**多智能体逻辑编程策略梯度**

摘要

人类通常拥有对社会规则进行归纳的能力。在社会生活的活动中，人们通常会总结一些社会规则指导行为，从而对环境进行更好的适应。在一般的多智能体算法中，智能体通常利用神经网络对策略进行参数化表示，导致它们学习到的策略不可解释与难以迁移到不同的环境中，在泛化能力与可解释性方面均存在不足。受到对人类归纳规则能力的启发，我们从MADDPG算法出发，引入可微分归纳逻辑编程对智能体的行为进行归纳学习，目标是学习到多智能体遵守的可解释的规则并指导智能体的行为。我们发展出了一个中心化的规则学习函数，并与中心化训练，去中心化执行动作的框架相结合，交替优化规则函数与策略函数的算法框架，称之为多智能体逻辑编程策略梯度算法(MALPPG)。在规则学习函数中，我们提出了将规则归纳学习问题的转化为对动作-状态进行分类问题的方法，并利用DILP提出的可微分推理函数加以训练求解。在井字棋实验中，我们对规则函数进行测试，得到了有意义的“围堵”对方的博弈规则。此外，我们提出了一个新的多智能体实验设计，电动扶梯实验，以模拟人群上扶梯的行为。在此实验中，新算法表现出了更好的效果，同时归纳得到了可解释的社会规则，在不同环境设置下的稳健性也高于传统多智能体算法。我们的实验表明，通过引入智能体对规则的归纳理解对于智能体的学习与适应新环境有着重要的意义。

**MULTI-AGRNT LOGIC PROGRAMMING POLICY GRADIENT**

**ABSTRACT**

Humans are capable of performing induction for social rules. In daily life, the capacity is critical for adapting to new environment. In traditional multi-agent reinforcement learning algorithms, the policy of an agent is often represented by neural network, which is a parameterized model. The lack of interpretability for neural network also exists in policy network. It’s also difficult to adapt to new environment using the same policy network, that is, generalization problem in reinforcement learning. Inspired by induction capacity for rules of human beings. We start from MADDPG, a classical multi-agent algorithm, and introduce a differentiable inductive logic programming framework for multi-agent reinforcement learning, expecting to learn meaningful rules and use them to improve performance of the agents. We develop a centralized-training rule function, combine it with centralized-training-decentralized-execution reinforcement learning framework, construct a pipeline of optimizing rule learner and policy network alternately, namely Multi-Agent Logic Programming Policy Gradient (MALPPG). In rule learner, we propose a method that be able to convert a rule learning problem to a classification problem for state-action pair. And we employ a differentiable infer function in DILP to solve it. Our rule learner is tested on tic-tac-toe game. It gets useful rules for blocking opponent. Besides, we design a new experiment for multi-agent system, escalator game. In this new game, our algorithm performs better than MADDPG, meanwhile, interpreted rules are learned by agents. Under different environment setting, MALPPG is also robust. Our experiment show that it’s critical to introduce a rule learner for multi-agent reinforcement learning.

[ABSTRACT 2](#_Toc40281083)

[第一章 引言 4](#_Toc40281084)

[1.1 研究背景 4](#_Toc40281085)

[1.2 研究现状 5](#_Toc40281086)

[1.3 论文贡献与结构 5](#_Toc40281087)

[第二章 相关理论基础 6](#_Toc40281088)

[2.1 归纳逻辑编程 6](#_Toc40281089)

[2.2 可微分归纳逻辑编程 6](#_Toc40281090)

[2.3 多智能体强化学习 7](#_Toc40281091)

[第三章 多智能体逻辑编程梯度策略算法构建 9](#_Toc40281092)

[第四章 实验设计与验证 12](#_Toc40281093)

[4.1井字棋实验 12](#_Toc40281094)

[4.2电动扶梯实验 14](#_Toc40281095)

[第五章 总结与展望 17](#_Toc40281096)

[参考文献 17](#_Toc40281097)

[致谢 18](#_Toc40281098)

# **第一章 引言**

## 1.1 研究背景

现实生活中很多问题可转化为多智能体强化学习的问题进行模拟。例如自动驾驶问题中，多辆自动驾驶汽车的配合问题，快递机器人的协同配合问题。多智能体强化学习问题由强化学习问题衍生而来，是单智能体强化学习的扩展，强化学习中存在的很多问题也存在在多智能体强化学习当中。与单智能体强化学习一样，多智能学习算法一般用神经网络参数化表示策略。神经网络表示的策略存在着根本问题，即可解释性较弱，无法对学习到的策略做出有意义的解释。然而，现实中，多智能体之间或与环境交互，往往会学习到一些默认的规则，但参数化的网络却无法准确表示这样的规则。另外的一个难点则是强化学习泛化能力较弱，对于环境稍有改变，之前训练好的模型的效果很有可能下降很多，由神经网络表示的策略或许得到了一些规则的模式，却无法很好的迁移到新的环境中。

观察日常生活，我们会发现人们在现实生活中适应环境的过程往往会总结出一些规则，通过归纳得到的规则对执行动作做出指导，通过环境反馈改变规则再去适应。这样的一些好处是，其一，可被归纳和解释的规则将有利于我们更好的对环境做出反应。其二，在新的，稍有改变的环境中，我们可以直接利用经验，即之前学习到的规则，可以更快的适应环境。

基于这个启发，我们考虑对于逻辑规则学习的方法来学习规则。符号学习是人工智能的一大分支，它的特点是模仿人的逻辑推理过程，由于可用逻辑表达式表示推理，可解释性极强，也具有很强的表达能力。所以，对于学习可解释的规则有巨大的优势。但缺点在于是离散化的，不可微分的，且往往需要数据非真即假，对数据真实性有较高要求，同时命题的表示往往只有真假命题，无法覆盖现实中的大多数情况。

随着深度学习的普及深入，在符号学习中引入深度学习成为趋势。如，Evans提出的可微分的归纳逻辑编程框架，DILP。Neural logic machines将命题张量化，并构造了基于张量化命题的深度学习框架。这些框架很好的沟通了符号学习与深度学习，实现了参数化逻辑归纳推理与训练的框架，解决了符号学习离散化和二值化，以及对数据敏感的缺点。在此基础上，我们希望基于这部分工作引入对智能体互动中社会规则的学习。

所以，在本文中，我们考虑到将多智能体强化学习算法与符号学习进行结合。引入利用微分化的逻辑推理学习框架对多智能体进行规则学习，再与智能体策略优化交互，实现更好的效果。希望达到如下的目标。其一，可以对智能体学习到的策略作出归纳与解释。其二，根据归纳出的规则指导智能体的行动，将达到更好的效果。其三，如果学习到一些可表示的规则，那么在环境发生一些改变时，智能体能够更好的迁移规则，在新环境中更快速的学习，增强其泛化能力。其四，在许多算法中，多智能体之间的相互作用往往无法有效的学习和表示，利用逻辑表达式的特点，可以学习到多个智能体之间的逻辑关系。

## 1.2 研究现状

在结合深度学习与符号学习方面，Evans提出了可微分的归纳逻辑编程框架。归纳逻辑编程是符号学习中的一类算法，Evans将非真即假的命题实值化，并且对逻辑子句引入权重，把逻辑子句的搜索参数化。利用构造的可微分推理函数将逻辑子句，背景命题等连接起来，构成可微分网络。可以对已知真假的命题0,1二值化处理，从而进行监督学习。Neural Logic Machines是符号化的神经网络框架，同时利用了神经网络作为强大的函数拟合器的优点与逻辑编程中对于关系的符号化表示的优点。给出了将逻辑命题张量化的方法与连接逻辑命题的神经网络框架，这个算法可在小规模数据集上得到好的规则并在泛化到大规模问题上也有良好的表现。Ingo Thon在将概率编程引入归纳逻辑编程，将规则学习器FOIL与ProbLog结合，在规则学习上有优秀的结果。

将符号学习方法引入强化学习中也有很多前人的工作。最早在1998年，Saso提出了一种可以进行关系学习的强化学习算法，称为relational reinforcement learning。该算法的贡献在于将强化学习中的Q函数以逻辑回归树替代，逻辑回归树作为Q-value的估计器，优点在于由于逻辑回归树的判断节点由逻辑命题组成，所以，对于状态-动作对有具体的逻辑解释，对强化学习的实体之间可以结构化表示。Google在论文[]中，对此进行了推广，利用self-attention模块替代逻辑回归树，在星际争霸的游戏中成功学到实体之间的关系。相似的有，在DeepMind的论文中，在多智能体环境中，引入图神经网络表示实体之间的联系。此外，在最近的研究中，NLRL算法将DILP的模型结构引入到强化学习中，通过将状态，动作编码为命题并赋予初值，利用DILP构造的Actor函数进行学习，在强化学习的泛化问题上取得了进展，引入DILP结构的强化学习在更复杂环境中有很好的泛化能力。

## 1.3 论文贡献与结构

针对多智能体强化学习中可解释性不强，智能体缺乏对规则的归纳总结，而且泛化能力较弱的情况。本文将可微分可参数化表示的符号推理算法引入多智能体强化学习中，赋予智能体的规则归纳的能力；建立了结合规则学习与强化学习结合的框架，使得策略优化和规则归纳模块相互反馈促进，归纳学习模块指导多智能体策略优化，而策略优化的反馈又促进智能体更好的学习规则。达到了如下目标，1. 学习出可解释的规则命题；2. 达到了更好的策略优化效果；3. 在环境变化中，算法保持了更好的稳健性；4. 提出了一个新的实验模型，扶梯实验，详细规定了它的状态转移与回报等。在这个实验中完整验证了算法的可靠性。

文章的结构如下。第二章引入对相关理论的介绍，包括归纳逻辑编程与可微分的归纳逻辑编程，与多智能体相关的算法。主要为后一章节的算法构建回顾基础知识。第三章开始描述我们对算法的创新与构建，包括理论的描述部分与一些细节实现部分。第四章介绍实验验证的结果，在井字棋实验上验证规则学习模块的有效性，开发了新的实验环境，扶梯实验，并在上面验证了归纳可解释规则的能力，与反馈策略优化提升学习效果的能力。最后一章是总结与对未来工作的规划。

# **第二章 相关理论基础**

## 2.1 归纳逻辑编程

**一阶逻辑编程(First-Order Logic Programming)；**有别于一般的程序设计语言使用命令，函数进行构造。逻辑编程语言是一类利用逻辑表达式进行编程以解决逻辑问题的编程语言。prolog是应用广泛的著名逻辑编程语言，本文将用prolog语言表示逻辑编程相关语句。

逻辑编程的基本数据结构是原子命题(atom)。原子命题α是一个元组其中p是谓词(predicate)， 是项(term)。项可以是变量或者常量。例如，even(X)表示X是偶数；father(X,Y)表示X是Y的父亲；规则的定义需要子句(clause)的概念。子句是如下形式的命题，

其中，都是原子命题，逗号表示与关系，箭头表示蕴含关系。所以这个语句的意义为原子命题蕴含。例如，

上式的意义为，如果X是Y的后继，Y是Z的后继，那么X是Z后继的后继。

**归纳逻辑编程(Inductive Logic Programming)；简称ILP。**归纳逻辑编程是符号人工智能的一个领域。它利用逻辑编程语言作为对样本，背景知识与假设的通用表示。归纳逻辑编程的任务是在给定背景知识与一些逻辑判断的数据集，包括正样本与负样本。一个ILP系统需要构造一个由逻辑编程语言构成的假设语句，即一组子句（clause）构成的规则。通过这个规则，可由背景知识推导出样本中的真命题而不包括样本中的假命题。

给定一组数据，ILP系统的目标是构造clause集合，满足

对于所有。其中，B是背景知识集合，N是负样本，P是正样本。

下面以推导目标谓词Buzz为例介绍ILP任务。在Buzz任务中，我们需要判断一个自然数X是不是3的倍数。即，当X = 3\*Y，Y是正整数时，Buzz(X)为真，反之为假。由此，我们可以给定一组包含真假命题的数据集P，N：

背景知识**B**是对自然数集的一个描述：

ILP系统的任务是找到一组规则R，使得通过对背景知识B应用R进行推导后能够推导出符合P,N数据集真假的结论。对于Buzz问题，其中的一个解R是：

上述命题中，pred1,pred2是辅助命题。对这组命题的解读可从下至上逐步代入。可解释为，pred2(X,Y)表示Y是Z后继的后继；pred1(X,Y)可表示Y是X后继的后继的后继；那么如果Y是3的倍数，且满足X和Y有pred2的关系，即X是Y的后继的后继的后继，那么X也是3的倍数；最上面一条说明了，0也是3的倍数。

## 2.2 可微分归纳逻辑编程

ILP的优点在于可以高效的利用少量数据进行符号推理，对样本数据有极强的可解释性。但是缺点在于对于部分标签失真的模糊数据集健壮性不强；以及不涉及符号的数据无法应用。神经网络的优点在于对于有噪声的数据集有较强的拟合能力，且鲁棒性较强。通过结合两者，可以得到一个兼具符号推理能力与强数据拟合能力的神经网络结构。Evans在文献[1]中对此作出了尝试，通过将连续可微的算符代替离散的操作，构建了一个可微分的逻辑编程框架(DILP)。因为具有可微分的性质，故而可以通过建立loss函数进行优化，通过优化loss得到ILP问题的解。DILP算法具有较强解释性与高效的数据利用率和泛化能力。

将ILP微分化的重要一步是将基本命题连续化，有别于一般的对命题非真即假的判断。首先我们需要对每个命题指定一个**值(valuation)** 表示一个命题为真的概率。则**valuation**定义为一个命题到[0,1]区间实数值的映射：

其中，G是基本命题(ground atoms)。例如命题P = {r/2}表示有两个terms的命题，谓词为r。对于所有常数C={a ,b}；对r(a ,a),r(a ,b),r(b ,b),r(b ,a)都指派一个[0,1]实数表述为真的程度。对于数据集的正样本P，每个命题赋值1。对每个数据集的负样本命题赋值0。所以，现在我们已构造好了标签数据集**L**。

我们期望的可微分模型应该有如下形式，

即对于命题α，通过构造的模型，输出值应为在权重为W，背景知识为**B**条件下的概率值。期望概率p能尽可能预测标签为真的概率。从而可以构造loss函数为:

DILP的核心算法表示为函数

DILP的输入为由两部分组成，背景知识命题与由模板构造的命题集合。由于背景知识为真命题，B中所有命题的value全为1。此外，我们需要指定用于推导的命题形式，如我们需要使用pred(X,Y),succ(X,Y),zero(X)推导目标命题target(X,Y)。那么需要够构造命题的形式为,

即指定谓词pred, succ, zero, target以及它们的term个数，代入所有常数c的组合后构造出输入命题的集合，初始value设置为0。之后对于输入命题进行组合产生clause集合。对于每两两clauses组合给以权重。权重表示了clauses在推导命题中的重要程度。在DILP中，对应于逻辑编程中的推理步骤，是逻辑推理的连续可微形式。当赋有初始值的命题通过构造的clauses pair 由函数进行推导T步之后，将更新命题value值为。下一步中，我们将模型输出的value值与标签数据计算loss值，对模型参数W进行反向传播更新。

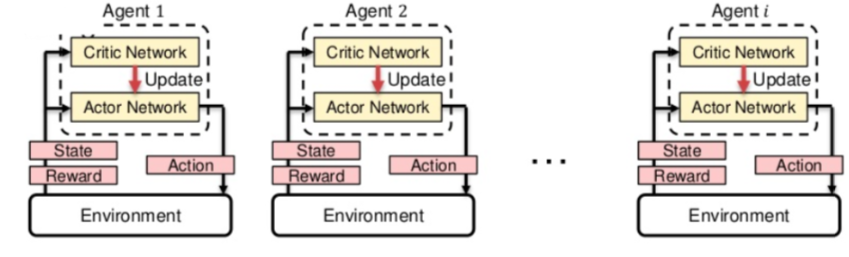
对规则的学习，我们只需提取出权重较大的clauses即可组成规则R。

## 2.3 多智能体强化学习

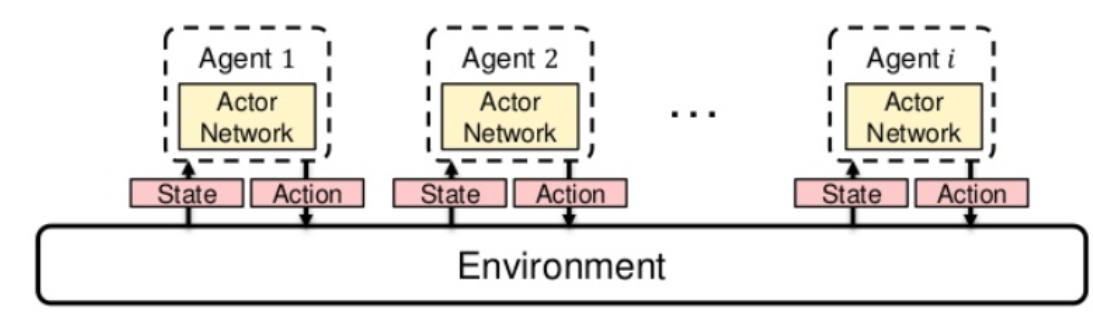
多智能体是单智能体学习算法的扩展。但是由于涉及到多个智能体之间的相互作用，比如竞争合作。环境的变化更为不稳定，联合动作空间维度为指数增加。故而多智能体强化学习算法相对而言较为复杂。

类似于单智能体系统，多智能体强化学习算法建立在马尔科夫博弈(又称随机博弈)的基础上。一个马尔科夫博弈被定义为一个元组。其中n表示agent数量；S是环境的状态；是第i个agent的动作；是第i个agent的奖励函数,，表示在采取联合动作后由状态s跳转至动作s’ 得到的奖励。T是状态转移函数；[目标函数与更新]

从单智能体向多智能体推广，有两种较为直观简单的方法。第一种是完全分布式的方法，分散执行动作，分散训练网络。对每个agent，都有自己的actor-critic函数。每次采样时，每个agent在环境中独立执行动作，相互之间无任何通信。训练网络时独立训练。这个方法称之为multiple single agent方法。如图2-1所示：



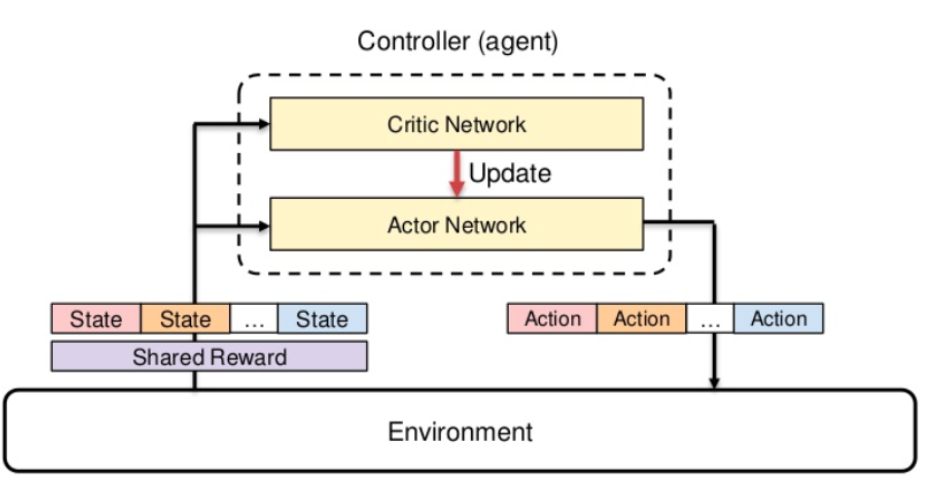
(1)训练网络



(2)执行动作

**图2-1** multiple single agent

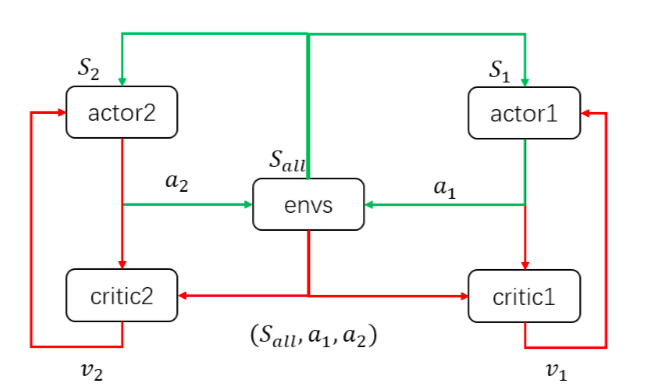
第二种方法是完全中心化的方法，即将所有agents视为一个agent，每次需要预测并执行一个联合动作，共享一个actor-critic网络与相同的奖励函数，并进行集中的训练。其架构如图2-2所示：



**图2-2 中心化多智能体算法**

第一种方法的缺点在于,对每个agent而言，环境是不稳定的。因为对每个agent来说，其它agents视为环境，则在每一次agent执行动作时，其它agents也执行相应的动作。那么环境已经改变，无法保证环境的转移概率是稳定的，即。而第二种方法的缺点在于动作空间维度爆炸。

MADDPG算法由google于2018年提出[2]。它通过建立集中式训练，分布式执行的框架，解决了从单智能体到多智能体推广中环境不稳定与动作空间维度爆炸的问题。如图2-3所示，每个智能体的critic函数的输入为所有的状态分量，与每个智能体的动作。由于critic函数掌握了所有的状态信息与动作信息，可以提供集中式训练。对于所有agents的动作进行评判打分。集中训练的评价函数提供了对全局的信息的评判，避免了环境的不稳定。每个智能体的actor函数，只输入局部观察信息，并输出动作。这样对于动作函数，避免了动作空间维度爆炸的难题。



**图2-3 MADDPG算法框架**

# **第三章 多智能体逻辑编程梯度策略算法构建**

在之前的章节中，我们描述了现有的多智能体算法存在的不足。在这一章节中，我们将提出一种新的算法，多智能体逻辑编程梯度策略方法(Multi-Agent Logic Programming Policy Gradient; MALPPG)，目的是为了解决多智能体系统中规则学习的问题。接下来我将描述如何通过在MADDPG算法中扩展归纳逻辑编程来解决这一问题。

MADDPG算法建立在随机博弈的理论基础之上，需要先定义一个随机博弈；为了能够在MADDPG算法中建立归纳逻辑编程的语言，我们首先需要建立一个与之对应的具有逻辑命题的随机博弈元组。我将其称之为逻辑编码器(Logic Encoder)。一个Encoder需要定义一对映射；其中,

其中O表示每个agent能接收到的局部观测信息(observations)，是每个观测分量。并且；即O是状态S分量的子集。映射表示将每个可观测分量编码为原子命题。映射表示将动作空间中的每个动作编码为相应的命题。同理可以定义逻辑解码器(Logic Decoder)；逻辑解码器就是定义一对逆映射，分别为的逆映射。逻辑编码器与逻辑解码器建立了沟通ILP与多智能体系统的桥梁，从而建立了一个具有逻辑命题的马尔科夫博弈组。

无论是游戏，博弈，还是现实生活中多体交互的场景。除开规则制定者指定的规则限制之外，当智能体逐渐在环境中稳定适应，对于每个agent，总存在一些普适的规则，无论可表示与否，如果能挖掘出它们在交互过程中得到的隐式规则，并以此为指导，将会使智能体的决策更有效。受到这个思想的启发，我们将在多智能体算法的训练与执行中应用逻辑规则学习的算法模拟智能体总结规则并应用的过程。首先，我们从智能体在环境中的执行与采样数据开始说起。由流程图[5]所示，智能体的规则学习与强化学习将交替进行。在最初的环境采样与学习阶段，智能体不受规则约束，则它的动作选择仅为actor函数的输出。在此阶段，当智能体逐渐得到对一些高奖励动作的有效判断后，同时我们在环境中采样动作-状态-奖励序列，作为规则学习的输入。在采样中，对于智能体i我们获得了序列。每一个动作-状态-奖励组将提供一个具有标签的数据。

在接下来的模块，我们将建立起与多智能体强化学习相适配的规则监督学习框架。我们定义需要学习到的规则为一组clause的集合。规则是一个由部分观测信息到动作的映射，。在逻辑命题中，其形式为。称之为目标命题(target predicate)。我们将需要对动作空间里的每个动作视为目标命题，目标是得到当满足怎么样的情况之下，即，将采取动作。对采样得到的每个序列对，表示了在观测到信息为的情况之下，采取动作，将得到奖励。它提供了对于观测信息，执行某动作的好坏的评判。当智能体已经在环境中得到较稳定的具有高回报的对时，我们认为这样的序列得到了有效的对动作的预测。先从简单情况考虑，把问题化归为二分类问题，那么对于某一目标命题，我们认为如果动作-状态对满足，那么将第i个动作-状态对标记为正样本，反之，标记为负样本，即。同时，将作为背景知识；此外，背景知识还应包括一些基本公理与定义，例如，在涉及自然数的命题判断时，我们应该包括对自然数集的定义，

所以，综上所述，我们构造了ILP问题的数据特征**：**

其中，是目标动作命题，N是序列长度，。在DILP系统中，我们令所有**B**与**P**中所有命题的valuation为1；令**N**中所有命题的valuation为0。将规则学习问题转化为一个二分类的问题。更一般的，我们将在进一步的工作中考虑将问题推广为一个回归问题。可以做如下推广，考虑到将目标动作分类为P,N两类将导致数据的噪音过大，鲁棒性不强。我们将直接对动作序列中的每个动作赋值。但需要在智能体较少的情况下应用DQN等算法。具体来说，可作如下赋值，对每对，计算，对于所有的子序列，根据Q值进行聚类，可更为细致的判断动作对于观测信息的价值多少。从而根据聚类结果，将动作序列赋值到[0,1]区间。从而构造更为合理的标签。

得到具有logic解释的智能体采样数据后，系统将交替到对Rule函数的训练。这里可以有两种解决方案。第一种，分布式的训练方式，每个智能体训练一个Rule函数；第二种，中心化的训练方式，即所有智能体共同训练一个Rule函数。在我们的算法中，考虑到每个智能体的等价性与训练成本，智能体i与智能体j根据所做出的决策，具有相互独立且平等的特征，可以视为同一个智能体的不同动作序号。所以，我们采用中心化的训练方式，将所有智能体的采样数据合并作为DILP系统输入。根据前述的基础知识，Rule函数的输出为，。输出的是每个根据模板生成的命题所对应的valuation。输出的命题包括了P,N与B中的所有命题。为了与P,N中的标签数据求loss值，利用提取出与P,N数据集中相关的命题。根据二分类问题的特点，我们使用交叉熵作为loss函数：

我们需要对每个动作作为目标命题采样数据集进行训练。只有当loss收敛到某一阈值时，认为这个系统学习得到的相对应与此目标动作的规则是适用的，否则将进行下一个目标动作的训练或等待智能体的A-C函数被进一步更新。Rule函数训练结束后，我们需要求得Rule函数学习到的规则。由于权重表示了描述了对于目标命题，某规则子句在多大程度上准确定义了这个目标命题。由此，我们可将规则提取公式写为，

在实际训练中，往往这个方法提取的规则不够稳定。而我们通常会利用如下方法，

其中， 表示将取权重排序前m个的clauses。表示将取频率最高的n个clauses。取并集意味着将在多次的训练中合并规则结果。

下一阶段，系统将交替到maddpg训练。但与前一阶段不同的是，这一阶段，每个智能体的Actor函数还将受到规则的约束。具体来说，智能体在执行动作中，当观测信息满足规则约束的body时，那么将优先执行规则约束的动作。

首先，因为无法直接应用由clause构成的规则。我们需要将由clause表示的规则进行解码。由前面章节所述的逻辑解码器解码

在智能体与环境进行交互选择动作时，我们将ddpg的动作输出修改为，

其中，是智能体的Actor函数；使得智能体具有探索的性质；而函数prior则表示了的优先级将高于。构造了新的动作选择函数后，智能体就可在新规则下进行训练学习。

对于每个智能体的Actor函数参数更新与Critic函数参数更新，还是按照在multi-agent情况下，对ddpg的更新公式。即

MALPPG的框架如图3-1所示，整体而言，智能体对A-C函数与Rule函数的训练将交替进行。智能体在环境中交互得到数据后经A-C函数训练，在下一阶段执行交互动作时将由Rule函数接替训练。若收敛后得到规则，将在规则约束下进行下一阶段的采样与A-C函数更新。以此类推，在将训练结束前进行若干阶段的交替执行。在训练与执行中，Critic函数与Rule函数是中心化的训练。在执行中，是去中心化执行预测并执行动作。Rule函数与Critic函数将共享智能体与环境交互得到的序列对。

Decoder

Rule

a

o

a

o

Encoder

**图 3-1 MALPPG算法框架**

# **第四章 实验设计与验证**

## 4.1井字棋实验

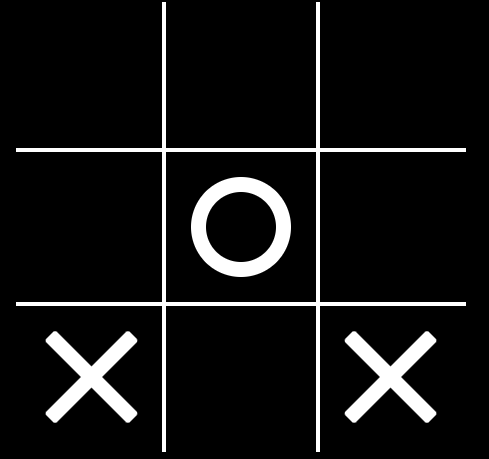
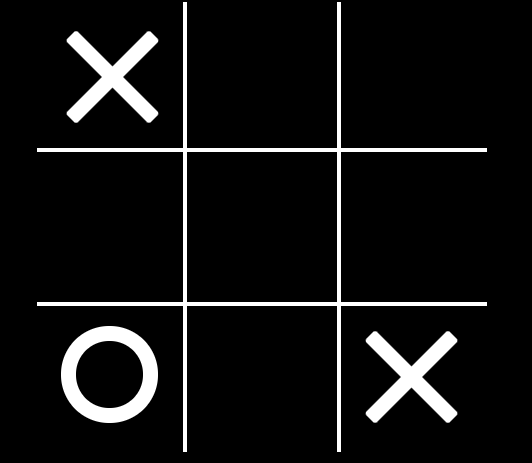
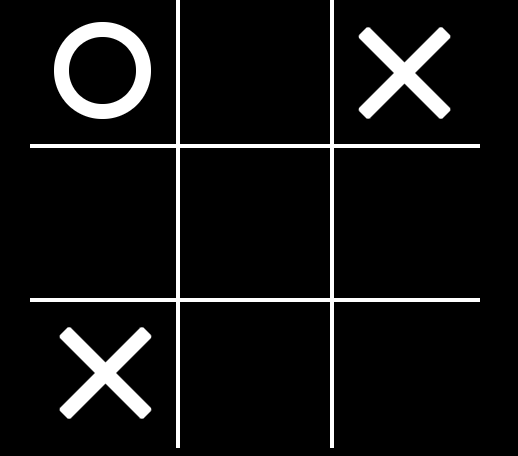
**井字棋实验**(tic-tac-toe)；井字棋是经典的多智能体强化学习实验。由两个智能体组成博弈的两方。两方在3\*3的棋盘上交替落子，直到有一方胜利或平局，胜利标准是存在一排或一列或一个斜边的三个子全为某一方的子。如图4-2(1)所示，

对于每个落点有-1,0,1三种状态，分别代表对方落子，空位，我方落子。井字棋的状态S是9个格子的状态组合。每个智能体的动作为将棋子放置在某一位置(X,Y)。奖励函数设计为，当某方最终胜利时获得奖励1，另一方记0。过程中的每一步奖励可由最终奖励用一个衰减系数递减。

在logic-encoder部分，我们将动作设为命题place(X,Y)；将状态S编码为mine(X,Y),opponent(X,Y),empty(X,Y)的组合。其中，mine(X,Y)表示为我方在(X,Y)占有一子；opponent (X,Y)表示为对方在(X,Y)占有一子；empty(X,Y)表示为在(X,Y)位置为空位。其次，为了增强子句的表达能力，我们引入了4个辅助命题加以判断学习。这四个命题分别为，Invented1(X,Y), Invented2(X,Y), Invented3(X), Invented4(Y)。

在这一部分中，我们仅验证两个智能体在博弈中，rule函数对井字棋实验能否学到有意义的规则。在井字棋的规则学习中，针对不同的初始状态，我们得到了如下的一些结果。“X”表示为对方占据，“O”表示我方落子。

初始状态分别如图(1)(2)(3)



1. (2) (3)

**图4-1 井字棋初始状态**

对于状态(1),学习到的规则是

对于规则的解读，可从目标命题是辅助命题的子句开始，逐步代入到更上层的子句中，最后归纳出指向目标动作的命题。在这组命题中，我们从下往上看，当坐标(X,X)是空位，且Y是X的后继时，可推出Invented4(X)。将此解释代入第一个子句，我们可以得到，落子的位置横坐标X满足对角位(X,X)是空位，且落子的位置横坐标X满足对角位(Y,Y)是空位，且X,Y均不是最大值。对于此初始状态，满足这几个条件的位置坐标是(2,2)。所以，这组规则能够比较好的归纳出，当对方是先手且即将在对角线上连满3子时，我方的怎么样的落子阻止对方不胜。

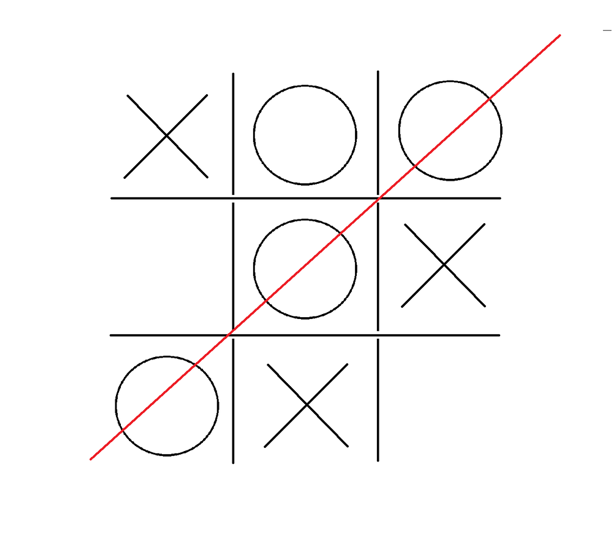
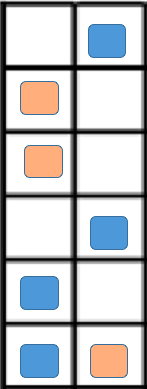
对于状态(2)，学习到的规则是

与前一组规则类似，这个子句规定了，X与Y坐标需满足分别在(X,X),(Y,Y)位上是空位。可以看出，它也成功学习到了如何“围堵”的规则。

对于状态(3)，学习到的规则是

对于这个子句，X坐标需满足X=Y+1，且Y坐标需满足在某第Y列有空位，同时，由前一个要求可知，Y不等于最大值。由初始状态，我们可以得到，满足要求的落子位置为(3,2)。

从井字棋实验可以得到，Rule函数可以有效的学习到一些可解释的规则。在这些规则指导下，可对对方进行围堵，阻止对方立刻获胜。由此验证了Rule函数的有效性。我们将在接下来的实验中，将对我们设计的整个算法验证是否能够提高多智能体学习算法的效果，同时得到可解释的规则。

Busy

NotBusy

1. 井字棋实验 (2)电动扶梯实验

**图4-2 实验环境示意图**

## 4.2电动扶梯实验

**电动扶梯实验**；电动扶梯实验模拟行人在乘坐电动扶梯时的场景。行人分为两类，繁忙的行人与不繁忙的行人，我们期望通过学习之后，繁忙的行人能够更快速的通过扶梯。在现实生活中，我们往往会不自觉的遵守这样一种规则。当扶梯上同时有繁忙与不繁忙的行人时，不忙的行人倾向于站在同一列并不行动，而由于另一列空位较多，不容易造成拥堵的风险，赶时间的行人将会移动到另一列上并前进。

扶梯有两列，扶梯高度可自定义，记为H；扶梯的每一格只能由至多一个智能体占据。扶梯上的智能体有两种可能的属性，繁忙状态(Busy)或非繁忙状态(NotBusy)，对应于现实中赶时间或者不赶时间的人。处于繁忙状态的人更倾向于更快的到达扶梯顶部，而处于非繁忙状态的人倾向于由扶梯带动向上走，而非自己走。在环境的一个时钟单位内，扶梯向上走一格，所有的智能体的高度会同时向上移动一格。在同一时钟单位内，所有智能体都会预测并执行一个动作。所以，在扶梯向上移动的过程中，所有智能体都会执行完一个动作。

每个智能体的动作空间一致，均为(up, switch, stay)三个动作。up代表智能体选择向上走一格；switch代表智能体选择朝扶梯另一边走。stay表示智能体选择不动。在reward的设计上，由于繁忙状态的智能体倾向于更快到达扶梯顶部。所以，当它选择动作up时，reward设为0，如果动作为stay或switch，那么将进行扣分，奖励为-1。同理，对于不繁忙状态的智能体，它们倾向于更省力的动作。则当它们选择up时，将得到奖励-1，反之，当它们执行动作，switch或stay时，它们的奖励为0。

对于整个训练过程，首先，对于每个智能体，不设游戏结束的标志，而代之以当时钟走过一定的时间，那么整个游戏结束，环境进行重置。其次，为了使每个agent对于繁忙与非繁忙状态都有记忆。我们设置，当每次到达扶梯顶部时，智能体自动回到扶梯第一格，但是繁忙与否的状态将会被反转。这样的设计也保证了，扶梯上每个智能体的数量一定，便于训练。

当agent1与agent2同时执行动作时，有可能会出现两个agent同时跳转到同一格的扶梯，从而导致冲突。所以，在该环境中，我们规定所有agents执行动作的顺序由上至下，从左至右执行。如果发生冲突，那么后执行动作的智能体将动作重置为stay，在replay buffer中记录为重置的动作。奖励与动作stay的奖励一致。其次，另一种冲突形式为最顶部的智能体无法转移到底部。原因是，由于扶梯的上升与智能体执行动作的同时性，处于顶部前两行的智能体都有可能转移到底部。则最多可能有4个agent转移到底部，由于底部只有两格，将出现冲突拥堵的情况。在这种情况下，我设置了等待队列用于智能体等待，并沿用之前的冲突处理方式，将动作重置为stay。

在logic-encoder部分，需将每个智能体的观察信息编码为基本命题。每个智能体的观察信息为扶梯上所有智能体的位置。将观察视野再限制到局部，规定如下：若智能体为繁忙状态，记命题为busy(X)，反之，记为notBusy(X)。若前方有智能体，那么记为front(X)；若后方有智能体，记为back(X)；若另一侧被其他智能体占据，记为side(X)；其中，X为(state, action)序列的序号。需要预测的目标命题为三个动作(up(X),switch(X),stay(X))。

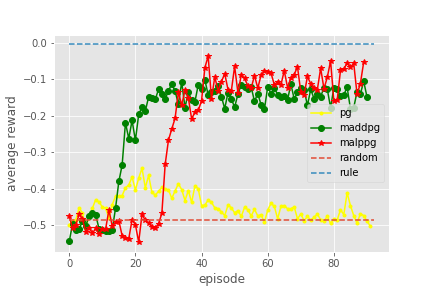
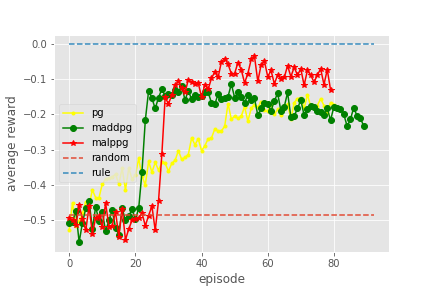
我们期望学到如前所述的社会规则，即我们称之为“左行右立”的规则。同时我们希望达到的目标是我们的算法能够尽可能的接近直接按照规则进行行动带来的效果。所以在本次实验中，我将设置一个rule-base的多智能体，人为的规定如下规则，并按照这样的规则行动。规则是，如果一个智能体处于繁忙状态，那么它将靠左；如果它已经在左边一列，且其前方并没有智能体，那么它将向前走；如果它的前方有其他智能体，那么它将静止不动。如果一个不繁忙状态的智能体，那么它将自动靠右，如果已经处于右方的一列，那么它将静止。除此之外，我们的Baseline算法有以下三个，1. random的智能体，将随机选择动作；2. policy gradient算法，一个针对于单智能体的强化学习算法，在这个算法中，我们将每个智能体视为独立的个体，相互之间并不通信，并将其他智能体视为环境的一部分，每个智能体训练一个自己的Actor函数，通过policy gradient进行优化；3. MADDPG算法，是在前一章节阐述的多智能体强化学习算法。每个智能体有自己的Actor函数和共同的Critic函数。我们将结果展示在图7与表1中。

**表1 不同算法的平均奖励对比**

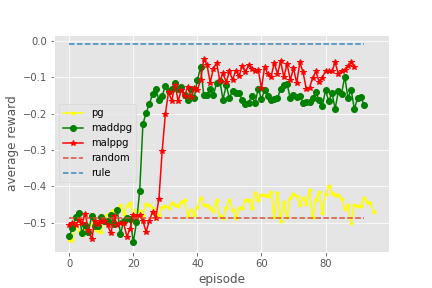
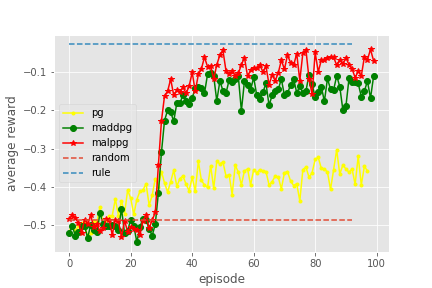
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | random | policy gradient | maddpg | malppg |
| 3agents/height10 | -0.484 | -0.175 | -0.174 | **-0.0857** |
| 5agents/height10 | -0.488 | -0.368 | -0.143 | **-0.0835** |
| 7agents/height10 | -0.492 | -0.491 | -0.183 | **-0.115** |
| 5agents/height8 | -0.486 | -0.454 | -0.143 | **-0.0987** |
| 5agents/height15 | -0.486 | -0.460 | -0.146 | **-0.0863** |

我们验证了在不同的扶梯高度下，不同数量的智能体在扶梯环境中的表现。如表2所示，对比三种算法的所有智能体的平均奖励。可以看出在所有情况下，malppg的效果均优于其他三种方法。对比次优的maddpg算法都有50%左右的提升。随着智能体数量与扶梯高度的比值提升，智能体之间越容易相互拥挤。可以看出随着比值提高，malppg的平均奖励也在减小，但是仍然优于其他几种算法。梯度策略方法在智能体的数量超过5的时，其平均奖励下降到了与随机算法相当。在智能体较少时，PG方法与MADDPG算法效果相当。PG算法是适用于单智能体的强化学习算法，所以在智能体较少时效果很好，而智能体较多有较多的相互作用时，效果急剧下降。MADDPG算法总体来说表现平稳，当随着智能体数量与扶梯高度的比值升高时，其平均奖励稍有降低，且在智能体数量较少的时候，MADDPG的平均奖励也有下滑。MADDPG表现出了多智能体算法对于处理多个智能体相互作用情况下的优点，在智能体相互拥挤的情况下表现远远优于PG算法，但智能体数量较少时却有所下降，其原因猜测是智能体相互作用不明显，其中心化的Critic函数作用有限，并且信息冗余。而malppg算法在除了7agents/height10的情况下有下降，在除此之外的情况之下，其平均奖励相差不大，表现出了较强的稳健性与泛化能力，即对环境改变的影响不敏感。原因是，malppg算法学习到了具有普适意义的逻辑规则，这样一种规则在大部分情况下都适用。例如，规则“左行右立”在无论电梯多长，人数多少的情况下都可大大减少通行时间。因为规则具有这样一定的抽象能力，从而抽象规则的适用范围也更广，从而实现了在多种环境改变下依然稳健的效果。

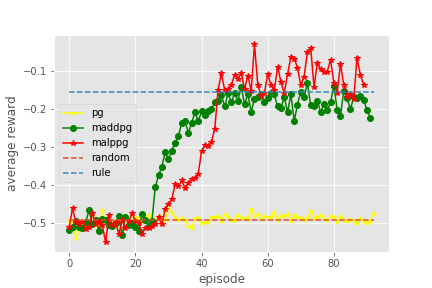
图4-3(1)-(5)是各种算法的平均奖励的收敛情况。从图中可以看出，malppg算法都越来越接近rule-base的智能体。在7agents/height10情况下，人为规则失效很明显，基于规则的智能体平均奖励远低于其他情况下，但这时的malppg算法却依然保持高奖励，甚至超过了rule-base的智能体，达到了对人为制定的规则的超越。PG算法在智能体较少的时候收敛到与MADDPG相差不多，却在智能体增加时越来越接近随机算法。MADDPG收敛情况都可以，却距离rule-base智能体有一定的差距。



(1) height: 10 agents: 3 (2) height: 8 agents: 5



(3) height: 10 agents: 5 (4) height: 15 agents: 5



(5) height: 10 agents: 7

**图4-3 不同环境设置下收敛结果比较**

malppg算法的另一个目的是期望能学习到可解释的有意义的规则，在malppg训练过程中，我们抽取了它得到的规则子句。有如下几条，

以上规则子句可以做出这样的解释。对于第一条，当一个智能体当前处于Busy状态时，那么它将向前走一步。对于第二条，当一个智能体的前方与后方同时有其他智能体时，则选择转换到扶梯的另一边。对于处于繁忙状态的智能体而言，毋庸置疑的是，选择在有可能的时候尽可能向上走是很节约时间的办法。此外，对于第二条规则，如果一个智能体处于Busy状态，那么对于它而言前后方都有智能体时，选择转换到另一边将有利于它再之后的步骤中向上走而不被阻挡。如果一个智能体处于NotBusy的状态，那么选择转换到另一边，如果它的后方智能体是Busy状态，那么将有利于它的后方智能体向前走一步以节省时间。所以，从这一条规则看来，malppg算法学到了一种尽量错开前后方智能体的规则，对应于社会规则中的“左行右立”，达到了我们预期的目的。

除此之外，社会规则还有另一种形式，即在某种状态下，禁止智能体的某一个或几个动作。对此，我们进行了实验，验证能否发现形式为禁止的规则。结果如下，

对上述规则的解读是，处于NotBusy状态下的智能体禁止向前走；前后方都有智能体时，智能体禁止不动；当智能体处于Busy状态，且后方有智能体时，禁止转换；上述的这些规则也都较为合理合法。不繁忙状态的智能体向前走将损失奖励，所以禁止；当前后方都有智能体时，为了避免拥堵，从而禁止不行动；当一个智能体是繁忙状态且后方有智能体时，它更应该倾向于向前走，既能得到奖励也避免堵塞，从而禁止转换。

# **第五章 总结与展望**

5.1 结论

受到人类智力对于规则归纳的启发。在本文中，我们为多智能体强化学习引入了可微分归纳逻辑编程框架，用于对智能体学习到的策略进行规则归纳，并由此反馈策略优化改进策略。我们首先在井字棋实验上验证了智能体对于规则归纳的能力。然后，我们还提出了新的实验设计，电动扶梯实验，模拟了人在走扶梯时的场景。在这个实验上，验证了智能体可以归纳出有意义的社会规则，并且策略效果优于maddpg等算法。在不同的环境设置上的表现上看，该算法也有较好的稳健性。

5.2 下一步的研究工作

本文的研究工作还有很多局限性。首先需要在更多，更复杂的场景的实验上验证理论的有效性。其次，该算法在更大规模的复杂问题上有很高的复杂度。所以在未来的研究工作中，我们将会引入适用于大规模逻辑推理的框架，在更多场景中验证。除此之外，多个智能体之间会有各种相关的关系，在许多场景下，多智能体之间的关系非常适合利用逻辑命题表示并加以训练学习，而一般的多智能体算法却在智能体之间的关系表示上有局限性，而在本次的研究工作中，还未对该方面进行更多的探索。在未来的研究工作中，我们会继续对此进行研究。

# **参考文献**

[1] Buşoniu, Lucian, Robert Babuška, and Bart De Schutter. "Multi-agent reinforcement learning: An overview." Innovations in multi-agent systems and applications-1. Springer, Berlin, Heidelberg, 2010. 183-221.

[2] Džeroski, Sašo, Luc De Raedt, and Hendrik Blockeel. "Relational reinforcement learning." International Conference on Inductive Logic Programming. Springer, Berlin, Heidelberg, 1998.

[3] Džeroski, Sašo, Luc De Raedt, and Kurt Driessens. "Relational reinforcement learning." Machine learning 43.1-2 (2001): 7-52.

[4] Evans, Richard, and Edward Grefenstette. "Learning explanatory rules from noisy data." Journal of Artificial Intelligence Research 61 (2018): 1-64.

[5] Dong, Honghua, et al. "Neural logic machines." arXiv preprint arXiv:1904.11694 (2019).

[6] Tacchetti, Andrea, et al. "Relational forward models for multi-agent learning." arXiv preprint arXiv:1809.11044 (2018).

[7] Lowe, Ryan, et al. "Multi-agent actor-critic for mixed cooperative-competitive environments." Advances in neural information processing systems. 2017.

[8] Zhang, Kaiqing, Zhuoran Yang, and Tamer Başar. "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms." arXiv preprint arXiv:1911.10635 (2019).

[9] Blockeel, Hendrik, and Luc De Raedt. "Top-down induction of first-order logical decision trees." Artificial intelligence 101.1-2 (1998): 285-297.

[10] Wen, Ying, et al. "Probabilistic recursive reasoning for multi-agent reinforcement learning." arXiv preprint arXiv:1901.09207 (2019).

[11] Jiang, Zhengyao, and Shan Luo. "Neural Logic Reinforcement Learning." arXiv preprint arXiv:1904.10729 (2019).

[12] Zambaldi, Vinicius, et al. "Relational deep reinforcement learning." arXiv preprint arXiv:1806.01830 (2018).

[13] Zambaldi, Vinicius, et al. "Deep reinforcement learning with relational inductive biases." International Conference on Learning Representations (2019).

[14] Serafini, Luciano, and Artur d'Avila Garcez. "Logic tensor networks: Deep learning and logical reasoning from data and knowledge." arXiv preprint arXiv:1606.04422 (2016).

[15] De Raedt, Luc, and Ingo Thon. "Probabilistic rule learning." International conference on inductive logic programming. Springer, Berlin, Heidelberg, 2010.

[16] Manhaeve, Robin, et al. "Deepproblog: Neural probabilistic logic programming." Advances in Neural Information Processing Systems. 2018.

[17] De Raedt, Luc, et al. "Inducing probabilistic relational rules from probabilistic examples." Twenty-Fourth International Joint Conference on Artificial Intelligence. 2015.

[18] <https://www.slideshare.net/deawooKim/deep-multiagent-reinforcement-learning>

[19] <https://www.cnblogs.com/initial-h/p/9429632.html>

# **致谢**