Knowledge-based Sequential Decision-Making under Uncertainty

Shiqi Zhang (SUNY Binghamton, USA)
Mohan Sridharan (University of Birmingham, UK)

szhang@cs.binghamton.edu; m.sridharan@bham.ac.uk
Tutorial Objectives

- Motivate knowledge-based sequential decision making under uncertainty
- Describe related concepts in knowledge representation, reasoning and learning with simple robotics examples
- Draw on own work and work by others to describe architectures that illustrate knowledge-based sequential decision making under uncertainty
- Explore interplay between knowledge representation, reasoning and learning with architecture examples
- Will not discuss specific “solvers” for logical or probabilistic reasoning; the architectures described will use such solvers
Tutorial Outline

● Introduction

● Basics:
  ○ Knowledge representation: declarative, probabilistic, hybrid
  ○ Reasoning: logic-based, MDP, POMDP
  ○ Learning: reinforcement

● Example architectures:
  ○ Knowledge guides reasoning
  ○ Knowledge guides learning
  ○ Learning for knowledge revision

● Discussion

*Shiqi Zhang* (SUNY Binghamton) & Mohan Sridharan (U. of Birmingham)
Knowledge-based Sequential Decision-making under Uncertainty

- Sequential decision-making (SDM):
  - More than one action often required to complete complex tasks
  - Subsequent actions often depend on the effects of actions that precede them

- Reasoning (planning, diagnostics) under uncertainty:
  - Actions in complex, practical domains are non-deterministic
  - Local, unreliable observations; partial observability

- Knowledge-based:
  - Considerable commonsense knowledge available in practical applications
  - Reasoning with this knowledge can improve decision making and guide learning
Knowledge Representation, Reasoning and Learning

● How is knowledge represented?
  ○ Knowledge representation (KR) is a fundamental research area in AI
  ○ Representations include logic, probability, graphs, etc

● How to reason with knowledge?
  ○ Different reasoning mechanisms based on the underlying representation

  Query → KRR → Conclusions

● Why learning?
  ○ Reasoning with incomplete knowledge results in incorrect or suboptimal outcomes
  ○ Exploit ability to observe domain and action outcomes, learn from trial and error

● Representation, reasoning and learning are inter-dependent!
Overview of Knowledge-based SDM

How to use knowledge to help agent better select actions?

Declarative knowledge

How to augment knowledge with SDM experience?

Probabilistic Planning (MDPs, POMDPs, etc)

Reinforcement Learning (model-free, model-based, etc)

Sequential Decision-Making (SDM) under Uncertainty

State, reward

Action
SDM Applications

- Robotics; used often in tutorial
- Finance
- Urban planning
- Healthcare

- Games
- Transportation
- E-commerce
- ... and many more ...

Common Applications

- autonomous driving
- business operations
- robotics
- language & dialogue (structured prediction)
- finance

Image from Sergey Levine
Motivating Example

Consider a robot assisting humans in an indoor domain.

- The robot has to find and move objects to locations or people.
- Has some prior knowledge of locations, objects and object properties.
- Humans provide limited feedback.
- Noisy sensing and actuation.
Tutorial Outline

- Introduction
- **Basics:**
  - Knowledge representation: declarative, probabilistic, hybrid
  - Reasoning: logic-based, MDP, POMDP
  - Learning: reinforcement
- **Example architectures:**
  - Knowledge guides reasoning
  - Knowledge guides learning
  - Learning for knowledge revision
- Discussion

*Shiqi Zhang* (SUNY Binghamton) & Mohan Sridharan (U. of Birmingham)
SDM paradigms: Broad Classification

- **Logic-based commonsense reasoning**
  - Logics to represent uncertainty, *commonsense knowledge and theories of action*
  - Challenges: comprehensive domain knowledge, quantitative models of uncertainty

- **Probabilistic reasoning** or **decision-theoretic planning**
  - Compute an action policy *when domain model is known and probabilistic*
  - Challenges: long planning horizons, large state and action spaces

- **Reinforcement learning (RL)**
  - Learn an action policy through trial and error *when domain model is unknown*
  - Challenges: exploration/exploitation tradeoff, credit assignment, structured knowledge
Logic-based Knowledge Representation

- Many different logics: first order, non-monotonic, temporal
- We discuss non-monotonic logics; often Prolog-style statements
  
  \[\text{Head :- Body.}\]
  
  "Head is true if Body is true"

- Particular example: Answer Set Prolog [Gelfond, Kahl 2014]

- Action language: formal model of part of natural language used to describe transition diagrams [Gelfond, Lifschitz 1998]; many options, e.g., AL, B, C etc

- In AL: hierarchy of basic sorts, statics, fluents, actions

- Statements: causal law, state constraint, executability condition

- Statements of AL provide system description: signature and axioms.
Declarative Knowledge: Answer Set Prolog

● **Signature:**
  ○ Basic sorts: *robot, place, object, cup, book, printer*
  ○ Statics: *next_to(place, place), obj_weight(O, weight)*
  ○ Fluents: *loc(robot) = place, in_hand(robot, object)*
  ○ Actions: *move(robot, place), pickup(robot, object), serve(robot, object, person)*

● **Axioms:**
  ○ Causal laws:
    move(rob, Pl) **causes** loc(rob) = Pl
    pickup(rob, O) **causes** in_hand(rob, O)
  ○ State constraints:
    loc(O) = Pl **if** loc(rob) = Pl, in_hand(rob, O)
  ○ Executability conditions:
    **impossible** pickup(rob, O) **if** loc(rob) = Pl1, loc(O) = Pl2, Pl1 != Pl2
    **impossible** pickup(rob, O) **if** obj_weight(O, heavy)
Declarative Knowledge: Answer Set Prolog

- Appealing properties of ASP:
  - Default negation and epistemic disjunction; things can be true, false, and unknown
    - $p$: $p$ is believed to be false
    - not $p$: $p$ is not believed to be true
  - Only believe what you are forced to believe!
  - Represent recursive definitions, defaults, causal relations, self-reference, and language constructs occurring in non-mathematical domains
  - Unlike classical first order logic, supports non-monotonic logical reasoning, i.e., revise previously held conclusions.

- Domain representation: system description $D$ and history $H$.
- History contains records of the form:
  - obs(fluent, boolean, timestep)
  - hpd(action, timestep)

- Translate $D$ and $H$ to ASP program (automatic tools) for reasoning.
Probabilistic Knowledge Representation

- Many representations possible; we focus on *Probabilistic Graphical Models* (PGMs) that probabilistically model state transitions, causal relationships etc.

- PGMs use a **graph** to express conditional independence between random variables.

- We are particularly interested in directed acyclic PGMs (also called *Bayesian networks*).
Probabilistic Knowledge Representation

- Many representations possible; we focus on **Probabilistic Graphical Models** (PGMs) that probabilistically model state transitions, causal relationships etc.

- **Joint probability as product of conditional probabilities and marginals:**
  \[ P(C, S, R, W) = P(W | S, R) \cdot P(S | C) \cdot P(R | C) \cdot P(C) \]

- We only discuss the PGMs:
  - Learned by agent/robot from environment; or
  - Constructed using human input or feedback

Human, world, or both
Hybrid Knowledge Representation

- Combine logics and probabilities
- Literals hold true with some probability

Left: an example of MLN

Two constants: **Anna** (A) and **Bob** (B)

Compute the probability of:

- Anna and Bob being friends given their smoking habits
- Bob having cancer given his friendship with Anna and the likelihood of Anna having cancer
Representation of Probabilistic Planning Domains

- **PDDL** is developed for and maintained by the International Planning Competition (IPC) community [McDermott, Ghallab, et al. 1998], and is (arguably) the most popular declarative language for classical planning.

- **PPDDL** developed for describing MDP settings in 2004.

- In 2011, **Relational Dynamic Influence Diagram Language (RDDL)** developed for better expressiveness (c.f., PPDDL).

- **pBC+** developed for probabilistic reasoning about transition systems [Lee, Wang 2018].

These and other similar action languages are limited in terms of representing and reasoning with different descriptions of knowledge and uncertainty.
Tutorial Outline

- Introduction
- **Basics:**
  - Knowledge representation: declarative, probabilistic, hybrid
  - **Reasoning:** logic-based, MDP, POMDP
  - Learning: reinforcement
- **Example architectures:**
  - Knowledge guides reasoning
  - Knowledge guides learning
  - Learning for knowledge revision
- Discussion

*Shiqi Zhang* (SUNY Binghamton) & Mohan Sridharan (U. of Birmingham)
Logics for Reasoning

- Reasoning includes planning, diagnostics and inference.
- Strategy depends on representation; many solvers have been developed.
- Map reasoning task to:
  - Resolution and theorem proving, e.g., with First Order Logic.
  - Constraint satisfaction problem (CSP).
  - Satisfiability (SAT) problem, e.g., with ASP.
- We do not focus on solvers in this tutorial; instead, we explore how they can be used to formulate and solve problems.
- Let us explore how reasoning is accomplished using CR-Prolog, a variant of ASP with consistency-restoring (CR) rules [Balduccini, Gelfond, 2003].
CR-Prolog Program

- Convert D and H as program: \( \neg (D, H) \)
- Signature and axioms of D, inertia axioms:
  \[
  \text{holds}(F, I+1) :- \text{holds}(F, I), \text{not} \ -\text{holds}(F, I+1) \\
  -\text{holds}(F, I+1) :- -\text{holds}(F, I), \text{not} \ \text{holds}(F, I+1)
  \]
- Reality checks, closed world assumptions for defined fluents and actions
  \[
  :- \text{holds}(F, I), \ \text{obs}(F, \text{false}, I) \\
  :- -\text{holds}(F, I), \ \text{obs}(F, \text{true}, I)
  \]
- Observations, actions, defaults from H, e.g., initial state default + CR rule:
  \[
  \text{holds}(\text{loc}(X) = \text{library}, 0) :- \ \text{textbook}(X), \text{not} \ -\text{holds}(\text{loc}(X) = \text{library}, 0) \\
  -\text{holds}(\text{loc}(X) = \text{library}, 0) :\pm \ \text{textbook}(X), \text{not} \ -\text{holds}(\text{loc}(X) = \text{library}, 0)
  \]
- Planning and diagnosis reduced to computing answer sets of program.
CR-Prolog Planning Example

- Goal: \text{loc(book1, office2)}, \text{-in_hand(rob, book1)}

- Given: \text{textbook(book1), loc(rob) = kitchen, ..., next_to(kitchen, office2)}, \text{next_to(library, kitchen)}, ... 

- Based on default knowledge: \text{move(rob, library), pickup(rob, book1), move(rob, kitchen), move(rob, office2), putdown(rob, book1)}
Challenges in using Logics for Reasoning

- Modeling and reasoning with sensing and actuation uncertainty.
- Domain knowledge often incomplete and may change.
- Fine-grained reasoning necessary (e.g., grasping) but computationally expensive.

*Will return to these later*
Probabilistic Reasoning: Bayes Rule and Filter

- Joint and conditional probability of random variables: $P(A, B)$, $P(A | B)$
- Basic Bayes rule: $P(A, B) = P(A | B) P(B) = P(B | A) P(A)$
  $$P(A | B) = \frac{P(B | A) P(A)}{P(B)}$$
- Bayes filter for state estimation (prediction and correction):
  - **Bayes filter:**
    $$\forall x_t : \text{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) \text{bel}(x_{t-1}) \, dx_{t-1}$$
    $$\text{bel}(x_t) = \eta \, p(z_t | x_t) \, \text{bel}(x_t)$$
  - **Discrete Bayes filter:**
    $$\forall k : \overline{p}_{k,j} = \sum_i p(X_t = x_k | u_t, X_{t-1} = x_i) \, p_{i,t-1}$$
    $$p_{k,j} = \eta \, p(z_t | X_t = x_k) \, \overline{p}_{k,j}$$
- $X$ (or $S$) = state, $U$ (or $A$) = action, $Z$ = observation (i.e., measurement)
- **Bayes filter is the basis of most probabilistic reasoning systems**
Probabilistic Reasoning: Markov Decision Process (MDP)

- **Markov property** is assumed to hold for MDP (and later RL)
  - First-order: given current state, next state is conditionally independent of previous states
  - Simplifies computation of policies for complex real-world problems

- MDP is an *SDM framework* under the Markov assumption [Puterman 2014]

- An MDP is a 4-tuple <S, A, T, R>
  - States, Actions, Transitions, and Rewards
  - T: S × A × S’ ↦ [0, 1]
  - R: S × A × S’ ↦ ℙ

- Solving an MDP produces a policy:
  - π: S ↦ A
Probabilistic Reasoning: Partially Observable MDPs (POMDPs) [Kaelbling, Littman, Cassandra. 1998]

- Partial observability and non-determinism
- POMDP tuple \( <S, A, Z, T, O, R> \):
  - \( Z \): set of observations
  - \( O \): observation function: \( P(z \in Z | s \in S, a \in A) \)
    \[
    O: S \times A \times Z \mapsto [0, 1]
    \]
- Maintain \textit{belief state} (or belief), a probability distribution over states, using observations
- Solving a POMDP produces a policy mapping beliefs to actions.
  \[
  \pi: B \mapsto A
  \]
MDPs and POMDPs

Observability

Partial → POMDP → Belief state → MDP

Probabilistic planning over a long, unspecified horizon…
MDPs and POMDPs as DBN

- MDPs and POMDPs are essentially Dynamic Bayesian Networks (DBNs)
MDPs and POMDPs as DBN

- MDPs and POMDPs are essentially Dynamic Bayesian Networks (DBNs)

\[ b'(s') = \eta O(o \mid s', a) \sum_{s' \in S} T(s' \mid s, a)b(s) \]

where \( \eta = 1 / \Pr(o \mid b, a) \) is a normalizing constant
MDPs and POMDPs Algorithms

- Many MDP and POMDP algorithms:
  - Bellman equation, Value Iteration (VI); classical solvers
  - Monte Carlo tree search (MCTS), point-based (approximate) methods [Shani, Pineau, Kaplow 2013]
  - And many more...

![Diagram showing the interaction between world model, goal, MDP/POMDP algorithms, policy, and interact]

World model

Goal

MDP/POMDP algorithms

Policy

Interact
Challenges in MDPs and POMDPs Algorithms

- MDP/POMDP algorithms computationally expensive for large complex domains.
- Policy often assumed to be stationary.
- *By themselves, not well-suited for commonsense reasoning.*

*Will return to these later*
Tutorial Outline

● Introduction

● Basics:
  ○ Knowledge representation: declarative, probabilistic, hybrid
  ○ Reasoning: logic-based, MDP, POMDP
  ○ Learning: reinforcement

● Example architectures:
  ○ Knowledge guides reasoning
  ○ Knowledge guides learning
  ○ Learning for knowledge revision

● Discussion

Shiqi Zhang (SUNY Binghamton) & Mohan Sridharan (U. of Birmingham)
Learning for Decision Making

- Domain knowledge incomplete and can become inconsistent

- Decisions made can be incorrect or sub-optimal:
  - Moving on newly polished surface
  - Inaccurate model of sensors or domain objects

- Different ways to learn knowledge and use for decision making:
  - Supervised learning from labeled training samples
  - Unsupervised learning
  - ...
  - Learning through trial and error

- We focus on **Reinforcement Learning** for decision making
Reinforcement learning (RL)

- Basic idea:
  - State fully observable, actions non-deterministic
  - Attempt different actions, receive feedback in the form of rewards
  - Agent learns to act so as to maximize expected cumulative rewards

- Still have an MDP:
  - Set of states and actions.
  - Learn policy $\pi: S \rightarrow A$
  - *No knowledge of domain models* ($T$, $R$); *trial and error approach*
Reinforcement learning (RL) [Sutton 2018]

- Different “threads” of RL
  - Trial and error approach; origins in psychology.
  - Dynamic programming approach for stochastic control problems
  - **Temporal difference methods**

- Challenges:
  - Exploration/exploitation, generalization.
  - Credit assignment
  - Model design, reward specification
  - Delayed consequences

![Image from David Silver]
**RL Algorithms Taxonomy**

- **Model-based:**
  - Compute model parameters $T$, $R$; solve MDP for value function $V(s)$ or $Q$-value function $Q(s,a)$

- **Model-free:**
  - Directly compute $V(s)$ or $Q(s, a)$ from samples $(s, a, r, s')$

- **Policy-based:**
  - Compute state-action mapping

- **Advanced algorithms:**
  - State-action abstractions, function approximation through deep learning
Tutorial Outline

- Introduction
- Basics:
  - Knowledge representation: declarative, probabilistic, hybrid
  - Reasoning: logic-based, MDP, POMDP
  - Learning: reinforcement
- Example architectures:
  - Knowledge guides reasoning
  - Knowledge guides learning
  - Learning for knowledge revision
- Discussion

Shiqi Zhang (SUNY Binghamton) & Mohan Sridharan (U. of Birmingham)
Logical Inference Guides Probabilistic Planning

- **Logical reasoning** to compute **informative priors** for planning with partial observability

- Components:
  - ASP-based inference with commonsense knowledge sets probabilistic priors
  - Probabilistic planning with these priors using hierarchical POMDPs
  - Reason about domain-level priors

Zhang, Sridharan, Wyatt. 2015
Logical Inference Guides Probabilistic Planning

- Early work on commonsense (logical) reasoning guiding probabilistic state estimation
- Computing *probabilistic* priors from *logical* knowledge uses postulates (e.g., objects from a class are often co-located) and psychophysics
- Knowledge from similar domains provide priors for early termination

Zhang, Sridharan, Wyatt. 2015
Logical-Probabilistic Reasoning about Belief State

- Algorithm **CORPP**: (logical-probabilistic) commonsense reasoning and probabilistic planning
- Logical reasoning for filtering out irrelevant states
- Probabilistic reasoning for associating probability with each state

Zhang, Stone. 2015
Logical-Probabilistic Reasoning about Belief State

- CORPP was used with *spoken dialog system* for sequential decision-making

**Robot needs to identify \(<Coffee, Office 1, Bob>\), through spoken dialog**

<table>
<thead>
<tr>
<th>Time:</th>
<th>9:00am</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rooms:</td>
<td>Office 1, Office 2, ...</td>
</tr>
<tr>
<td>Persons:</td>
<td>Alice, Bob, Carol, ...</td>
</tr>
<tr>
<td>Items:</td>
<td>Coffee, Sandwich, ...</td>
</tr>
</tbody>
</table>

- Dialog manager (a planner) maintains a *belief distribution* over possible service requests
- Reasoning is for initializing belief distributions with *informative priors*
Dynamically Factored Belief State

- Robot receives both *sensory information* and human-provided *declarative knowledge*
- How to accurately incorporate the (noisy, relational) information to achieve goals in POMDP setting?

![Diagram showing POMDP with observations drawn from two sources](image-url)
Dynamically Factored Belief State

- **Idea:**
  - Join factors when *their variables are correlated* through observational information
  - Separate factors when uncorrelated

Robotic cooking domains:

- Involving both locations and ingredients
- Robot is tasked with gathering ingredients and using them to cook a meal

Chitnis, Kaelbling, and Lozano-Perez. 2018
Knowledge-based belief estimation

Probabilistic planning over a long, unspecified horizon…

Zhang, Stone. 2015
Zhang, Sridharan, Wyatt. 2015
Chitnis, Kaelbling, and Lozano-Perez. 2018
Logical-Probabilistic Reasoning about Dynamics

- Interleaved CORPP (iCORPP)
  - Reasons about world dynamics with logical-probabilistic knowledge
  - Dynamically constructs transition systems (MDP/POMDPs) for adaptive planning

Transition probability of a navigation action depends on many factors: weather, near-window status, time, human positions, etc

*It’s infeasible to consider all in the (PO)MDPs*
iCORPP *dynamically builds (PO)MDPs* by reasoning with knowledge about world dynamics
Knowledge-based Dynamics Estimation

Belief state

MDP

Partial POMDP

Full MDP

Observability

Probabilistic planning over a long, unspecified horizon…

Zhang, Khandelwal, Stone. 2017
Switching Planner

- Switches between classical planner and probabilistic planner depending on level of uncertainty \([\text{Hanheide et al., 2017}]\)

- Classical planner: **Continual Planning** \([\text{Brenner, Nebel, 2009}]\)
  - Interleaves planning, plan execution and plan monitoring
  - Actions assert that preconditions will be met when that point in plan execution reached
  - Replanning triggered if preconditions are not met during execution or are met earlier

- Probabilistic planning computes actions executed in the physical world.
Switching Planner

- Overall architecture:
  - Three-layered organization of knowledge (instance, default, diagnostic)
  - Three-layered architecture (competence, belief, deliberative)
  - Combines first-order logic and probabilistic reasoning for planning

- **Decision-Theoretic PDDL** (DTPDDL) used for representing both action preconditions and effects, as well as probabilistic transitions

- **Weak coupling** (transfer of information) between the two planning systems
REBA: Refinement-based KRR

- Represent and reason with tightly-coupled transition diagrams at two different resolutions [Sridharan et al., 2018, 2019]

- For any given goal, non-monotonic logical reasoning with commonsense knowledge at coarse-resolution provides sequence of abstract actions

- Each abstract transition implemented as sequence of fine-resolution concrete actions; automatically zoom to and reason probabilistically with part of fine-resolution diagram relevant to coarse-resolution transition

- Result of executing fine-resolution action updates coarse-resolution history for subsequent reasoning

- We use CR-Prolog for logical reasoning, hierarchical POMDPs for probabilistic reasoning.
REBA: Refinement-based KRR (Example)

- Examine the transition of a robot moving between two rooms at coarse-resolution and fine-resolution

Coarse resolution

\[ \text{loc}(\text{rob1}) = \text{office} \quad \text{loc}(\text{rob1}) = \text{kitchen} \]

move(rob1, kitchen)

Fine resolution

r1 (office)

\[ \text{loc}(\text{rob1}) = c1 \]

move(rob1, c2)

\[ \text{loc}(\text{rob1}) = c2 \]

r2 (kitchen)

\[ \text{loc}(\text{rob1}) = c5 \]

move(rob1, c6)

\[ \text{loc}(\text{rob1}) = c6 \]
REBA: Refinement-based KRR (Example)

- **Goal:** $\text{loc}(B) = \text{kitchen}$, $-\text{in\_hand}(\text{rob}, B)$, $\text{box}(B)$
- **Initial:** $\text{loc}(\text{rob}) = \text{office}$, $\text{obj\_weight}(\text{box1}, \text{heavy})$, $\text{arm}(\text{rob}, \text{pneumatic})$
- Based on default: $\text{loc}(\text{box1}) = \text{office}$
- One **coarse-resolution plan** from ASP-based inference:
  \[
  \text{move}(\text{rob}, \text{office}), \text{pickup}(\text{rob}, \text{box1}), \\
  \text{move}(\text{rob}, \text{kitchen}), \text{putdown}(\text{rob}, \text{box1})
  \]
- Assume $\text{rob}$ is in $\text{office}$. implement $\text{pickup}(\text{rob}, \text{box1})$; find+pickup $\text{box1}$
- Relevant literals: $\text{loc}(\text{rob}) = C1$, $\text{loc}(\text{box1}) = C2$ where $C1, C2 \in \text{office}$
- Possible **fine-resolution action sequence** (executed probabilistically):
  \[
  \ldots \\
  \text{mov}(\text{rob}, c3) \\
  \text{test}(\text{rob}, \text{loc}(\text{box1}), c3) \quad \% \text{box1 observed!} \\
  \text{pickup}(\text{rob}, \text{box1})
  \]
- Subsequent plan steps succeed
REBA: Refinement-based KRR

● Key contributions:
  ○ *Tight coupling* between transition diagrams
  ○ *Theory of observations*; formal definitions of *refinement* and *zooming*
  ○ *Automatic construction* of data structures for probabilistic reasoning
  ○ General methodology for design of software for robots; Dijkstra’s *step-wise refinement*
  ○ Combine strengths of declarative programming, probabilistic reasoning

● Advantages:
  ○ *Simplifies and speeds up design; increases confidence* in correctness of robot’s behavior
  ○ *Separation of concerns*; reuse of representations on other robots and domains
  ○ *Single framework for planning, diagnostics, inference*, trade-off accuracy and efficiency
  ○ Significant improvement in reliability and efficiency; *scales to complex domains*

Sridharan, Gelfond, Zhang, Wyatt. 2018
## Comparative Summary of Architectures

<table>
<thead>
<tr>
<th>Algorithm name</th>
<th>Logical knowledge</th>
<th>Probabilistic knowledge</th>
<th>Tight Coupling</th>
<th>Reason about Dynamics</th>
<th>Interleaved reasoning &amp; planning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switching planner (2017)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>ASP-POMDP (2015)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CORPP (2015)</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>iCORPP (2017)</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Dynamic Factorization (2018)</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>REBA (2018)</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Here “knowledge” refers only to declarative knowledge
- Tight coupling refers to the transfer of all (and only) the relevant information between the logical and probabilistic reasoning components
Tutorial Outline

● Introduction

● Basics:
  ○ Knowledge representation: declarative, probabilistic, hybrid
  ○ Reasoning: logic-based, MDP, POMDP
  ○ Learning: reinforcement

● Example architectures:
  ○ Knowledge guides reasoning
  ○ Knowledge guides learning
  ○ Learning for knowledge revision

● Discussion

Shiqi Zhang (SUNY Binghamton) & Mohan Sridharan (U. of Birmingham)
Domain Approximation for Reinforcement LearnING (DARLING)

- **Reasoner** provides a rational way to constrain the exploration, while **RL** eases the requirements on the model accuracy.

- DARLING is composed of three steps:
  1. **Plan generation**: find all **reasonable plans** (cost < threshold)
  2. **Plan filtering**: exclude “certainly-suboptimal” plans, e.g., those with redundant actions, and generate **partial policy**
  3. **Execution and learning**: try only actions that are returned by the partial policy in exploration

Door status unknown initially: door being open with increasing probability

Domain map, and states traversed during the first and last 50 episodes by the RL (Sarsa) and PRL (knowledge-based RL) agents
Domain Approximation for Reinforcement Learning (DARLING)

DARLING uses declarative action knowledge to guide robot exploration in reinforcement learning -- robot only tries the *reasonable* actions.
Symbolic Deep Reinforcement Learning (SDRL)

- **Symbolic planner**: action knowledge for long-term planning
- **Controller**: DRL for learning for each subtask based on intrinsic rewards;
- **Meta-controller**: learning extrinsic rewards from the controller’s performance, and propose new intrinsic goals to the planner

![Diagram of Symbolic Deep Reinforcement Learning (SDRL)]
Symbolic Deep Reinforcement Learning (SDRL)

- hDQN cannot reach the “400” score with 2.5M samples
- The variance of SDRL is smaller than the hDQN’s
- Symbolic planner guides primitive sub-policy learning

Montezuma’s Revenge, & the optimal policy
Symbolic Deep Reinforcement Learning (SDRL)

- SDRL uses an RL agent to interact with the “real world”, and reports to the task level agent (task planner) with abstraction.

- The refinement idea is similar to the REBA architecture [Sridharan et al 2018, 2019], while SDRL learns from the task-completion experience.

- SDRL is a follow-up work of PEORL [Yang, Lyu, Liu, Gustafson, 2018] where perception of RL is symbolic.
KRR-RL: integrated logical-probabilistic KRR and model-based RL

- **Logical-probabilistic KRR** allows:
  - Human (logical) knowledge used to specify transition dependency
  - Model-based RL (R-Max) for filling in transition probabilities

KRR-RL agent learns domain dynamics from “small” tasks to get prepared to accomplish “large” tasks.
KRR-RL: integrated logical-probabilistic KRR and model-based RL

A delivery task requires both dialog and navigation actions

- In spare time, agent learns from navigation tasks to prepare for upcoming delivery tasks
- Robot is more cautious on delivery tasks that require significant navigation efforts

Lu, Zhang, Stone, Chen. 2018
KRR-RL: integrated logical-probabilistic KRR and model-based RL

KRR-RL Assumptions:

● Domain experts (human) are good at providing *qualitative* actions preconditions and effects

● Model-based RL algorithms do well in learning *quantitative* uncertainty of action knowledge
TMP-RL: Integrated Task-Motion Planning and RL

- Task and motion planning (TMP) algorithms generate plans at both symbolic and continuous spaces
  - TMP solutions are sensitive to unexpected domain uncertainty and changes

- TMP-RL features two nested planning-learning loops
  - In the inner TMP loop, the robot generates a low-cost, feasible task-motion plan
  - In the outer loop, the plan is executed, and the robot learns from the execution experience via model-free RL
TMP-RL: Integrated Task-Motion Planning and RL

- TMP-RL performs the best in learning rate
- TMP and TMP-RL have smaller variance during execution
- TMP does not improve over time
## Summary of knowledge-based RL

<table>
<thead>
<tr>
<th>Algorithm name</th>
<th>Prob. KR</th>
<th>Different resolutions</th>
<th>Lookahead in KR</th>
<th>Representation learning</th>
<th>Model based RL</th>
<th>Motion planning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DARLING (2016)</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SDRL (2018)</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>KRR-RL (2018)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>PEORL (2018)</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TMP-RL (2018)</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

There is also research on integrating cognitive architectures with reinforcement learning, such as SHARSHA (2001) and Soar-RL (2004). These (and other such) cognitive architectures support learning and inference.
Tutorial Outline

● Introduction

● Basics:
  ○ Knowledge representation: declarative, probabilistic, hybrid
  ○ Reasoning: logic-based, MDP, POMDP
  ○ Learning: reinforcement

● Example architectures:
  ○ Knowledge guides reasoning
  ○ Knowledge guides learning
  ○ Learning for knowledge revision

● Discussion

Shiqi Zhang (SUNY Binghamton) & Mohan Sridharan (U. of Birmingham)
Learning for Knowledge Revision

- Many approaches possible for revising domain knowledge:
  - Learning action models from observed effects [Gil, 1994]
  - Searching joint space of hypotheses and observations [Simon, Lea, 1974]

- Our focus on declarative knowledge:
  - Inductive learning of causal laws [Otero, 2003]
  - Expand theory of actions, revise ASP system descriptions [Balduccini, 2007; Law et al., 2018]
  - Process perceptual input to learn in cognitive architecture [Laird, 2012]

- Interactive task learning [Chai et al., 2018; Laird et al., 2017]:
  - Labeled examples or reinforcement; Relational RL [Driessens, Ramon, 2003]
  - Learning task knowledge using RRL [Block, Laird, 2017]

- Challenges:
  - Generalization, e.g., of equivalent axioms with redundant parts
  - Actions with delayed effects
  - Observations from active exploration and reactive action execution
Relational Reinforcement Learning

- Combines RL with relational/inductive learning, e.g., Q-RRL algorithm
- Relational representation of states, actions

- Typically uses *logical decision trees*:
  - Learn relationally equivalent states and actions
  - Each example is a relational database, e.g., state description in planning task
  - First-order logic instead of attribute-value representations
  - Prolog-style queries as tests in internal nodes; *binary decision trees (BDT)*

- Declarative bias for learning relational representations of policies

- **Challenges:**
  - RRL typically for particular planning task (e.g., stack blocks), *difficult to learn generic knowledge across tasks (and MDPs)*
  - Computationally expensive in most practical robotics domains
REBA-Interactive Learning

- Combines declarative programming, probabilistic reasoning and relational reinforcement learning [Sridharan, Meadows, 2017, 2018].

- Learn parts of system description (represented as CR-Prolog programs):
  - Action descriptions (i.e., actions, preconditions, effects), action capabilities (affordances)
  - Axioms including causal laws, executability conditions
REBA-Interactive Learning

● Non-monotonic logical reasoning (with or without probabilistic reasoning) used for planning and diagnostics (as in REBA).

● Interactive learning:
  ○ Verbal input to learn action relations and causal laws;
  ○ Active exploration (RRL) of action preconditions and effects;
  ○ Reactive exploration (RRL) of unexpected action outcomes

● **ASP-based reasoning guides learning:**
  ○ Determines transitions to explore further
  ○ Selects and defines relevant MDPs for RRL (active/reactive exploration)

● Learned domain knowledge used for subsequent reasoning

● **Tight coupling:** bidirectional flow of control and relevant information between reasoning and learning
Our Binary Decision Tree

- Generalizes over MDPs; policy for subsequent Q-learning
- Computationally efficient, more reliable, *scales better*
- Nodes: test of domain literals
- Path from root to leaf: partial state-action pair
- Expansion at leaf if adding a test reduces Q-value variance
- *Generates candidate axioms*
Learning for Knowledge Revision (Example)

- **Goal:**
  \( \text{loc}(C) = \text{office}, \text{-in_hand}(\text{rob, } C) \), \( \text{cup}(C) \)

- **Initial:**
  \( \text{loc}(\text{rob}) = \text{office} \),
  \( \text{objweight}(\text{cup1}, \text{light}) \), \( \text{obj_surface}(\text{cup1}, \text{brittle}) \)

- Based on default:
  \( \text{loc}(\text{cup1}) = \text{kitchen} \)

- One **coarse-resolution plan** from ASP-based inference:
  \( \text{move(rob, kitchen)} \), \( \text{pickup(rob, cup1)} \),
  \( \text{move(rob, office)} \), \( \text{putdown(rob, cup1)} \)

- Assume *rob* moves successfully to the *kitchen*.

- Next action to implement: \( \text{pickup(rob, cup1)} \); to find+pickup *cup1*
Learning for Knowledge Revision (Example)

- Relevant literals: $\text{loc(rob)} = C_1, \text{loc(cup1)} = C_2$ where $C_1, C_2$ can be any cell in kitchen
- Possible fine-resolution action sequence (executed probabilistically):
  
  ```
  ... 
  mov(rob, c3) 
  test(rob, loc(cup1), c3) % cup1 observed! 
  pickup(rob, cup1) 
  ... 
  ```

- Robot moves to office and puts cup down; cup is then observed to be broken:
  
  $\text{obs(obj_status(cup1, damaged), true, 4)}$

- This unexpected outcome triggers RRL to learn previously unknown generic axiom:
  
  $\text{putdown(rob, C) causes obj_status(C, damaged) if obj_surface(C, brittle)}$
Tutorial Outline

● Introduction
● Basics:
  ○ Knowledge representation: declarative, probabilistic, hybrid
  ○ Reasoning: logic-based, MDP, POMDP
  ○ Learning: reinforcement
● Example architectures:
  ○ Knowledge guides reasoning
  ○ Knowledge guides learning
  ○ Learning for knowledge revision
● Discussion

Shiqi Zhang (SUNY Binghamton) & Mohan Sridharan (U. of Birmingham)
Discussion

- Key capabilities supported by *knowledge-based SDM under uncertainty*:
  - Non-deterministic action outcomes, partial observability
  - Reasoning with (incomplete) declarative knowledge
  - Efficient learning from interaction experience

- Important *challenges* to be addressed by future work:
  - *Representation for KRR*: logics, probabilistic, hybrid? Integration takes considerable effort if different components have different representations
  - *Benchmark problems* and algorithms; comparing and evaluating architectures is difficult
  - *Formal analysis* for trustworthy behavior: completeness and soundness guarantees
  - *Scaling* to large knowledge bases/ontologies and complex relationships
  - *Explainable decision making*

*Shiqi Zhang* (SUNY Binghamton) & Mohan Sridharan (U. of Birmingham)


References

References

References


Questions and comments