A survey of knowledge-based sequential decision-making under uncertainty

Zhang, Shiqi; Sridharan, Mohan

DOI:
10.1002/aaai.12053

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (Harvard):
Zhang, S & Sridharan, M 2022, 'A survey of knowledge-based sequential decision-making under uncertainty', *AI Magazine*, vol. 43, no. 2, pp. 249-266. https://doi.org/10.1002/aaai.12053

Received: 8 March 2021 | Accepted: 31 March 2021

DOI: 10.1002/aaai.12053

ARTICLE

A survey of knowledge-based sequential decision-making under uncertainty

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

[1]Department of Computer Science, The State University of New York at Binghamton, Binghamton, New York, USA

[2]Intelligent Robotics Lab, School of Computer Science, University of Birmingham, Birmingham, UK

Correspondence
Shiqi Zhang, Department of Computer Science, The State University of New York at Binghamton, Binghamton, NY, USA.
Email: zhangs@binghamton.edu

Funding information
NSF, Grant/Award Number: NRI-1925044; Ford Motor Company; OPPO (Faculty Research Award); US Office of Naval Research, Grant/Award Numbers: N00014-17-1-2434, N00014-20-1-2390; Asian Office of Aerospace Research and Development, Grant/Award Number: FA2386-16-1-4071; UK Engineering and Physical Sciences Research Council, Grant/Award Number: EP/S032487/1

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 [PP] and reinforcement learning [RL]) 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.

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, that is, **probabilistic planning (PP)** and **reinforcement learning (RL)** 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, for example, "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, for example, statistical relational AI, logic programming, RL, and robotics. Also, these algorithms have not always considered the needs of agents in dynamic domains, for example, 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. Also, figuring out how best to combine the principles of RDK and SDM remains an open grand challenge in AI, with connections to deep philosophical questions about the representation, manipulation/use, and acquisition of knowledge in humans and machines, and about the broader impacts of such methods. This survey paper seeks to stimulate cross-pollination of ideas between the communities working on different aspects of this grand challenge, by highlighting the key achievements and open problems. To achieve this objective while keeping the list of related papers manageable, we limit our scope to algorithms that use RDK to facilitate SDM, and focus on the following question:

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

We also limit our attention to algorithms developed for an agent making sequential decisions under uncertainty in dynamic domains. Furthermore, to explain the key concepts, we often draw on our expertise in developing such methods for robots. Figure 1 provides an overview of the survey's theme[1]. We begin by describing some key concepts related to RDK and SDM systems ("Background"), followed by the factors we use to characterize the RDK-for-SDM systems ("Characteristic factors"). We then describe some representative RDK-for-SDM systems ("RDK-for-SDM methods") and discuss open problems in the design and use of such systems ("Challenges and opportunities").

BACKGROUND

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

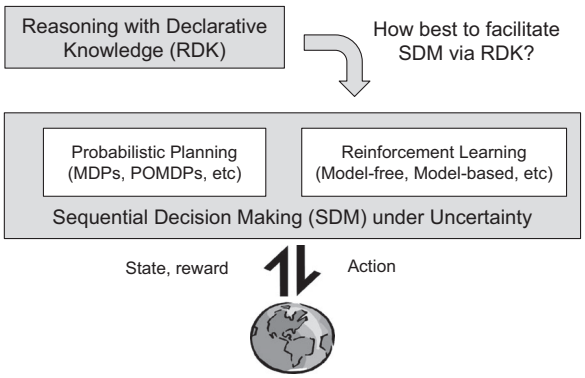

FIGURE 1 An overview of this survey: *reasoning with declarative knowledge* (RDK) for *sequential decision making* (SDM)

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, for example, 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*, that is, computing a sequence of actions to achieve any given goal, monitoring the execution of actions, and replanning if needed. This is different from PP that computes and uses policies to choose actions in any given state or belief state ("Sequential decision-making").

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)," that is, 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 et al. 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, that is, 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, that is, 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 languages

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 (Haslum et al. 2019), 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, for example, 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, Nau, and Traverso 2016), 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, Ramakrishnan, and Smolka 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, Gelfond, and Rushton 2009). We will discuss some of these later in this paper.

Sequential decision-making

We consider two classes of SDM methods: **PP** (Puterman 2014) and **RL** (Sutton and Barto 2018), depending on the availability of world models. A common assumption in these methods is the first-order Markov property, that is, the next state is assumed to be conditionally independent of all previous states given the current state. Also, actions are assumed to be nondeterministic, that is, they do not always provide the expected outcomes, and the state is assumed to be fully or partially observable. Unlike classical planning (see "Reasoning with declarative knowledge"), 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 $\langle S, \mathcal{A}, T, R\rangle$ whose elements define the set of states, set of actions, the probabilistic state transition function $T : S \times \mathcal{A} \times S \to [0, 1]$, and the reward specification $R : S \times \mathcal{A} \times S' \to \mathfrak{R}$. 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, Littman, and Cassandra 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 $O : \mathcal{S} \times \mathcal{A} \times Z \rightarrow [0, 1]$ 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:

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

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, for example, $R(s, a)$ 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, that is, $Q(s, a)$, 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, for example, to approximate the value function (Mnih et al. 2015), and deep policy-based methods, for example, (Schulman et al. 2015; 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.

CHARACTERISTIC FACTORS

Before we discuss RDK-for-SDM algorithms and systems, we describe the factors that we use to characterize these systems. 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 individual factors are orthogonal to each other, that is, the choice of a particular value for one factor can (for the most part) be made independent of the choice of value for the other factors.

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.

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 and 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, for example, 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.

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.

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 require 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 how RDK is used for SDM methods. For instance, 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, that is, 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 for 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*, for example, probability distributions over the underlying states (Kaelbling, Littman, and Cassandra 1998), and implicit representations computed with neural networks (Hausknecht and Stone 2015). 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, for example, 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.

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 versus 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 versus 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, for example, 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**, for example, 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**, for example, 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, for example, 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.

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, whereas 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, for example, reasoning can be used to trigger, inform, and guide efficient knowledge acquisition. However, they also present some challenges, for example, in suitably reconciling differences between existing knowledge and learned knowledge. These advantages and challenges

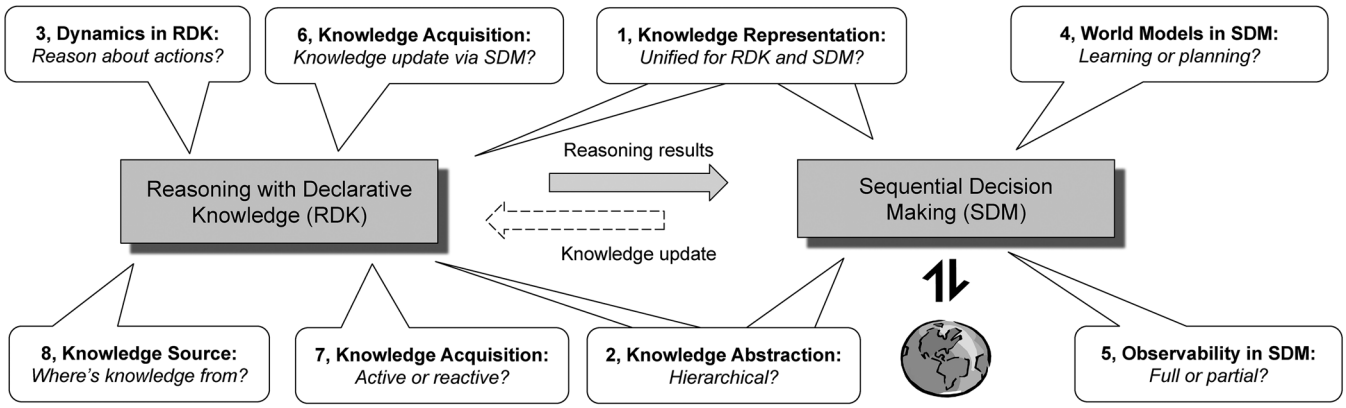

FIGURE 2 Characteristic factors in the development of RDK-for-SDM methods. The individual factors are discussed in details in "Characteristic factors," and are also used for the discussions of representative algorithms in "RDK-for-SDM methods"

are discussed below in the context of representative RDK-for-SDM methods.

RDK-FOR-SDM METHODS

In this section, we review some representative RDK-for-SDM systems by grouping them based on their primary contributions. First, "Representation-focused systems" discusses some systems that primarily focus on the knowledge representation challenges in RDK-for-SDM. "Reasoning-focused systems" and "Knowledge acquisition-focused systems" 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.

Representation-focused systems

As stated in "Reasoning with declarative knowledge," many generic hybrid representations have been developed to support both logical and probabilistic reasoning with knowledge and uncertainty.

Unified RDK-for-SDM representations

Developing a unified representation for RDK and SDM maps to developing a unified representation for logical and probabilistic reasoning, which has been a fundamental problem in robotics and AI for decades. Frameworks and methods based on unified representations provide significant expressive power, but they also impose a significant computational burden despite the ongoing work on developing more efficient (and often approximate) reasoning algorithms for such unified paradigms.

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, 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, Halpern, and Levesque 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, for example, first-order relational POMDPs that leverage symbolic programming for the specification of POMDPs with first-order abstractions (Juba 2016; Sanner and Kersting 2010).

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 "Characteristic factors."

		Uni. Rep.	Abs. Rep.	Dyn. RDK	RL SDM	Par. Obs.	On. Acq.	ML RDK	Rew. RDK
Representation	(Younes and Littman 2004)	●	○	/	○	○	○	○	○
	(Sanner 2010)	●	○	/	○	◐	○	○	●
	(Baral, Gelfond, and Rushton 2009)	●	○	/	○	◐	○	○	○
	(Wang, Zhang, and Lee 2019)	●	○	/	○	◐	●	●	●
	(Zhang, Khandelwal, and Stone 2017)	●	○	○	○	◐	●	○	●
Reasoning	(Sridharan et al. 2019)	○	●	●	○	●	●	●	○
	(Illanes et al. 2020)	○	●	●	●	○	○	○	○
	(Yang et al. 2018; Lyu et al. 2019)	○	●	●	●	○	●	○	○
	(Furelos-Blanco et al. 2020)	○	●	●	●	○	●	●	●
	(Göbelbecker, Gretton, and Dearden 2011)	○	○	●	○	●	○	○	○
	(Garnelo, Arulkumaran, and Shanahan 2016)	○	○	○	●	○	●	●	○
	(Chitnis, Kaelbling, and Lozano-Pérez 2018)	○	○	○	○	●	●	○	○
	(Zhang, Sridharan, and Wyatt 2015; Zhang and Stone 2015)	○	○	○	○	●	○	○	○
	(Amiri, Shirazi, and Zhang 2020)	○	○	○	○	●	●	●	○
	(Grounds and Kudenko 2005)	○	●	●	●	○	○	○	●
	(Hoelscher et al. 2018)	○	○	○	○	●	●	○	●
	(Icarte et al. 2018; Camacho et al. 2019)	○	○	●	●	○	○	○	●
	(Zhang et al. 2019)	○	○	○	●	○	○	○	○
	(Leonetti, Iocchi, and Stone 2016)	○	○	●	●	○	●	○	○
	(Eysenbach, Salakhutdinov, and Levine 2019)	○	●	●	●	○	●	●	○
Acquisition	(Konidaris, Kaelbling, and Lozano-Perez 2018; Gopalan et al. 2020)	○	●	●	○	○	●	●	○
	(Thomason et al. 2015; Amiri et al. 2019)	○	○	○	○	●	●	○	○
	(She and Chai 2017)	○	○	●	●	○	●	●	○
	(Mericli et al. 2014)	○	○	●	○	○	●	○	○
	(Samadi, Kollar, and Veloso 2012)	○	○	○	○	○	●	●	○

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

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, Gelfond, and Rushton 2009). P-log has been used to specify MDPs for SDM tasks, for example, 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, Gelfond, and Zhang 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 PP with infinite horizons requires a significant engineering effort.

Linked RDK-for-SDM representations

As stated earlier in the context of Factor 1 in "Representational factors," 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, Gretton, and Dearden 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, Patoglu, and Erdem 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* (Freeman and Pfenning 1991). This principle has also been explored in fields such as software engineering and programming languages (Lovas 2010; Lovas and Pfenning 2010), but without any theories of actions and change that are important in robotics and AI. One 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, Giacomo, and Lesperance 2018). The principle of refinement has also been used to construct abstractions of ASP programs, with the objective of shrinking the domain size while preserving the structure of the rules (Saribatur, Eiter, and Schuller 2021). An example of tightly coupled systems in robotics is the refinement-based architecture (REBA) that 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 with limited commonsense domain knowledge at the coarse-resolution 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. The formal definition of refinement, zooming, and the connections between the transition diagrams enables smooth transfer of relevant information and control, and improves scalability. It also enables the robot to represent and reason with sophisticated cognitive theories in the coarse resolution, for example, with an adaptive theory of intentions (Gomez, Sridharan, and Riley 2021).

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.

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 for state estimation

RDK methods can be used for estimating the current world state in order to guide SDM. Although practical domains often include many objects with different attributes, and multiple relationships between these objects, only a small subset of these objects, attributes, and relationships may be relevant to any particular task that an agent has to perform. Researchers have, therefore, used RDK methods to identify the task-relevant information to guide state estimation and SDM. For instance, the state of the world has been represented using knowledge predicates and assumptive predicates, which were then used for planning based on declarative action knowledge and probabilistic rules (Hanheide et al. 2017). This approach, embedded within a three-layered architecture, was used for applications such as object search, and semantic mapping. Another example is the CORPP system that uses P-log (Balai, Gelfond, and Zhang 2019) to reason with probabilistic declarative knowledge in order to generate informative priors for POMDP planning (Zhang and Stone 2015). In a human–robot dialog domain, CORPP demonstrated that commonsense knowledge, such as "people like coffee in the mornings" and "office doors are closed over weekends", is useful for guiding dialog actions. Other researchers have exploited factored state spaces to develop algorithms that use probabilistic declarative knowledge to efficiently compute informative priors for POMDPs (Chitnis, Kaelbling, and Lozano-Pérez 2018). In particular, they developed an efficient belief state representation that dynamically selects an appropriate factoring to guide SDM, and demonstrated its effectiveness in robot cooking tasks. These methods separate the variables modeled at different levels and (manually) link relevant variables between the levels, improving scalability and dynamic response. These links enable 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 establishing the 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, Kaelbling, and Lozano-Pérez 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, Muise, and McIlraith 2017). There is also work on an action language called pBC+, which supports the definition of MDPs and POMDPs over finite and infinite horizons (Wang, Zhang, and Lee 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, that is, 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, for example, 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, for example, 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 (Efthymiadis and Kudenko 2013; Grounds and Kudenko 2005; Grzes and Kudenko 2008). 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, Khandelwal, and Stone 2017). Another system, 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 of a robot arm stacking boxes. There has also been work on "reward machines" that uses linear temporal logic to represent and reason with declarative knowledge, especially temporal constraints implied by phrases such as "until" and "eventually," in order to automatically generate additional rewards for RL that are potentially non-Markovian (Icarte et al. 2022).

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, Iocchi, and Stone 2016); this approach has been evaluated on real robots navigating office environments to locate people of interest.

An algorithm called GDQ uses action knowledge to generate artificial, "oppotimistic" experience to give RL agents a warm-up learning experience before letting them interact with the real world (Hayamizu et al. 2021). Another similar approach uses RDK to guide an agent's exploration behavior (formulated as SDM) in nonstationary 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, Meadows, and Gomez 2017).

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).

Knowledge acquisition while acting

Some RDK-for-SDM systems allow the agent to acquire knowledge while also simultaneously reasoning and executing actions in dynamic domains. Such systems can often support online and offline knowledge acquisition, with active and reactive aspects. For example, ASP-based non-monotonic logical reasoning has been used to guide relational RL (i.e., SDM) and decision-tree induction in order to learn previously unknown actions and domain axioms; this knowledge is subsequently used for RDK (Sridharan and Meadows 2018). This system supports reactive knowledge acquisition, with reasoning used to trigger and guide learning only when some unexpected outcomes are observed (e.g., to acquire knowledge of previously unknown constraints), as well as active, online knowledge acquisition, with the robot acquiring previously unknown knowledge based on explicit exploration (e.g., of the potential effects of new actions).

Knowledge acquisition from experience

There is a well-established literature of RDK-for-SDM systems, including many described above, acquiring or revising knowledge of domain dynamics in a supervised or semi-supervised *training* phase. The robot could, for instance, be asked to execute different actions and observe the corresponding outcomes in scenarios with known ground truth information (Sridharan et al. 2019; Zhang, Khandelwal, and Stone 2017). More recently, some RDK-for-SDM systems have built on recent developments

in data-driven methods (e.g., deep learning and RL) to acquire knowledge. 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, Salakhutdinov, and Levine 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, Kaelbling, and Lozano-Perez 2018). Knowledge acquisition in these systems is often offline (i.e., batch of data collected from the robot is processed offline to extract knowledge); this acquisition can be achieved by targeted exploration (i.e., active) or 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, knowledge acquisition is active and offline, and requires significantly fewer training samples compared to end-to-end systems. In another RDK-for-SDM system, non-monotonic logical reasoning is used to guide deep network learning and active acquisition of previously unknown axioms describing the behavior of these networks (Mota, Sridharan, and Leonardis 2021; Riley and Sridharan 2019).

Knowledge acquisition from humans, web, and other sources

For some RDK-for-SDM systems, 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). In a 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). In other work, human (verbal) descriptions of observed robot behavior have been used to extract knowledge of previously unknown actions and action effects, which is merged with existing knowledge in the RDK component (Sridharan and Meadows 2018). More recent work in the context of a system enabling an agent to respond to a human's questions about its decisions and evolution of beliefs, has also enabled the agent to interactively construct questions to resolve ambiguities in the human's questions (Mota and Sridharan 2021).

Some researchers have equipped their RDK-for-SDM systems with the ability to acquire domain knowledge using data available on the Web (Samadi, Kollar, and Veloso 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."

CHALLENGES AND OPPORTUNITIES

Over the last few decades, researchers have made significant progress in developing sophisticated methods for RDK and for SDM 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 "Representation-focused systems," existing methods integrating RDK and SDM methods are predominantly based on unified or linked representations. General-purpose methods often use a unified representation and associated reasoning methods for different descriptions of domain knowledge, for example, 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. These hybrid representations can also form the foundation of modern neuro-symbolic AI (Garcez et al. 2019; Hitzler and Sarker 2022) methods for reasoning and learning.

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 are 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 (see discussion in "Reasoning-focused systems" and "Knowledge acquisition-focused systems"); 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 (Laird et al. 2017; Langley 2017).

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. Humans can be a rich source of information but there is often a nontrivial 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 (Rosenthal, Veloso, and Dey 2012). 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 that integrate RDK and SDM focus on decision making (or reasoning) tasks. There are 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 (Garrett et al. 2021). 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 with predictive models and learning can be used to identify (on demand) and revise the relevant variables in the control laws for the tasks at hand (Mathew et al. 2019; Sidhik, Sridharan, and Ruiken 2021). At the same time, real-world control tasks often require a very different representation of domain attributes, for example, 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, for example, 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 (Langley 2017), in the design of the desired integrated system (Blount, Gelfond, and Balduccini 2015; Gomez, Sridharan, and Riley 2021). Such a system could then automatically determine the best use of available resources and algorithms depending on the domain attributes and tasks at hand.

Explainability and trust

With the increasing use of AI and machine learning methods in different applications, there is renewed focus within the research community on enabling humans to understand the operation of these methods (Anjomshoae et al. 2019; Miller 2019). Issues such as explainability or trust remain open problems for RDK-for-SDM systems, especially those that integrate reasoning and learning in complex domains. At the same time, the design of these systems provides promising research threads to be

explored further. For instance, the use of logics for representing and reasoning with commonsense knowledge in the RDK component of such systems provides a foundation for making the associated reasoning and learning more transparent. Research also indicates that the underlying representation and established knowledge representation tools can be exploited to reliably and efficiently trace beliefs and provide on-demand explanations at the desired level of abstraction, before, during, or after task execution (Sridharan and Meadows 2019; Mota, Sridharan, and Leonardis 2021). A key challenge would be rigorously study trust and explainability from the viewpoint of a nonexpert human interacting with these systems.

Evaluation measures and benchmarks

The complexity of the components of RDK-for-SDM systems, and the connections between of these components, make it rather challenging to isolate and evaluate the impact of the underlying representation, reasoning methods, and learning methods. Often, the observed performance of a particular algorithm (e.g., for planning) is influenced by the design of this algorithm and the connections between this algorithm and other methods in the system. A key direction for further research is the definition of common measures and 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 measuring the accuracy and computational efficiency of individual components (e.g., planning and task completion accuracy, learning rate, execution time) to examine the effects of the links between the components. These measures could, for instance, explore scalability to more complex domains and tasks. Here, complexity could refer to the type and amount of knowledge encoded in the system; the type, duration, and number of operations to be performed by the robot; and the number and duration of interactions between the different components (of the system) required to complete the task. In addition, evaluation could consider qualitative measures of performance, for example, the ability to complete different tasks, the ability to provide interactive explanations, or the satisfaction of humans interacting with the system.

The benchmarks used for evaluation should not be limited to providing datasets or scenarios for evaluating individual algorithms. Similar to the evaluation measures, the benchmarks should instead challenge the robot to explore and use the interplay between the different components of the system being evaluated, for example, use reasoning to guide knowledge acquisition, and use the learned knowledge to inform reasoning. In this context, many different domains hold promise in terms of being suitable for evaluation of such RDK-for-SDM systems; these include *games* (Yang et al. 2018; Zhang et al. 2019), *interactive dialog* (Amiri et al. 2019; Zhang and Stone 2015), *robot navigation and exploration* (Hanheide et al. 2017; Leonetti, Iocchi, and Stone 2016), and *scene understanding* (Chitnis, Kaelbling, and Lozano-Pérez 2018; Jiang et al. 2019; Mota and Sridharan 2019; Mota, Sridharan, and Leonardis 2021).

ACKNOWLEDGMENTS

Related work in the Autonomous Intelligent Robotics (AIR) group at SUNY Binghamton was supported in part by grants from NSF (NRI-1925044), Ford Motor Company (URP Awards), OPPO (Faculty Research Award), and SUNY RF. Related work in the Intelligent Robotics Lab (IRLab) at the University of Birmingham was supported in part by the U.S. Office of Naval Research Science of Autonomy Awards N00014-17-1-2434 and N00014-20-1-2390, the Asian Office of Aerospace Research and Development award FA2386-16-1-4071, and the UK Engineering and Physical Sciences Research Council award EP/S032487/1. The authors thank collaborators on research projects that led to the development of the ideas described in this paper.

CONFLICT OF INTEREST

The authors have no conflicts of interest to report.

ORCID

Shiqi Zhang https://orcid.org/0000-0003-4110-8213
Mohan Sridharan https://orcid.org/0000-0001-9922-8969

ENDNOTE

[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.

REFERENCES

Amiri, S., S. Bajracharya, C. Goktolgal, J. Thomason, and S. Zhang. 2019. "Augmenting Knowledge through Statistical, Goal-oriented Human–Robot Dialog." In *Priceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Amiri, S., M. S. Shirazi, and S. Zhang. 2020. "Learning and Reasoning for Robot Sequential Decision Making under Uncertainty." In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*.

Anderson, J. R., and C. J. Lebiere. 2014. *The Atomic Components of Thought*. New York, NY, United States: Psychology Press.

Anjomshoae, S., A. Najjar, D. Calvaresi, and K. Framling. 2019. "Explainable Agents and Robots: Results from a Systematic Literature Review." In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Montreal, Canada.

Bacchus, F., J. Y. Halpern, and H. J. Levesque. 1999. "Reasoning about Noisy Sensors and Effectors in the Situation Calculus." *Artificial Intelligence* 111(1-2): 171–208.

Balai, E., M. Gelfond, and Y. Zhang. 2019. "P-log: Refinement and a New Coherency Condition." *Annals of Mathematics and Artificial Intelligence* 86(1-3): 149–92.

Banihashemi, B., G. D. Giacomo, and Y. Lesperance. 2018. "Abstraction of Agents Executing Online and their Abilities in Situation Calculus." In Proceedings of the International Joint Conference on Artificial Intelligence, Stockholm, Sweden. July 13–9.

Baral, C., M. Gelfond, and N. Rushton. 2009. "Probabilistic Reasoning with Answer Sets." *Theory and Practice of Logic Programming* 9(1): 57–144. January.

Barendregt, H. P. 1984. *The Lambda Calculus*, Volume 3. Amsterdam: North-Holland Publishing Company, Elsevier.

Belle, V., and H. J. Levesque. 2018. "Reasoning about Discrete and Continuous Noisy Sensors and Effectors in Dynamical Systems." *Artificial Intelligence* 262: 189–221.

Blount, J., M. Gelfond, and M. Balduccini. 2015. "A Theory of Intentions for Intelligent Agents." In *Proceedings of the International Conference on Logic Programming and Nonmonotonic Reasoning*, 134–42. Springer.

Camacho, A., R. T. Icarte, T. Q. Klassen, R. A. Valenzano, and S. A. McIlraith. 2019. "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)*.

Chitnis, R., L. P. Kaelbling, and T. Lozano-Pérez. 2018. "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)*.

Chitnis, R., and T. Lozano-Pérez. 2020. "Learning Compact Models for Planning with Exogenous Processes." In *Proceedings of the Conference on Robot Learning*, 813–22.

Colmerauer, A., and P. Roussel. 1996. "The Birth of Prolog." In *History of Programming Languages—II*, 331–67. New York, NY, United States: ACM.

den Broeck, G. V., I. Thon, M. van Otterlo, and L. D. Raedt. 2010. "DTProbLog: A Decision-Theoretic Probabilistic Prolog." In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Efthymiadis, K., and D. Kudenko. 2013. "Using Plan-based Reward Shaping to Learn Strategies in Starcraft: Broodwar." In *Proceedings of the 2013 IEEE Conference on Computational Intelligence in Games*.

Eysenbach, B., R. R. Salakhutdinov, and S. Levine. 2019. "Search on the Replay Buffer: Bridging Planning and Reinforcement Learning." In *Proceedings of the Advances in Neural Information Processing Systems*, 15246–57.

Ferreira, L. A., R. A. Bianchi, P. E. Santos, and R. L. de Mantaras. 2017. "Answer Set Programming for Non-stationary Markov Decision Processes." *Applied Intelligence* 47(4): 993–1007.

Fierens, D., G. V. D. Broeck, J. Renkens, D. Shterionov, B. Gutmann, I. Thon, G. Janssens, and L. D. Raedt. 2015. "Inference and Learning in Probabilistic Logic Programs using Weighted Boolean Formulas." *Theory and Practice of Logic Programming* 15(3): 358–401.

Fikes, R. E., and N. J. Nilsson. 1971. "Strips: A New Approach to the Application of Theorem Proving to Problem Solving." *Artificial intelligence* 2(3-4): 189–208.

Freeman, T., and F. Pfenning. 1991. "Refinement Types for ML." In *Proceedings of the ACM SIGPLAN 1991 Conference on Programming Language Design and Implementation*, 268–77.

Furelos-Blanco, D., M. Law, A. Russo, K. Broda, and A. Jonsson. 2020. "Induction of Subgoal Automata for Reinforcement Learning." In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*.

Garcez, A., M. Gori, L. Lamb, L. Serafini, M. Spranger, and S. Tran. 2019. "Neural-symbolic Computing: An Effective Methodology for Principled Integration of Machine Learning and Reasoning." *Journal of Applied Logics* 6(4): 611–31.

Garnelo, M., K. Arulkumaran, and M. Shanahan. 2016. "Towards Deep Symbolic Reinforcement Learning." In *Proceedings of the Deep Reinforcement Learning Workshop at the 30th Conference on Neural Information Processing Systems*.

Garrett, C. R., R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. 2021. "Integrated Task and Motion Planning." *Annual Review of Control, Robotics, and Autonomous Systems* 4: 265–93.

Gebser, M., R. Kaminski, B. Kaufmann, and T. Schaub. 2012. *Answer Set Solving in Practice, Synthesis Lectures on Artificial Intelligence and Machine Learning*. San Rafael, California: Claypool Publishers.

Gelfond, M., and D. Inclezan. 2013. "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): 105–20.

Gelfond, M., and Y. Kahl. 2014. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. Cambridge, United Kingdom: Cambridge University Press.

Ghallab, M., D. Nau, and P. Traverso. 2016. *Automated Planning and Acting*. Cambridge, United Kingdom: Cambridge University Press.

Göbelbecker, M., C. Gretton, and R. Dearden. 2011. "A Switching Planner for Combined Task and Observation Planning." In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 964–70.

Gomez, R., M. Sridharan, and H. Riley. 2021 "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* 89(1): 179–208. February.

Gopalan, N., E. Rosen, G. Konidaris, and S. Tellex. 2020. "Simultaneously Learning Transferable Symbols and Language Groundings from Perceptual Data for Instruction Following." In *Proceedings of the Robotics: Science and System XVI*.

Gorlin, A., C. R. Ramakrishnan, and S. A. Smolka. 2012. "Model Checking with Probabilistic Tabled Logic Programming." *Theory and Practice of Logic Programming* 12(4-5): 681–700.

Grounds, M., and D. Kudenko. 2005. "Combining Reinforcement Learning with Symbolic Planning." In *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*, 75–86. New York, NY, United States: Springer.

Grzes, M., and D. Kudenko. 2008. "Plan-based Reward Shaping for Reinforcement Learning." In *Proceedings of the 2008 4th International IEEE Conference Intelligent Systems*, Volume 2, 10–22. IEEE.

Halpern, J. 2003. *Reasoning about Uncertainty*. Cambridge, MA United States: MIT Press.

Hanheide, M., M. Göbelbecker, G. S. Horn, A. Pronobis, K. Sjöö, A. Aydemir, P. Jensfelt, et al. 2017. "Robot Task Planning and Explanation in Open and Uncertain Worlds." *Artificial Intelligence* 247: 119–50.

Haslum, P., N. Lipovetzky, D. Magazzeni, and C. Muise. 2019. "An Introduction to the Planning Domain Definition Language." *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13(2): 1–187.

Hausknecht, M., and P. Stone. 2015. "Deep Recurrent Q-learning for Partially Observable MDPs." In *Proceedings of the AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents (AAAI-SDMIA15)*.

Hayamizu, Y., S. Amiri, K. Chandan, K. Takadama, and S. Zhang. 2021. "Guiding Robot Exploration in Reinforcement Learning Via Automated Planning." In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Helmert, M. 2006. "The Fast Downward Planning System." *Journal of Artificial Intelligence Research* 26: 191–246.

Hitzler, P., and M. K. Sarker. 2022. *Neuro-symbolic Artificial Intelligence: The State of the Art*. Amsterdam, Netherlands: IOS Press.

Hoelscher, J., D. Koert, J. Peters, and J. Pajarinen. 2018. "Utilizing Human Feedback in POMDP Execution and Specification." In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*.

Icarte, R. T., T. Klassen, R. Valenzano, and S. McIlraith. 2018. "Using Reward Machines for High-level Task Specification and Decomposition in Reinforcement Learning." In *Proceedings of the International Conference on Machine Learning (ICML)*, 2112–21.

Icarte, R. T., T. Q. Klassen, R. Valenzano, and S. A. McIlraith. 2022. "Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning." *Journal of Artificial Intelligence Research* 73: 173–208.

Illanes, L., X. Yan, R. T. Icarte, and S. A. McIlraith. 2020. "Symbolic Plans as High-level Instructions for Reinforcement Learning." In *Proceedings of the International Conference on Automated Planning and Scheduling*, Volume 30, 540–50.

Jiang, Y., N. Walker, J. Hart, and P. Stone. 2019. "Open-world Reasoning for Service Robots." In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, Volume 29, 725–33.

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

Kaelbling, L. P., M. L. Littman, and A. R. Cassandra. 1998. "Planning and acting in partially observable stochastic domains." *Artificial Intelligence* 101(1-2): 99–134.

Konidaris, G., L. P. Kaelbling, and T. Lozano-Perez. 2018. "From Skills to Symbols: Learning Symbolic Representations for Abstract High-level Planning." *Journal of Artificial Intelligence Research* 61: 215–89.

Laird, J. E. 2012. *The Soar Cognitive Architecture*. Cambridge, MA United States: The MIT Press.

Laird, J. E., K. Gluck, J. Anderson, K. D. Forbus, O. C. Jenkins, C. Lebiere, D. Salvucci, et al. 2017. "Interactive Task Learning." *IEEE Intelligent Systems* 32(4): 6–21.

Langley, P. (2017, February 4-9,). "Progress and Challenges in Research on Cognitive Architectures." In Proceedings of the Thirty-first AAAI Conference on Artificial Intelligence, San Francisco, USA.

Langley, P., and D. Choi. 2006. "An Unified Cognitive Architecture for Physical Agents." In *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI)*.

Leonetti, M., L. Iocchi, and P. Stone. 2016. "A Synthesis of Automated Planning and Reinforcement Learning for Efficient, Robust Decision-making." *Artificial Intelligence* 241: 103–30.

Lovas, W. 2010. "Refinement Types for Logical Frameworks." PhD thesis, Carnegie Mellon University.

Lyu, D., F. Yang, B. Liu, and S. Gustafson. 2019. "SDRL: Interpretable and Data-efficient Deep Reinforcement Learning Leveraging Symbolic Planning." In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*.

Mathew, M., S. Sidhik, M. Sridharan, M. Azad, A. Hayashi, and J. Wyatt. 2019. "Online Learning of Feed-Forward Models for Task-Space Variable Impedance Control." In *Proceedings of the IEEE-RAS International Conference on Humanoid Robotics*.

McCarthy, J. 1978. "History of Lisp." In *History of Programming Languages*, 173–85. New York City, NY United States: ACM.

McGuinness, D. L., F. Van Harmelen. 2004. Owl Web Ontology Language Overview.

Mericli, C., S. D. Klee, J. Paparian, and M. Veloso. 2014. "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)*.

Milch, B., B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov (2006). "BLOG: Probabilistic Models with Unknown Objects." In *Statistical Relational Learning*. Cambridge, MA United States: MIT Press.

Miller, T. 2019. "Explanations in Artificial Intelligence: Insights from the Social Sciences." *Artificial Intelligence* 267: 1–38.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, et al. 2015. "Human-level Control through Deep Reinforcement Learning." *Nature* 518(7540): 529.

Mombourquette, B., C. Muise, and S. A. McIlraith. 2017. "Logical Filtering and Smoothing: State Estimation in Partially Observable Domains." In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 3613–21.

Mota, T., and M. Sridharan. 2019. "Commonsense Reasoning and Knowledge Acquisition to Guide Deep Learning on Robots." In *Proceedings of the Robotics Science and Systems*, Freiburg, Germany. June 22–6.

Mota, T., and M. Sridharan. 2021. "Answer Me This: Constructing Disambiguation Queries for Explanation Generation in Robotics." In *Proceedings of the IEEE International Conference on Development and Learning (ICDL)*. August 23–6.

Mota, T., M. Sridharan, and A. Leonardis. 2021. "Integrated Commonsense Reasoning and Deep Learning for Transparent Decision Making in Robotics." *Springer Nature Computer Science* 2(242): 1–18.

Nath, A., and P. Domingos. 2009. "A Language for Relational Decision Theory." In Proceedings of the International Workshop on Statistical Relational Learning, Leuven, Belgium. July 2–4.

Lovas, W. and F. Pfenning. 2010. "Refinement Types for Logical Frameworks and Their Interpretation as Proof Irrelevance." *Logical Methods in Computer Science* 6: 1–50.

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

Puterman, M. L. 2014. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ United States: John Wiley & Sons.

Raedt, L. D., and A. Kimmig. 2015. "Probabilistic Logic Programming Concepts." *Machine Learning* 100(1): 5–47.

Richardson, M., and P. Domingos. 2006. "Markov Logic Networks." *Machine Learning* 62(1): 107–36.

Riley, H., and M. Sridharan. 2019. "Integrating Non-monotonic Logical Reasoning and Inductive Learning With Deep Learning for Explainable Visual Question Answering." *Frontiers in Robotics and AI, Special issue on Combining Symbolic Reasoning and Data-Driven Learning for Decision-Making* 6: 20. December.

Rosenthal, S., M. Veloso, and A. K. Dey. 2012. "Is Someone in this Office Available to Help Me?." *Journal of Intelligent & Robotic Systems* 66(1): 205–21.

Samadi, M., T. Kollar, and M. Veloso. 2012. "Using the Web to Interactively Learn to Find Objects." In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*.

Sanner, S. 2010. "Relational Dynamic Influence Diagram Language (RDDL): Language Description."

Sanner, S., and K. Kersting. 2010. "Symbolic Dynamic Programming for First-order POMDPs." In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*.

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

Saribatur, Z., T. Eiter, and P. Schuller. 2021. "Abstraction for Non-ground Answer Set Programs." *Artificial Intelligence* 300: 103563.

Saribatur, Z., V. Patoglu, and E. Erdem. 2019. "Finding Optimal Feasible Global Plans for Multiple Teams of Heterogeneous Robots using Hybrid Reasoning: An Application to Cognitive Factories." *Autonomous Robots* 43(1): 213–38.

Scheutz, M., P. Schermerhorn, J. Kramer, and D. Anderson. 2007. "First Steps Towards Natural Human-Like HRI." *Autonomous Robots* 22(4): 411–23.

Schulman, J., S. Levine, P. Abbeel, M. Jordan, and P. Moritz. 2015. "Trust Region Policy Optimization." In *Proceedings of the International Conference on Machine Learning*, 1889–97.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*.

She, L., and J. Chai. 2017. "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)*, 1634–44.

Sidhik, S., M. Sridharan, and D. Ruiken. 2021. "Towards a Framework for Changing-Contact Manipulation Tasks." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. September 27–October 1.

Sridharan, M., M. Gelfond, S. Zhang, and J. Wyatt. 2019. "Reba: A Refinement-based Architecture for Knowledge Representation and Reasoning in Robotics." *Journal of Artificial Intelligence Research* 65: 87–180.

Sridharan, M., and B. Meadows (2018, December). "Knowledge Representation and Interactive Learning of Domain Knowledge for Human-Robot Collaboration." *Advances in Cognitive Systems* 7: 77–96.

Sridharan, M., and B. Meadows. 2019. "Towards a Theory of Explanations for Human-Robot Collaboration." *Kunstliche Intelligenz* 33(4): 331–42. December.

Sridharan, M., B. Meadows, and R. Gomez. 2017. "What Can I Not Do? Towards an Architecture for Reasoning about and Learning Affordances." In Proceedings of the International Conference on Automated Planning and Scheduling, Pittsburgh, USA. June 18–23.

Sutton, R. S., and A. G. Barto. 2018. *Reinforcement Learning: An Introduction*. Cambridge, MA United States: MIT Press.

Thomason, J., S. Zhang, R. Mooney, and P. Stone. 2015. "Learning to Interpret Natural Language Commands through Human–Robot Dialog." In *Proceedings of the 24th International Conference on Artificial Intelligence*, 1923–9.

Towell, G. G., and J. W. Shavlik. 1994. "Knowledge-based Artificial Neural Networks." *Artificial Intelligence* 70(1-2): 119–65.

Wang, Y., S. Zhang, and J. Lee. 2019. "Bridging Commonsense Reasoning and Probabilistic Planning Via a Probabilistic Action Language." *Theory and Practice of Logic Programming (TPLP)* 19(5-6): 1090–106.

Yang, F., D. Lyu, B. Liu, and S. Gustafson. 2018. "PEORL: Integrating Symbolic Planning and Hierarchical Reinforcement Learning for Robust Decision-making." In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*.

Younes, H. L., and M. L. Littman. 2004. "PPDDL1.0: The Language for the Probabilistic Part of IPC-4." In *Proceedings of the 2004 International Planning Competition*.

Zhang, H-D., C. Zhen-Hao, C. Jun-Yang, Z. Yi, L. De-Fu, W. Kai-Shun and L. Fang-Zhen. 2022. "Dynamic decision making framework based on explicit knowledge reasoning and deep reinforcement learning." *Journal of Software*.

Zhang, S., P. Khandelwal, and P. Stone. 2017. "Dynamically Constructed (PO)MDPs for Adaptive Robot Planning." In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*.

Zhang, S., M. Sridharan, and J. L. Wyatt. 2015. "Mixed Logical Inference and Probabilistic Planning for Robots in Unreliable Worlds." *IEEE Transactions on Robotics* 31(3): 699–713.

Zhang, S., and P. Stone. 2015. "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)*.

Zhu, W. 2012. *Plog: Its Algorithms and Applications*. PhD thesis, Texas Tech University.

AUTHOR BIOGRAPHIES

Shiqi Zhang is an Assistant Professor of Computer Science, at the State University of New York (SUNY) at Binghamton (USA). He was a Postdoctoral Fellow at The University of Texas at Austin (USA) from 2014 to 2016, and received his Ph.D. in Computer Science (2013) from Texas Tech University (USA). Before that, he received his Master's and B.S. from Harbin Institute of Technology in China. Dr. Zhang's research lies at the intersection of artificial intelligence and robotics.

Mohan Sridharan is a Reader in Cognitive Robot Systems in the School of Computer Science at the University of Birmingham (UK). Prior to his current appointment, he held faculty positions at Texas Tech University (USA) and The University of Auckland (NZ). He received his Ph.D. in Electrical and Computer

Engineering from The University of Texas at Austin (USA). Dr. Sridharan's research interests include cognitive systems, knowledge representation and reasoning, machine learning, and computational vision in the context of human–robot and human–agent collaboration.

How to cite this article: Zhang, S., and M. Sridharan. 2022. A survey of knowledge-based sequential decision-making under uncertainty. *AI Magazine* 43: 249–66.
https://doi.org/10.1002/aaai.12053