domain [Camacho et al., 2019].

Guiding SDM-based Exploration When the main objective of SDM is exploration or discovery of particular aspects of the domain, RDK can be used to inform and guide the trade-off between exploration and exploitation, and to avoid poor-quality exploration behaviors in SDM. For instance, the DARLING algorithm uses RL to explore and compute action sequences that lead to long-term goals under uncertainty, with RDK being used to filter out unreasonable actions from exploration [Leonetti et al., 2016]; this approach has been evaluated on real robots navigating office environ-

ments to locate people of interest. Another similar approach uses RDK to guide an agent's exploration behavior (formulated as SDM) in non-stationary environments [Ferreira et al., 2017], and to learn constraints that prevent risky behaviors in video games [Zhang et al., 2019]. There is also work on non-monotonic logical reasoning with commonsense knowledge to automatically determine the state space for relational RL-based exploration of previously unknown action capabilities [Sridharan et al., 2017].

4.3 Knowledge Acquisition-focused Systems

Next, we discuss some RDK-for-SDM systems whose main contribution is the acquisition (and revision) of domain knowledge used for RDK. This knowledge can be obtained through manual encoding and/or automated acquisition from different sources (Web, corpora, sensor inputs).

Learning from Experience Many recent methods to acquire the declarative knowledge have been based on the recent developments in data-driven approaches (e.g., deep learning and RL). For instance, the symbols needed for task planning have been extracted from the replay buffers of multiple trials of deep RL, with similar states (in the replay buffers) being

grouped to form the search space for symbolic planning [Eysenbach *et al.*, 2019]. In robotics domains, a small number of real-world trials have been used to enable a robot to learn the symbolic representations of the preconditions and effects of a door-opening action [Konidaris *et al.*, 2018]. Knowledge acquisition in these systems is offline and reactive. Researchers have also enabled robots to simultaneously acquire latent space symbols and language groundings based on prior demonstration trajectories paired with natural language instructions [Gopalan *et al.*, 2020]; in this case, learning is predominantly offline and active, and requires significantly fewer training samples compared to end-to-end systems. Other research has demonstrated the use of relational RL guided by ASP-based non-monotonic logical reasoning and decision-tree induction to learn previously unknown actions and domain axioms that are subsequently used for RDK [Sridharan and Meadows, 2018]. In this system, knowledge acquisition is both online and offline, and active and reactive.

Learning from Human Input and Active Search Researchers have developed a dialog-based interactive approach for situated task specification, with the robot learning new actions and their preconditions through verbal instructions [Mericli *et al.*, 2014]. Other researchers have developed methods that acquire domain knowledge using data available on the Web [Samadi *et al.*, 2012]. Information (to be encoded in first-order logic) about the likely location of paper would, for instance, be found by analyzing the results of a web search for "kitchen" and "office". In another related approach, SDM has been used to manage human-robot dialog, which helps a robot acquire knowledge of synonyms (e.g., "java" and "coffee") that are used for RDK [Thomason *et al.*, 2015]. Building on this work, other researchers have developed methods to add new object entities to the declarative knowledge in RDK-for-SDM systems [Amiri *et al.*, 2019].

5 Challenges and Opportunities

Over the last few decades, researchers have made significant progress in developing sophisticated methods for reasoning with declarative knowledge and for sequential decision making under uncertainty. In recent years, improved understanding of the complementary strengths of the methods developed in these two areas has also led to the development of sophisticated methods that seek to integrate and exploit these strengths. These integrated systems have provided promising results, but they have also identified several open problems and opened up many directions for further research. Below, we discuss some of these problems and research directions:

Representational Choices: As discussed in Sections **??** and **??**, existing methods integrating RDK and SDM methods are typically based on one of two broad classes of representations. General-purpose methods often use a unified representation and associated reasoning methods for different descriptions of domain knowledge, e.g., a unified representation for logic-based and probabilistic descriptions of knowledge. On the other hand, integrated systems developed specifically for robotics and other dynamic domains link rather than unify the different representations, including those at different abstractions, trading correctness for computational efficiency.

A wide range of representations and reasoning methods are possible within each of these two classes; these need to be explored further to better understand the choice (of representation and reasoning methods) best suited to any particular application domain. During this exploration, it will be important to carefully study any trade-offs made in terms of the expressiveness of the representation, the ability to support different abstractions, the computational complexity of the reasoning methods, and the ability to establish that the behavior of the robot (or agent) equipped with the resulting system satisfies certain desirable properties.

Incremental and Interactive Learning: Irrespective of the representation and reasoning methods used for RDK, SDM, or a combination of the two, the knowledge encoded will be incomplete and/or cease to be relevant over a period of time in any practical, dynamic domain. In the age of "big data", certain domains provide ready availability of a lot of labeled data from which the previously unknown information can be learned, whereas such labeled training data is scarce in other domains; in either case, the knowledge acquired from the data may not be comprehensive. Also, it is computationally expensive to learn information from large amounts of data. Incremental and interactive learning thus continues to be an open problem in systems that integrate RDK and SDM. Promising results have been obtained by methods that promote efficient learning by using reasoning to trigger learning only when it is needed and limit (or guide) learning to those concepts that are relevant to the tasks at hand; such methods need to be developed and analyzed further. Another interesting research thrust is to learn *cumulatively* from the available data and merge the learned information with the existing knowledge such that reasoning continues to be efficient as additional knowledge is acquired over time.

Human "in the loop": Many methods for RDK, SDM, or RDK-for-SDM, assume that any prior knowledge about the domain and the associated tasks is provided by the human in the initial stages, or that humans are available during task execution for reliable feedback and supervision. These assumptions do not always hold true in practice. Research indicates that humans can be a rich source of information but there is often a non-trivial cost associated with acquiring and encoding such knowledge from people. Since it is challenging for humans to accurately specify or encode domain knowledge in complex domains, there is a need for methods that consider humans as collaborators to be consulted by a robot based on necessity and availability. Such methods will need to address key challenges related to the protocols for communication between a robot and a human, considering factors such as the expertise of the human participants and the availability of humans in social contexts. Another related problem that is increasingly getting a lot of attention is to enable a reasoning and learning system to *explain* its decisions and beliefs in human-understandable terms.

Combining Reasoning, Learning, and Control: As discussed in this paper, many methods than integrate RDK and SDM focus on decision making (or reasoning) tasks. There also some methods that include a learning component and some that focus on robot control and manipulation tasks.

However, robots that sense and interact with the real world often require a system that combines reasoning, learning, and control capabilities. Similar to the combination of reasoning and learning (as mentioned above), tightly coupling reasoning, learning, and control presents unique advantages and unique open problems in the context of integrated RDK and SDM. For instance, reasoning and learning can be used to identify (on demand) the relevant variables that need to be included in the control laws for the tasks at hand. At the same time, real world control tasks often require a very different representation of domain attributes, e.g., reasoning to move a manipulator arm may be performed in a discrete, coarser-granularity space of states and actions whereas the actual manipulation tasks being reasoned about need to be performed in a continuous, finer-granularity space. There is thus a need for systems that integrate RDK and SDM, and suitably combine reasoning, learning, and control by carefully exploring the effect of different representational choices and the methods being used for reasoning and learning.

Scalability and Teamwork: Despite considerable research, algorithms for RDK, SDM, or a combination of the two, find it difficult to scale to more complex domains. This is usually due to the space of possible options to be considered, e.g., the size of the data to be reasoned with by the RDK methods, and the size of the state-action space to be considered by the SDM methods. All of these challenges are complicated further when applications require a team of robots and humans to collaborate with each other. For instance, representational choices and reasoning algorithms may now need to carefully consider the capabilities of the teammates before making a decision. As described earlier, there are some promising avenues to be explored further. These include the computational modeling and use of principles such as relevance, persistence, and non-procrastination, which are well-known in cognitive systems, in the design of the desired integrated system. Such a system could then automatically determine the best use of available resources and algorithms depending on the domain attributes and tasks at hand.

Evaluation, Benchmarks, and Challenges: The complexity and connectedness of the components of the architectures and algorithms developed for RDK-for-SDM make it rather challenging to evaluate the representation, reasoning, and learning capabilities. A key direction for further research is the definition of common measures and benchmark tasks for the evaluation of such architectures; doing so would provide deeper insights into the development and use of such architectures. The evaluation measures will need to go beyond the basic measures such as accuracy (e.g., of task completion) and time (e.g., for planning, execution etc) to examine the connectedness of the components. These measures could, for instance, explore scalability to more complex domains and explanations while minimizing the amount of knowledge that needs to be encoded or used for reasoning. The benchmark tasks, in a similar manner, will need to challenge the robot to jointly perform multiple operations, e.g., use reasoning to guide knowledge acquisition, and use the learned knowledge to inform reasoning.

Acknowledgments

References

[Amiri *et al.*, 2019] Saeid Amiri, Sujay Bajracharya, Cihangir Goktolgal, Jesse Thomason, and Shiqi Zhang. Augmenting knowledge through statistical, goal-oriented human-robot dialog. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[Amiri *et al.*, 2020] Saeid Amiri, Mohammad Shokrolah Shirazi, and Shiqi Zhang. Learning and reasoning for robot sequential decision making under uncertainty. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[Anderson and Lebiere, 2014] John R Anderson and Christian J Lebiere. *The atomic components of thought*. Psychology Press, 2014.

[Bacchus *et al.*, 1999] Fahiem Bacchus, Joseph Y. Halpern, and Hector J. Levesque. Reasoning about Noisy Sensors and Effectors in the Situation Calculus. *Artificial Intelligence*, 111(1-2):171–208, 1999.

[Balai *et al.*, 2019] Evgenii Balai, Michael Gelfond, and Yuanlin Zhang. P-log: refinement and a new coherency condition. *Annals of Mathematics and Artificial Intelligence*, 86(1-3):149–192, 2019.

[Banihashemi *et al.*, 2018] Bita Banihashemi, Giuseppe De Giacomo, and Yves Lesperance. Abstraction of Agents Executing Online and their Abilities in Situation Calculus. In *International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, July 13-19, 2018.

[Baral *et al.*, 2009] Chitta Baral, Michael Gelfond, and Nelson Rushton. Probabilistic Reasoning with Answer Sets. *Theory and Practice of Logic Programming*, 9(1):57–144, January 2009.

[Barendregt and others, 1984] Hendrik P Barendregt et al. *The lambda calculus*, volume 3. North-Holland Amsterdam, 1984.

[Belle and Levesque, 2018] Vaishak Belle and Hector J. Levesque. Reasoning about discrete and continuous noisy sensors and effectors in dynamical systems. *Artificial Intelligence*, 262:189–221, 2018.

[Camacho *et al.*, 2019] Alberto Camacho, Rodrigo Toro Icarte, Toryn Q Klassen, Richard Anthony Valenzano, and Sheila A McIlraith. LTL and beyond: Formal languages

for reward function specification in reinforcement learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[Chitnis and Lozano-Pérez, 2020] Rohan Chitnis and Tomás Lozano-Pérez. Learning compact models for planning with exogenous processes. In *Conference on Robot Learning*, pages 813–822, 2020.

[Chitnis *et al.*, 2018] Rohan Chitnis, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrating human-provided information into belief state representation using dynamic factorization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[Colmerauer and Roussel, 1996] Alain Colmerauer and Philippe Roussel. The birth of Prolog. In *History of programming languages—II*, pages 331–367. ACM, 1996.

[den Broeck *et al.*, 2010] Guy Van den Broeck, Ingo Thon, Martijn van Otterlo, and Luc De Raedt. DTProbLog: A Decision-Theoretic Probabilistic Prolog. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

[Efthymiadis and Kudenko, 2013] Kyriakos Efthymiadis and Daniel Kudenko. Using plan-based reward shaping to learn strategies in starcraft: Broodwar. In *Proceedings of the 2013 IEEE Conference on Computational Inteligence in Games*, 2013.

[Eysenbach *et al.*, 2019] Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 15246–15257, 2019.

[Ferreira *et al.*, 2017] Leonardo A Ferreira, Reinaldo AC Bianchi, Paulo E Santos, and Ramon Lopez de Mantaras. Answer set programming for non-stationary markov decision processes. *Applied Intelligence*, 47(4):993–1007, 2017.

[Fierens *et al.*, 2015] Dann Fierens, Guy Van Den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. Inference and Learning in Probabilistic Logic Programs using Weighted Boolean Formulas. *Theory and Practice of Logic Programming*, 15(3):358–401, 2015.

[Fikes and Nilsson, 1971] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[Furelos-Blanco *et al.*, 2020] Daniel Furelos-Blanco, Mark Law, Alessandra Russo, Krysia Broda, and Anders Jonsson. Induction of subgoal automata for reinforcement learning. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[Garnelo *et al.*, 2016] Marta Garnelo, Kai Arulkumaran, and Murray Shanahan. Towards deep symbolic reinforcement learning. In *Deep Reinforcement Learning Workshop at the 30th Conference on Neural Information Processing Systems*, 2016.

[Gebser *et al.*, 2012] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice, Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan Claypool Publishers, 2012.

[Gelfond and Inclezan, 2013] Michael Gelfond and Daniela Inclezan. Some Properties of System Descriptions of AL_d. *Journal of Applied Non-Classical Logics, Special Issue on Equilibrium Logic and Answer Set Programming*, 23(1-2):105–120, 2013.

[Gelfond and Kahl, 2014] Michael Gelfond and Yulia Kahl. *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*. Cambridge University Press, 2014.

[Ghallab *et al.*, 2004] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Morgan Kaufmann Publishers Inc., 2004.

[Göbelbecker *et al.*, 2011] Moritz Göbelbecker, Charles Gretton, and Richard Dearden. A switching planner for combined task and observation planning. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 964–970, 2011.

[Gomez *et al.*, 2020] Rocio Gomez, Mohan Sridharan, and Heather Riley. What do you really want to do? Towards a Theory of Intentions for Human-Robot Collaboration. *Annals of Mathematics and Artificial Intelligence, special issue on commonsense reasoning*, 2020.

[Gopalan *et al.*, 2020] Nakul Gopalan, Eric Rosen, George Konidaris, and Stefanie Tellex. Simultaneously learning transferable symbols and language groundings from perceptual data for instruction following. In *Robotics: Science and System XVI*, 2020.

[Gorlin *et al.*, 2012] Andrey Gorlin, C. R. Ramakrishnan, and Scott A. Smolka. Model Checking with Probabilistic Tabled Logic Programming. *Theory and Practice of Logic Programming*, 12(4-5):681–700, 2012.

[Grounds and Kudenko, 2005] Matthew Grounds and Daniel Kudenko. Combining reinforcement learning with symbolic planning. In *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*, pages 75–86. Springer, 2005.

[Grzes and Kudenko, 2008] Marek Grzes and Daniel Kudenko. Plan-based reward shaping for reinforcement learning. In *2008 4th International IEEE Conference Intelligent Systems*, volume 2, pages 10–22. IEEE, 2008.

[Halpern, 2003] Joseph Halpern. *Reasoning about Uncertainty*. MIT Press, 2003.

[Hanheide *et al.*, 2017] Marc Hanheide, Moritz Göbelbecker, Graham S Horn, Andrzej Pronobis, Kristoffer Sjöö, et al. Robot task planning and explanation in open and uncertain worlds. *Artificial Intelligence*, 247:119–150, 2017.

[Helmert, 2006] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[Hoelscher *et al.*, 2018] Janine Hoelscher, Dorothea Koert, Jan Peters, and Joni Pajarinen. Utilizing human feedback in pomdp execution and specification. In *IEEE-RAS International Conference on Humanoid Robots*, 2018.

[Illanes *et al.*, 2020] León Illanes, Xi Yan, Rodrigo Toro Icarte, and Sheila A McIlraith. Symbolic plans as high-level instructions for reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 540–550, 2020.

[Juba, 2016] Brendan Juba. Integrated Common Sense Learning and Planning in POMDPs. *Journal of Machine Learning Research*, 17(96):1–37, 2016.

[Kaelbling *et al.*, 1998] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[Konidaris *et al.*, 2018] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.

[Laird, 2012] John E Laird. *The Soar Cognitive Architecture*. The MIT Press, 2012.

[Langley and Choi, 2006] Patrick Langley and Dongkyu Choi. An Unified Cognitive Architecture for Physical Agents. In *The Twenty-first National Conference on Artificial Intelligence (AAAI)*, 2006.

[Leonetti *et al.*, 2016] Matteo Leonetti, Luca Iocchi, and Peter Stone. A synthesis of automated planning and reinforcement learning for efficient, robust decision-making. *Artificial Intelligence*, 241:103–130, 2016.

[Lyu *et al.*, 2019] Daoming Lyu, Fangkai Yang, Bo Liu, and Steven Gustafson. SDRL: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[McCarthy, 1978] John McCarthy. History of lisp. In *History of programming languages*, pages 173–185. 1978.

[McDermott *et al.*, 1998] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. Pddl-the planning domain definition language, 1998.

[McGuinness *et al.*, 2004] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. 2004.

[Mericli *et al.*, 2014] Cetin Mericli, Steven D Klee, Jack Paparian, and Manuela Veloso. An interactive approach for situated task specification through verbal instructions. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems (AAMAS)*, 2014.

[Milch *et al.*, 2006] Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: Probabilistic Models with Unknown Objects. In *Statistical Relational Learning*. MIT Press, 2006.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[Mombourquette *et al.*, 2017] Brent Mombourquette, Christian Muise, and Sheila A McIlraith. Logical filtering and smoothing: state estimation in partially observable domains. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3613–3621, 2017.

[Nath and Domingos, 2009] Aniruddh Nath and Pedro Domingos. A Language for Relational Decision Theory. In *International Workshop on Statistical Relational Learning*, Leuven, Belgium, July 2-4, 2009.

[Poole, 2000] David Poole. Abducing through Negation as Failure: Stable Models within the Independent Choice Logic. *Journal of Logic Programming*, 44(1-3):5–35, 2000.

[Puterman, 2014] Martin L Puterman. *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[Raedt and Kimmig, 2015] Luc De Raedt and Angelika Kimmig. Probabilistic Logic Programming Concepts. *Machine Learning*, 100(1):5–47, 2015.

[Richardson and Domingos, 2006] Matthew Richardson and Pedro Domingos. Markov Logic Networks. *Machine learning*, 62(1), 2006.

[Samadi *et al.*, 2012] Mehdi Samadi, Thomas Kollar, and Manuela Veloso. Using the web to interactively learn to find objects. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*, 2012.

[Sanner and Kersting, 2010] Scott Sanner and Kristian Kersting. Symbolic dynamic programming for first-order pomdps. In *Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

[Sanner, 2010] Scott Sanner. Relational dynamic influence diagram language (RDDL): Language description. 2010.

[Sarathy and Scheutz, 2018] Vasanth Sarathy and Matthias Scheutz. A Logic-based Computational Framework for Inferring Cognitive Affordances. *IEEE Transactions on Cognitive and Developmental Systems*, 10(1):26–43, March 2018.

[Saribatur *et al.*, 2019] Zeynep Saribatur, Volkan Patoglu, and Esra Erdem. Finding Optimal Feasible Global Plans for Multiple Teams of Heterogeneous Robots using Hybrid Reasoning: An Application to Cognitive Factories. *Autonomous Robots*, 43(1):213–238, 2019.

[Scheutz *et al.*, 2007] Matthias Scheutz, Paul Schermerhorn, James Kramer, and David Anderson. First Steps Towards Natural Human-Like HRI. *Autonomous Robots*, 22(4):411–423, 2007.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[She and Chai, 2017] Lanbo She and Joyce Chai. Interactive learning of grounded verb semantics towards human-robot communication. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1634–1644, 2017.

[Sridharan and Meadows, 2018] Mohan Sridharan and Ben Meadows. Knowledge Representation and Interactive Learning of Domain Knowledge for Human-Robot Collaboration. *Advances in Cognitive Systems*, 7:77–96, December 2018.

[Sridharan *et al.*, 2017] Mohan Sridharan, Ben Meadows, and Rocio Gomez. What can I not do? Towards an Architecture for Reasoning about and Learning Affordances. In *International Conference on Automated Planning and Scheduling*, Pittsburgh, USA, June 18-23, 2017.

[Sridharan *et al.*, 2019] Mohan Sridharan, Michael Gelfond, Shiqi Zhang, and Jeremy Wyatt. Reba: A refinement-based architecture for knowledge representation and reasoning in robotics. *Journal of Artificial Intelligence Research*, 65:87–180, 2019.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Thomason *et al.*, 2015] Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. Learning to interpret natural language commands through human-robot dialog. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1923–1929, 2015.

[Towell and Shavlik, 1994] Geoffrey G Towell and Jude W Shavlik. Knowledge-based artificial neural networks. *Artificial intelligence*, 70(1-2):119–165, 1994.

[Wang *et al.*, 2019] Yi Wang, Shiqi Zhang, and Joohyung Lee. Bridging commonsense reasoning and probabilistic planning via a probabilistic action language. *Theory and Practice of Logic Programming (TPLP)*, 19(5-6):1090–1106, 2019.

[Yang *et al.*, 2018] Fangkai Yang, Daoming Lyu, Bo Liu, and Steven Gustafson. PEORL: integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[Younes and Littman, 2004] Håkan LS Younes and Michael L Littman. PPDDL1.0: The language for the probabilistic part of IPC-4. In *Proceedings of the 2004 International Planning Competition*, 2004.

[Zhang and Stone, 2015] Shiqi Zhang and Peter Stone. CORPP: commonsense reasoning and probabilistic planning, as applied to dialog with a mobile robot. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

[Zhang *et al.*, 2015] Shiqi Zhang, Mohan Sridharan, and Jeremy L Wyatt. Mixed logical inference and probabilistic planning for robots in unreliable worlds. *IEEE Transactions on Robotics*, 31(3):699–713, 2015.

[Zhang *et al.*, 2017] Shiqi Zhang, Piyush Khandelwal, and Peter Stone. Dynamically constructed (PO)MDPs for adaptive robot planning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[Zhang *et al.*, 2019] Haodi Zhang, Zihang Gao, Yi Zhou, Hao Zhang, Kaishun Wu, and Fangzhen Lin. Faster and safer training by embedding high-level knowledge into deep reinforcement learning. *arXiv preprint arXiv:1910.09986*, 2019.

[Zhu, 2012] Weijun Zhu. *Plog: Its algorithms and applications*. PhD thesis, Texas Tech University, 2012.