Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Symbolic Reinforcement Learning using Inductive Logic Programming

Author:
Kiyohito Kunii

Supervisor:
Professor Alessandra Russo

Submitted in partial fulfillment of the requirements for the MSc degree in MSc in Computing Science of Imperial College London

June 2018

Contents

Chapter 1

Introduction

There has been successful applications of deep reinforcement learning (DRL) a number of domains, such as video games ([8]), game of Go ([13]) and robotics ([7]). However, there are still a number of issues to overcome in this method. First, it requires a large amount of data for training the model, which requires a long time of learning process. Second, it is considered to be a black-box, meaning the decision making process is unknown to human user and therefore lacks explanation ability. Third there is no thoughts process to the decision making, which, as XX points out, is a fundamental to the artificial general intelligence. To tackle these problems, there are a number of different approaches the researchers have been experimenting.
One of the methods is to incorporate symbolic representations into the system to achive more data-efficient learning, which shows a promissing results of this approach ([3]). In this paper, we explore the potential of inductive logic problemming into reinforcement learning and attempt to solve the problems above. The advantages of symbolic reinforcement learning are, first of all, the decision making mechanism is understandable by humans rather than being black-box. Second, it resembles how human reason. There is some aspects of try-and-error in human learning, but they explore reasonings to efficiently learn the surrounding or situations. Also they effectly use previous experience (background knowledge) when encountered a similar situation. Finally, the recent advance of ILP research is remarkable and there are a number of new algorithsm that effectively work in non-monotonic senarios based on Answer Set Programmings (ASPs).
Recently there are a number of research that attempted to incorporate symbolic reasoning into DRL, but the combining inductive logic programming and reinforcement learning has not been explored. Because of the recent advancement of logic-based learning and deep reinforcement learning, combination of both approach would be a next explonation toward artificial general intelligence.
In this paper, our objectives is to explore this new reseach field and see how this combination of the two different learning methods could enhance the learning capability.
In this paper, I further explore incorporation of symbolic machine learning into reinforcement learning to achieve data-efficient learning using Inductive Learning of Answer Set Programs (ILASP), which is the state-of-art symbolic learning method that can be applied to imcomplete and more complex environment. This research

is inspired by [3], but in this paper I explore symbolic represetations process and include more learning aspect.
This background report will be the part of the final report and is organised as follows: In Chapter 2, necessary background of logic, inductive logic programming and reinforcement learning are described for this paper. Chapter 3 discusses previous research in this particular approach and see the drawbacks of each approach. Chapter 4 shows the tentative architecture of our new approach, using Inductive Learning of Answer Set Programs (ILASP) to generate model of the environment. We also describe the outline of the project. Finally the ethics checklist is provided in Chapter 5.

Chapter 2

Background

This section introduces necessary background of Inductive Logic Programming (Section 2.1) and Reinforcement Learning (Section 2.2), which provide the foundations of this work.

2.1 Inductive Logic Programming (ILP)

Inductive Logic Programming (ILP) is a subfield of machine learning research area aimed at the intersection between machine learning and logic-based knowledge representation [9]. The purpose of ILP is to inductively derive a hypothesis H that is a solution of a learning task, which coveres all positive examples and any of negative examples, given a hypothesis language for search space and cover relation ([2]). The possible explanation of covers relation is described by the entailment, as shown in Equation 2.1.

$$B \cap H \models E \tag{2.1}$$

where E consists of positive examples (E^+) and none of the negative examples (E^-). One of the advantage of ILP over statistical machine learning is that the hypothesis the agent learnt can be easily understood by a human, making the decision more transparent rather than black box.
TODO Limitations of Inductive Logic Programming:
In this section, we briefly introduce basic logic notions, Anser set programming (ASP)

2.1.1 Logic Basics

TODO Satisfiable
To compute an inference task, the syntax of the predicate logic program needs to be converted into Conjunctive Normal Form (CNF), or clausal theory, which consists of a conjunction of clauses, where a clause is disjunction of literals, and a literal can include positive literals and negative literals. A literal is either an atom *p* or *not p* (negation by failure).

example 2.1.1. Conjunctive Normal Form (Clausal Theory)

$$(\textit{ rain} \vee \textit{umbrella} \vee \textit{rain\_shoes}) \wedge (\textit{ sunny} \vee \textit{roofer})$$

A *Horn clause* is a subset of CNF, which is a clause with at most one positive literal, where a *definite clause* (also called a *rule* or a *fact*) contains exactly one positive literal, and a *denial* (or a *constraint*) is a clause with no positive literals. A *normal clause* extends Horn clause with negation as failure.

2.1.2 Stable Model Semantics

Having defined the syntax of clausal logic, we now introduce its semantics under the context of stable model. The semantics of the logic is based on the notion of *interpretation*, which is defined under a *domain*. A *domain* contains all the objects that exist. For the convenient reasons, we focus on a special interpretations called *Herbrand interpretations* rather than generation interpretations.
The *Herbrand domain* (a.k.a *Herbrand universe*) of clause sets *Th* is the set of all ground terms that are constants and function symbols appeared in *Th*.
The *Herbrand base* of *Th* is the set of all ground predicates that are formed by predicate symbols in *Th* and terms in the *Herbrand domain*.
The *Herbrand interpretation* of a set of definite clauses *Th* is a subset of the Herbrand base of *Th*, which is a set of ground atoms that are true in terms of interpretation.
Interpretation evaluate it to true Interpretation evaluate it to false
A *Herbrand Model* is a Herbrand interpretation if and only if a set *Th* of clauses is satisfiable. In other words, a set of clauses *Th* is unsatisfiable if no Herbrand model was found.
The *Herbrand Model* is a minimum Herbrand model if and only if none of its subsets is an Herbrand model. For definite logic programs, there is a unique minimal Herbrand model (the *Least Herbrand Model M(P)*). For normal logic programs, there may not be a least Herbrand Model.
Definite Logic Program is a set of definite rule, and a definite rule is of the form h $\leftarrow$ a_1, ..., a_n, where h , a_1, ..., a_n are all atoms. h is the *head* of the rule and a_1, ..., a_n are the body of the rule.
Normal Logic Program is a set of normal rule, and a normal rule is of the form h $\leftarrow$ a_1, ..., a_n, *not* b_1, ..., *not* b_n where h is the head of the rule, and a_1, ..., a_n, b_1, ..., b_n are the body of the rule (both the head and body are all atoms).
To solve a normal logic program *Th*, the program needs to be grounded. The *grounding* of *Th* is the set of all clauses that are c $\in$ *Th* and variables are replaced by terms in the *Herbrand Domain*.

example 2.1.2. (Grounding)

$$\text{P} = \begin{cases} p \leftarrow not\ q. \\ q \leftarrow p. \end{cases}$$

ground(P) in Example 2.1.2 is p:- not q. and q:- p. The algorithm of grounding start with the empty program Q = { } and the relevant grounding is constructed

by adding to each rule R to Q given that R is a ground instance of a rule in P and their positive body literals already occurs in the in the of rules in Q. The algorithm terminates when no more rules can be added to Q. Not only the entire program needs to be grounded in order for ASP solver to work, and, unlike Prolog, but also each rule must be *safe*. A rule *R* is safe if every variable that occurs in the head of the rule occurs at least once in body$^+$(R) .
Since there is no unique least Herbrand Model for a normal logic program, [4] defined Stable Model of a normal logic program. In order to obtain the Stable model of P, P needs to be converted using *Reduct* with respect to an interpretation X. First, if the body of any rule in P contains an atom which is not in X, those rules need to be removed. Second, all default negation atoms in the remaining rules in P need to be removed.

example 2.1.3. (Reduct)
X = p

$$\text{P} = \begin{cases} p \leftarrow not\ q. \\ q \leftarrow not\ p. \\ r \leftarrow p. \end{cases}$$

In the Example 2.1.2, P^x is p, r $\leftarrow$ p.. A stable model of P is an interpretaiton X if and only if X is the unique least Herbrand Model of ground(P)^x in the logic programs.

2.1.3 Anwer Set Programming (ASP) Syntax

Answer set of normal logic program P is a stable model, and Answer Set Programming (ASP) is a normal logic program with extensions: constrains, choice rules and optimisation. ASP program consists of a set of rules, where each rule consists of an atom and literals.A literal is either an atom *p* or its default negation (negation as a failure) *not p* .
A Constraint of the program P is of the form $\leftarrow$ a_1, ..., a_n, not b_1, ..., not b_n, where the rule has an empty head. The constraint filters any irrelevant answer sets. When computing ground(P)$_x$, the empty head becomes $\perp$, which cannot be in the Answer Set. There are two types of constraints: *hard constraints* and *hard constraints*. Hard constraints are strictly satisfied, whereas soft constraints are must not be satisfied but the sum of the violations should be minimised when solving ASP.
Choice rule can express possible outcomes given an action choice, which is of the form l{h_1,...,h_m}u $\leftarrow$ a_1, ..., a_n, not b_1, ..., not b_n where l and u are integers and h_i for $1 \leq i \leq m$ are atoms. The head is called *aggregates*.
optimisation statement is useful to order the answer sets in terms of preference, which is of the form #minimize[a_1=w_1,...a_n=w_n] or #maximize[a_1=w_1,...a_n=w_n] where w_1,..., w_n is an integer weight and a_1,...,a_n is an ground atom. ASP solvers compute the scores of the weighted sum of the sets of ground atoms based on the true answer sets, and find optimal Answer Sets which either maximise or minimise the score.

Clingo is one of ASP modern solvers that executes the ASP program and returns answer sets of the programs ([**?**]), and we will use *Clingo* for the implementation of this research.
TODO ASP has true, false and unknown

2.1.4 ILP Under Answer Set Semantics

There are several ILP non-monotoic learning frameworks under the Answer set semantics . We introduce two of them: *Cautious Induction* and *Brave Induction* ([11]), which are foundations of *Learning from Answer Sets* discussed in Section **??**. (for other ILP frameworks, see [10], [5], [1] and [2]).

Cautious Induction

Cautious Induction task is of the form $\langle$ B, E^+, E^- $\rangle$, where B is the background knowledge, E^+ is a set of positive examples and E^- is a set of negative examples.
$H \in ILP_{cautious} \langle$ B, E^+, E^- $\rangle$ if and only if there is at least one answer set A of B $\cup$ H (B $\cup$ H is satisfiable) such that for every anser set A of B $\cup$ H:
$\forall$ e $\in E^+$: e $\in$ A
$\forall$ e $\in E^-$: e $\notin$ A

example 2.1.4. Cautious Induction (Example)
B = $\begin{cases} exercises \leftarrow not\ eat\_out. \\ eat\_out \leftarrow exercises. \end{cases}$
E_+ = tennis
E_- = eat_out
One possible $H \in ILP_{cautious}$ is {tennis$\leftarrow$ exercises, $\leftarrow$ not tennis }.

The limitation of Cautious Induction is that positive examples must be true for all Answer Sets and negative examples must not be included in any of the Answer Sets. These conditions may be too strict in some cases, and Cautious Induction is not able to accept the case where positive examples are true in some of the Answer Sets but not all Answer Sets of the program.

example 2.1.5. Limitation of Cautious Induction (Example)
1{situation(P, awake), situation(P, sleep)}1 :- person(P).
person(john).

In the example 2.1.4, neither of situation(john, awake) or situation(john, sleep) is false in all answer sets. In this example, it only return person(john). Thus no examples could be given to learn the choice rule.

Brave Induction

Brave Induction task is of the form $\langle$ B, E^+, E^- $\rangle$ where, B is the background knowledge, E^+ is a set of positive examples and E^- is a set of negative examples. H $\in$

ILP_{brave} $\langle$ B, E^+, E^- $\rangle$ if and only if there is at least one answer set A of B $\cup$ H such that:
$\forall$ e $\in$ E^+ : e $\in$ A
$\forall$ e $\in$ E^- : e $\notin$ A

example 2.1.6. Brave Induction (Example)
B = $\begin{cases} exercises \leftarrow not\ eat\_out. \\ tennis \leftarrow holiday \end{cases}$
E_+ = tennis
E_- = eat_out
One possible H $\in$ ILP_{brave} is {tennis}, which returns {tennis, holidy, exercises} as Answer Sets.

The limitation of Brave indction that it cannot learn constraints as shown in the example 2.1.4.
TODO MORE EXPLANATION

example 2.1.7. Limitation of Brave Induction (Example)
B = $\begin{cases} 1\{situation(P, awake), situation(P, sleep)\}1 \leftarrow person(P). \\ person(C) \leftarrow super\_person(C). \\ super\_person(john). \end{cases}$
In order to learn the constraint hypothesis H = { $\leftarrow$ not situation(P, awake), super_person(P)}, it is not possible to find an optimal solution.

2.1.5 Inductive Learning of Answer Set Programs (ILASP)

Learning from Answer Sets (LAS)

Learning from Answer Sets (LAS) was developed in [6] to faciliate more complex learning task that neither Cautious Induction nor Brave Induction could learn. Examples used in LAS are *Partial Interpretations*, which are of the form $\langle$ e^{inc}, $e^{exc}\rangle$. (called *inclusions* and *exclusions* of e respectively). A Herbrand Interpretations extends a partial interpretation if it include all of e^{inc} and none of e^{exc}.
LAS is of the form $\langle$ B, S_M, E^+, E^- $\rangle$, where B is background knowledge, S_M is hypothesis space, and E^+ and E^- are examples of positive and negative partial interpretations. S_M consists of a set of normal rules, choice rules and constraints. S_M is specified by *language bias* of the learning task using *mode declaration*. Mode declaration, and specifies what can occur in an hypothesis by specifying the predicate, and has two parts: *modeh* and *modeb*. modeh and modeb are the predicates that can occur in the head of the rule and body of the rule respectively. Language bias is the specification of the language in the hypothesis in order to reduce search space for the hypothesis. Given a learning task T, the set of all possible inductive solutions of T is denoted as ILP_{LAS}(T), and a hypothesis H is an inductive solution of ILP_{LAS}(T) $\langle$ B, S_M, E^+, E^- $\rangle$ such that

1. $H \subseteq S_M$
2. $\forall e \in E^+ : \exists A \in$ Answer Sets$(B \cup H)$ such that A extends e
3. $\forall e \in E^- : \nexists A \in$ Answer Sets$(B \cup H)$ such that A extends e

example 2.1.8. LAS (Example)
TODO

Inductive Learning of Answer Set Programs (ILASP)

ILASP is an algorithm that is capable of solving LAS tasks, and is based on two fundamental concepts: positive solutions and violating solutions.
A hypothesis H is a positive solution if and only if

1. $H \subseteq S_M$
2. $\forall e^+ \in \exists A \in AS(B \cup H)$ such that A extends e^+

A hypothesis H is a violating solution if and only if

1. $H \subseteq S_M$
2. $\forall e^+ \in E^+ \exists A \in$ Answer Sets$(B \cup H)$ such that A extends e^+
3. $\exists e^- \in E^- \exists A \in$ Answer Sets$(B \cup H)$ such that A extends e^-

Given both definitions of positive and violating solutions, ILP_{LAS} is positive solutions that are not violating solutions.

Learning from Ordered Answer Sets (LOAS)

LOAS is an extension of ILASP by incorporating learning capability of weak constraints, is useful for modelling the agent's preferences ([**?**]). Examples used for learning are ordered pairs of partial answer sets, which are of the form o = $\langle$ e_1,e_2 $\rangle$. An ASP Program P *bravely respects o* if and only if

1. $\exists$ A_1, $A_2 \in$ Answer Sets(P) such that A_1 extends e_1, A_2 extends e_2, and $A_1 \succ_P A_1$

P *cautiously respects o* if and only if

1. $\forall$ A_1, $A_2 \in$ Answer Sets(P) such that A_1 extends e_1, A_2 extends e_2, and $A_1 \succ_P A_1$

LOAS task is of the form T = $\langle$ B, S_M, E_+, E_-, O^b,$O^c\rangle$ where O^b and O^c are brave and cautious orderings respectively, which are sets of ordering examples over set of positive partial interpretations E^+. A hypothesis H is an inductive solution of T if and only if

1. $H \subseteq S_M$ in $ILP_{LOAS}(T)$
2. $H' \in ILP_{LAS}(\langle B, S_{LAS}(M_h, M_b), E^+, E^- \rangle)$ where H' is the subset of H without weak constraints
3. $\forall o \in O^b$ $B \cup H$ bravely respects o
4. $\forall o \in O^c$ $B \cup H$ cautiously respects o

A Context-dependent Learning from Ordered Answer Sets

ILASP task containing a contex-dependent example
context-dependent partial interpretation (CDPI)
Context-dependent ordering example (CDOE)
Two advantages of adding contex-dependent are it increases the efficiency of learning tasks, and more expressive structure of the background knowlege to particular examples.

2.2 Reinforcement Learning

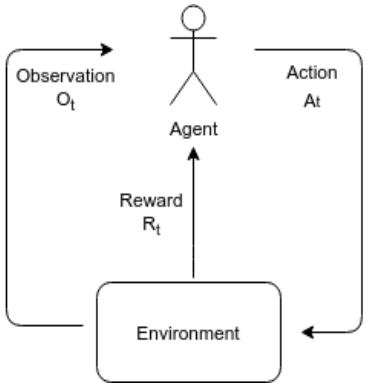

Figure 2.1: Agent and Environment

2.2.1 Markov Decision Process(MDP)

A state S_t is Markov if and only if $P[S_{t+1} \mid S_t] = P[S_{t+1} \mid S_1, ..., S_t]$ therefore the probability of reaching S_{t+1} depends solely on S_t, which captures all the relevant information from earilier history. (Puterman, 1994) There is an agent who interacts with an envirnment (Figure 2.1). At each time step, an action taken by the agent affect the environment state and the reward (or penality) it receives from the action outcome. (TODO Observations) When an agnet must make a sequence of decision, the sequential decision problem can be formalised using Markov decision process (MDP). MDPs formaly represent a fully observable environment of an agent for reinforcement learning.

A MDP is of the form $\langle$ S, A, T_a, R_a, γ $\rangle$ where:

- S is the set of finite states that is observable in the environment
- A is the set of finite actions taken by the agent
- T_a(s, s′) is a state transition in the form of probability matrix Pr(S_{t+1} = s′ | s_t = s, a_t = a), which is the probablity that action a in state s at time t will result in state s′ at time t+1.
- R is a reward function R_a(s, s′) = $\mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$, the expected immediate reward that action a in state s at time t will return
- γ is a discount factor $\gamma \in$ [0,1], which represents the preference of the agent for present rewards over future rewards

In MDPs, there is an element of delayed reward tradeoff between immediate rewards and its delayed reward
A solution to the sequential decision problem is called a policy π, a sequence of actions that leads to a solution
The transition and reward functions are not necessary to be known to compute π.
An optimal policy π^* is the one that maximise the total rewards in the environment.
The total reward with a discount factor is
The existence of the discount factor can be justified as follows:
TODO $\mathbb{E}[R_{t+1}| S_t = s]$
Reinforcement learning is a method to get approximated optimal solution.

2.2.2 Policy and Value Function

An agent follows a policy
A policy π is optimal if it maximises the action-value function
TODO on-policy and off-policy learning.
$A_t = \pi(S_t)$
One of the common possibilibies is that the agent chooses an action in order to maximise the discounted sum of future rewards.
A_t to maximise $R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...$
An action-value function evaluate a particular state by taking an action according the policy
$q_\pi = \mathbb{E}[R_{t+1}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... \mid S_t = s, A_t = a, A_{t+1} = a,]$

2.2.3 Temporal-Difference (TD) Learning

To solve MDP, one of the approaches is called Temporal-Difference (TD) Learning.
TD learns directly from episodes of experiences, which can be imcomplete. TD does not require knowledge of MDP transitions and rewards (model-free)
Update value

where $R_{t+1} + \gamma V(S_{t+1})$ is the estimated return (a.k.a TD target)
$R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is TD error.sudo add-apt-repository ppa:texworks/stable
TD updates the estimate by using the estimates of XXX (bootstrap).
The advantages of TD methods - does not require any model of an environment. - online learning

2.2.4 Q-Learning

Q-learning is off-policy TD learning defined in [Watkin 1989], where the agent will not rely on the models of the environment (model-free leaning) and only knows about the possible states and actionns. The transition states and reward probability functions are unknown to the agent. It is of the form:
where α is the learning rate, γ is a discount rate between 0 and 1. The equation is used to update the state-action value function called Q function. The function Q(S,A) predicts the best action A in state S to maximise the total cumulative rewards. The optimal Q-function $Q^*(s,a)$ represents XXX for the agent to selection action a given that it is in state s.
TODO INSERT Q-learning ALGORITHM HERE
Epsilon greedy

2.2.5 Model-based vs Model-free Learning

A model M is a representation of an MDP where the state space and action space are assumed to be known. The model represents state transitions and rewards. In this case, the problem to be solved becomes a planning problem. A planning is for a series of actions to achive the agent's goal. Most of the reinforcement learning problems are model-free learning, where M is not unknown and the agnet learns to achieve the goal by the interactions with the environment. Thus the agent knows only possible states and actions, and the transition state and reward probability functions are unknown.
the performance of model-based RL is limited to optimal policy given the model M. Thus when the model is not a representation of the true MDP, the planning algorithms will not lead to the optimal policy, but suboptimal policy. The solution to this problem is either to use model-free approach when the model is inaccurate or incorporate undertainty of the model using XXX.
One algorithm which combine both aspects of model learning is called Dyna ([14]), which is shown in Figure 2.2.
Dyna learns a model from real experience and use the model to generate simulated experience to update the evaluation functions.
This approach is more effective because the simulated experience is relatively easy to generate compared to having to have real experience, thus less iterations.
TODO ADD MATH

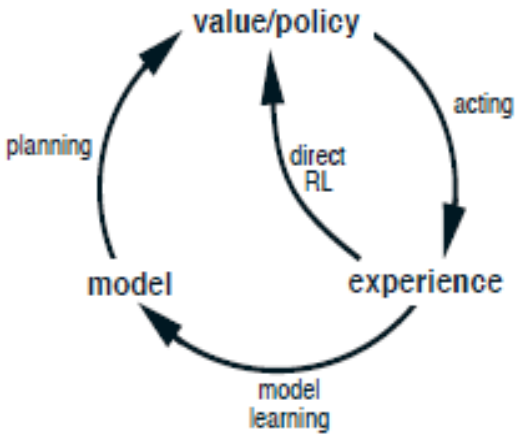

Figure 2.2: Relationships among learning, planning and acting **??**

Chapter 3

Related Work

The use of symbolic representations to achieve data-efficient learning was traditionally discussed in relational reinforcement learnign (RLL).
More recently, **??** introduced the proof of concept for incorporating symbolic front end as a means of converting low-dimensional symbolic representation into spatio-temporal representations, which will be the state transitions input of reinforcement learning.
just a symbolic representations and not learning segmentation was done manually.
As demonstrated in XXX, a single pixel can dramatically influence the policy (Maybe not so important here??).
This symbolic approach was followed by **??**, which improved simbolic representations in two ways. First,
Transparency and interpretable capability of the model is another important aspect for machine learning applications.
XXX[Programmati...] developed a programmatically interpretable reinforcement learning which finds a policy that can be represented in a human-readable programming language.
Two most studied approach for using previous learning exprience is meta-learning and transfer learning
Artificial General Intelligence
Definision of AGI
The history of data-efficient learning What other people have done in this.
The advance of statistical machine learning methods, especially deeep reinforcement learning
AlphaGo, and AlphaGo Zero
Study of symbolic machine learning roots from
Baysian Optimisation
Symbolic Deep reinforcement learning
Some implementation: German paper
The paper was the application of symbolic representatiojns into a very simple game to demonstrate this proof of concept would actually work. By contract, there is active reserach in symbolic machine learning, which focuses on logic=based learning rather than statistical machine learning. For example, XX shows the agent can learn XXX from a noisy examples, with only a very few training examples.

Incorporation of logic into reinforcement learning dates back to the study of relational reinforcement learning,
More recently there has been a number of attempts to incorporate ASP into reinforcement learning.
There are a number of reseached conducted in applying DNN to symbolic reasoning. For example,
[From GamePlay to Symbolic Reasoning]
TODO Explain how symbolic learning works

Chapter 4

Project Overview

4.1 Proposed Architecture

The proposed tentative architecture is shown in Figure 4.1. The overall architecture resembles Sutton's Dyna architecture ([14]), which combined both model-free learning and model-based learning.

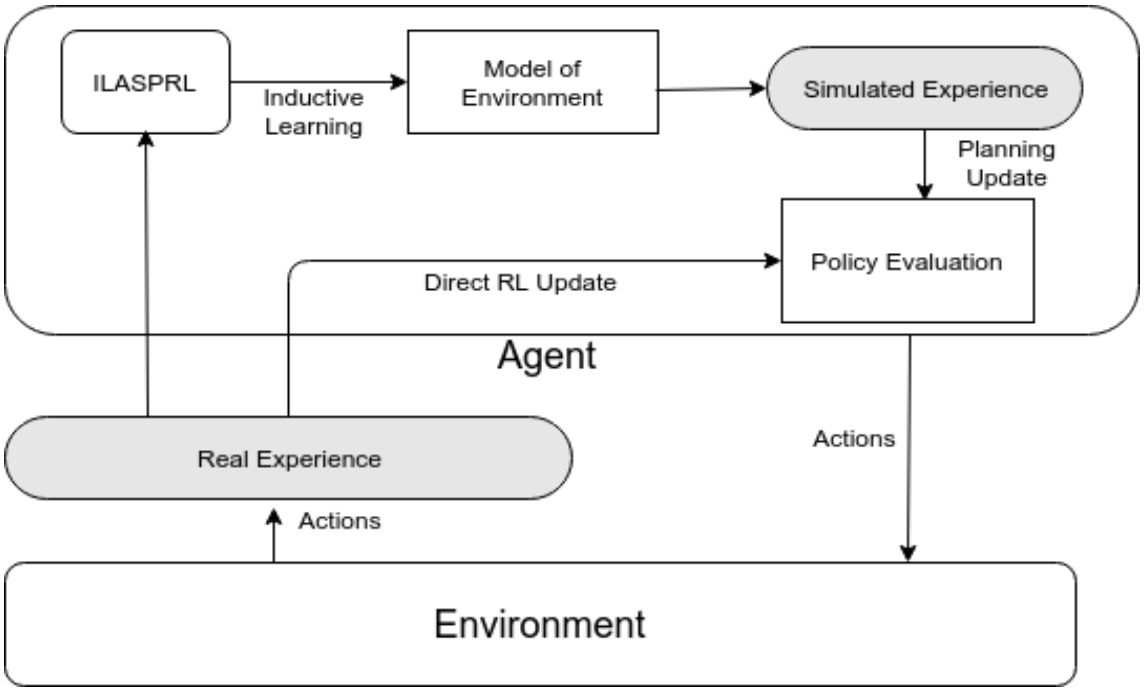

Figure 4.1: Proposed reinforcement learning architecture. ILASPRL learns to generate a model and updates based on the interaction with the environment, which is used to facilite the policy evaluation.

4.1.1 ASP Representation Generation

The inputs from the experience needs to be converted in ASP form, which can be used to proceed the inductive learning in ILASP(RL). Observations from the inter-

action with the environment are state transition, reward and action of the agent, which can be directly converted using XXX. Example 4.1.1 is ASP input form that is feeded into ILASP(RL) phase.

example 4.1.1. *(Examples for ILASPRL (input))*
agent_before(S1, T1).
agent_after(S1, T1). reward(R). action(A).

4.1.2 Model Generation and Update using ILASP(RL)

Once the input is converted, the agent must learn the following definition of the model of the environment.

example 4.1.2. *(Example of learning task (output))*
valid_move(C0, C1):- previous(C0, C1).
previous(C2, C1):-
agent_at(C0, C1), adjacent(C0, C2), not obstacle(C0, C2), not enemy(C0, C2).
state_transition(S1, S2, T2):- agent_before(S1, T1), agent_after(S2, T2).

The background knowldge is empty, and each example contain different transition of the agent.
The model is updated as the agent explore more environment.
In addition, when the environment has been changed during the exploration, the agent creates a new model for this new environment.
In two different but similar scenarios, the model is generalised based on the two models, and the agent's model is more general which covers both games. Thus in theory, the agent should still be able to exploit the efficient learning from the model-based learning, facilitating the transfer learning.
Example: model refinement in a simple Grid world
Implement baseline performance (DQN, Q-learning, ASPRL)
Due to the complex environment of the chosen game, I will not implemnet the original DSRL, since the feature extractions from the pixel would only work in a very simple pixel environment,
Pipeline
The basic architecture follows the similar method in XXX, but I also add symbolic learning to these extracted symbolic features using ILASP.
Apply ILASP to the ASP, which involves development of the pipeline of ILASP in Python
Finally use Q-learning that allow
which measurement would you use? (grid word, something else? GVGAL games)
Summarise different types of knowledge representations (Objects ?? relationship?)
Common sense
Lastly, I will also test the capability of transfer learning for this new method.

4.2 Project Outline

The next phase of the individual project is implementation and experiments. I will implement the proposed architecture using Python and ILASP (Clingo), and compare the performance
The platform I am planning to use to measure the performance of my approach will be GVG-AI Framework, which was created for the General Video Game AI Competition [1], which provides game environments for an agent that should be able to play a wide variety of games without knowing which games are to be played. The underline language is the Video Game Definition Language (VGDL), which is a high-level description language for 2D video games providing a platform for computational intelligence research ([12]).
Especially, I will measure the following aspects of the new Algorithms
Learning efficiency
Transfer Learning capability
XXX
The proposed architecture is not finalised and will be reviewed regularly as I proceed the development of the architecture.

[1] http://www.gvgai.net/

Chapter 5

Ethics Checklist

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		✓
Does your project involve the use of human embryos?		✓
Does your project involve the use of human foetal tissues / cells?		✓
Section 2: HUMANS		
Does your project involve human participants?		✓
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from Human Embryos/Foetuses i.e. Section 1)?		✓
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?		✓
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		✓
Does it involve processing of genetic information?		✓
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		✓
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?		✓
Section 5: ANIMALS		
Does your project involve animals?		✓
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		✓

If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		✓
Could the situation in the country put the individuals taking part in the project at risk?		✓
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		✓
Does your project deal with endangered fauna and/or flora /protected areas?		✓
Does your project involve the use of elements that may cause harm to humans, including project staff?		✓
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		✓
Section 8: DUAL USE		
Does your project have the potential for military applications?		✓
Does your project have an exclusive civilian application focus?		✓
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		✓
Does your project affect current standards in military ethics e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		✓
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		✓
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		✓
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?	✓	
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		✓
Section 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?	✓	

Will your project use or produce goods or information for which there are data protection, or other legal implications?		✓
Section 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		✓

Bibliography

[1] Domenico Corapi, Alessandra Russo, and Emil Lupu. Inductive Logic Programming in Answer Set Programming. *Inductive Logic Programming*, pages 91–97, 2012. pages 7

[2] Luc De Raedt. Logical settings for concept-learning. *Artificial Intelligence*, 95(1):187–201, 1997. pages 4, 7

[3] Marta Garnelo, Kai Arulkumaran, and Murray Shanahan. Towards Deep Symbolic Reinforcement Learning. sep 2016. pages 2, 3

[4] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. *5th International Conf. of Symp. on Logic Programming*, (December 2014):1070–1080, 1988. pages 6

[5] Katsumi Inoue, Tony Ribeiro, and Chiaki Sakama. Learning from interpretation transition. *Machine Learning*, 94(1):51–79, 2014. pages 7

[6] Mark Law, Alessandra Russo, and Krysia Broda. Inductive Learning of Answer Set Programs. *European Conference on Logics in Artificial Intelligence (JELIA)*, 2(Ray 2009):311–325, 2014. pages 8

[7] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-End Training of Deep Visuomotor Policies. apr 2015. pages 2

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015. pages 2

[9] S Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991. pages 4

[10] Ramon P Otero. Induction of Stable Models. *Conference on Inductive Logic Programming (ILP)*, pages 193–205, 2001. pages 7

[11] Chiaki Sakama and Katsumi Inoue. Brave induction: A logical framework for learning from incomplete information. *Machine Learning*, 76(1):3–35, 2009. pages 7

[12] Tom Schaul. A video game description language for model-based or interactive learning, 2013. pages 18

[13] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. pages 2

[14] Richard S. Sutton. Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. *Machine Learning Proceedings 1990*, 02254(1987):216–224, 1990. pages 12, 16