基于遗忘的反应式系统最弱充分条件研究

May 2, 2022

姓名：冯仁艳
导师：王以松
联合导师：Erman Acar[1]
学科专业：软件工程
研究方向：软件工程技术与人工智能

[1]LIACS, Leiden University, The Netherlands

目录

图 1: 系统故障引起的系列灾难现场

表 1: 由系统故障引起的重大事件概览

时间	事故原因	损失
1991 年	美国爱国者导弹系统舍入错误	28 名士兵死亡、100 人受伤等
1996 年	阿丽亚娜 5 火箭代码重用	火箭与其它卫星毁灭
1999 年	火星探测器用错度量单位	探测器坠毁并造成了 3.27 亿美元的损失
2011 年	温州 7.23 动车信号设备 在设计上存在严重的缺陷	动车脱节脱轨、多人失去生命

图 1: 系统故障引起的系列灾难现场

系

统正确对国防、太空勘测和交通运输至关重要。

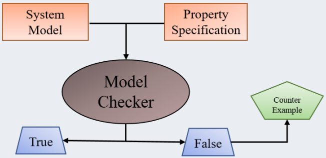
自动定理证明（Automated theorem proving）

令 ϕ_{imp} 和 ϕ_{spec} 分别表示系统模型和规范对应的时序逻辑公式：

- $\phi_{imp} \rightarrow \phi_{spec}$，或
- $\phi_{imp} \leftrightarrow \phi_{spec}$。

模型检测（Model Checking）

- $\mathscr{M} \models^{?} \phi_{spec}$.
- 反应式系统 (reactive system)：是指与环境有着持续不断交互的系统。
- 如何计算反应式系统的 WP?



研究背景和意义：简单的例子

Example (汽车制造企业模型)

一个汽车制造企业能够生产两种汽车：小轿车 (se) 和跑车 (sp)。每隔一段时间，该企业都会做一个生产决策 (d)，即：合理的生产计划。刚开始的时候，该企业做出了具有三个选择 (s) 的方案：

(1) 先生产足够的 se，然后在再生产 sp；

(2) 先生产足够的 sp，然后再生产 se；

(3) 同时生产 se 和 sp。

这一过程可以由图 2中的 Kripke 结构（带标签的状态转换图）$\mathscr{M} = (S, R, L)$ 形式化地展现出来，其中:

- $V = \{d, s, se, sp\}$ 为该工厂所需要考虑的原子命题集；
- $S = \{s_0, s_1, s_2, s_3, s_4\}$ 为状态空间；
- $R = \{(s_0, s_1), (s_1, s_2), (s_1, s_3), (s_1, s_4), (s_2, s_0), (s_3, s_0), (s_4, s_0)\}$ 为状态转换关系集；
- $L : S \to 2^V$ 为标签函数，具体地：$L(s_0) = \{d\}$、$L(s_1) = \{s\}$、$L(s_2) = \{se\}$、$L(s_3) = \{sp\}$ 和 $L(s_4) = \{se, sp\}$。

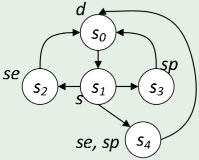


图 2: 汽车制造企业模型

假定，由于经济危机或者战略调整，导致该企业不能再生产跑车。这意味着所有规范和 Kripke 结构都不再需要考虑 sp 的，因此应该“移除”。

最强必要条件（SNC）和最弱充分条件（WSC）

SNC 和 WSC 分别用于描述给定理论下的最一般的结果（consequence）和最一般的诱因（abduction）[4]。满足下面两个条件的 φ 称为 q 在理论 Σ 下的 SNC:

(1) $\Sigma \models q \rightarrow \varphi$;

(2) 对任意 φ' 且 $\Sigma \models q \rightarrow \varphi'$，有 $\Sigma \models \varphi \rightarrow \varphi'$。

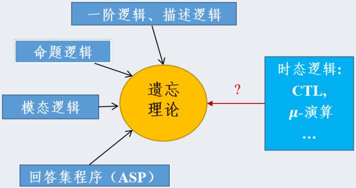
满足下面两个条件的 ψ 称为 q 在理论 Σ 下的（WSC）：

(1) $\Sigma \models \psi \rightarrow q$;

(2) 对任意 ψ' 且 $\Sigma \models \psi' \rightarrow q$，有 $\Sigma \models \psi' \rightarrow \psi$。

遗忘理论（Forgetting）

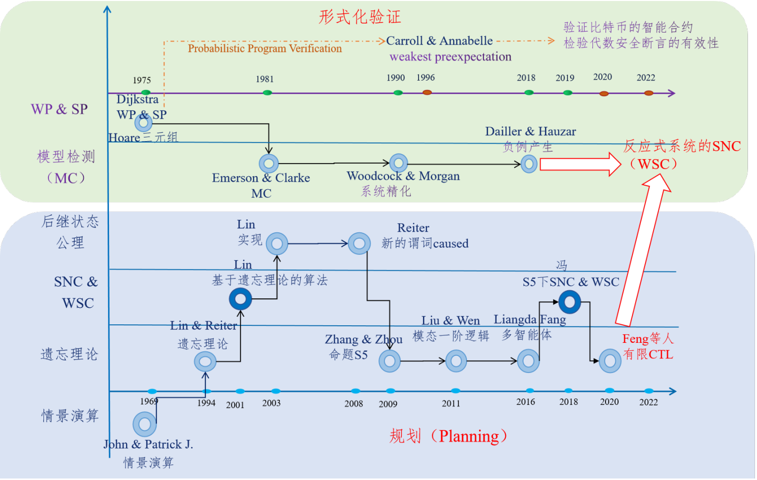
遗忘是一种从理论中抽取知识的技术 [5]，被用于规划[6,7] 和知识更新 中 [8]。非形式化地，对于逻辑语言 L 中的任意公式和原子集合，如果从该公式中遗忘掉该原子集合后得到的结果仍然在 L 中，则称遗忘存在，同时也称该公式和原子集合的遗忘存在。



目录

国内外研究现状

基于遗忘的反应式系统最弱充分条件研究
研究背景和意义
国内外研究现状
研究内容
背景知识
Kripke 结构
CTL 的语法和语义
μ-演算
CTL 和 mu-演算遗忘理论
CTL 遗忘理论
蒙特卡洛树搜索算法
基于手牌拆分的蒙特卡洛树搜索算法
实验比较结果
合作问题分析
算法缺点
结合卷积神经网络的蒙特卡洛树搜索



目录

研究内容

本论文研究反应式系统下，CTL 和 μ-演算的遗忘理论，并使用遗忘计算 WSC。具体为：

- CTL 和 μ-演算的遗忘理论
  - CTL 的遗忘理论
  - μ-演算的遗忘理论
- 遗忘理论在反应式系统的形式化验证和知识更新中的应用
  - 计算 WSC 和 SNC
  - 定义知识更新
- 计算 CTL 遗忘的算法
  - 基于模型的计算方法
  - 基于消解（resolution）的计算方法
  - 实现与实验分析

研究内容

本论文研究反应式系统下，CTL 和 μ-演算的遗忘理论，并使用遗忘计算 WSC。具体为：

- CTL 和 μ-演算的遗忘理论
  - CTL 的遗忘理论
  - μ-演算的遗忘理论
- 遗忘理论在反应式系统的形式化验证和知识更新中的应用
  - 计算 WSC 和 SNC
  - 定义知识更新
- 计算 CTL 遗忘的算法
  - 基于模型的计算方法
  - 基于消解（resolution）的计算方法
  - 实现与实验分析

研究内容

本论文研究反应式系统下，CTL 和 μ-演算的遗忘理论，并使用遗忘计算 WSC。具体为：

- CTL 和 μ-演算的遗忘理论
  - CTL 的遗忘理论
  - μ-演算的遗忘理论
- 遗忘理论在反应式系统的形式化验证和知识更新中的应用
  - 计算 WSC 和 SNC
  - 定义知识更新
- 计算 CTL 遗忘的算法
  - 基于模型的计算方法
  - 基于消解（resolution）的计算方法
  - 实现与实验分析

研究内容

本论文研究反应式系统下，CTL 和 μ-演算的遗忘理论，并使用遗忘计算 WSC。具体为：

- CTL 和 μ-演算的遗忘理论
  - CTL 的遗忘理论
  - μ-演算的遗忘理论
- 遗忘理论在反应式系统的形式化验证和知识更新中的应用
  - 计算 WSC 和 SNC
  - 定义知识更新
- 计算 CTL 遗忘的算法
  - 基于模型的计算方法
  - 基于消解（resolution）的计算方法
  - 实现与实验分析

研究内容

- CTL 和 μ-演算的遗忘理论
- 遗忘理论在反应式系统的形式化验证和知识更新中的应用
- 计算 *CTL* 遗忘的计算方法

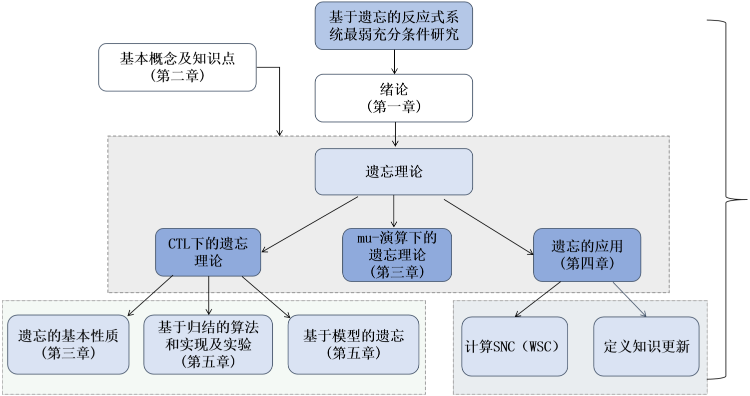


图 3: 文章组织结构示意图

目录

Kripke 结构

$\mathscr{A}$：原子命题的集合　　Ind：索引的集合

Definition (初始 Ind-Kripke 结构)

一个初始 Ind-Kripke 结构是一个五元组 $\mathscr{M} = (S, R, L, [\_], s_0)$，其中：

- S 是状态的非空集合，s_0 是 $\mathscr{M}$ 的初始状态（参见下文）；
- $R \subseteq S \times S$ 是状态转换函数，且对任意 $s \in S$，存在 $s' \in S$ 使得 $(s, s') \in R$；
- $L : S \to 2^{\mathscr{A}}$ 是一个标签函数；
- $[\_] : \mathrm{Ind} \to 2^{S \times S}$ 是一个函数，其使得对任意 $ind \in \mathrm{Ind}$，若 $s \in S$，则存在唯一一个 $s \in S$ 使得 $(s, s') \in [ind] \cap R$。

相关概念

- 初始 Kripke 结构 $\mathscr{M} = (S, R, L, s_0)$：从初始 Ind-Kripke 结构 $\mathscr{M}$ 中去掉 $[\_]$ 元素得到；
- Ind-Kripke 结构 $\mathscr{M} = (S, R, L, [\_])$：从初始 Ind-Kripke 结构 $\mathscr{M}$ 中去掉初始状态 s_0 得到；
- Kripke 结构 $\mathscr{M} = (S, R, L)$：从初始 Ind-Kripke 结构 $\mathscr{M}$ 中同时去掉 $[\_]$ 和 s_0 得到。

Kripke 结构

$\mathscr{A}$：原子命题的集合　　　Ind：索引的集合

Definition (初始 Ind-Kripke 结构)

一个初始 Ind-Kripke 结构是一个五元组 $\mathscr{M} = (S, R, L, [\_], s_0)$，其中：

- S 是状态的非空集合，s_0 是 $\mathscr{M}$ 的初始状态（参见下文）；
- $R \subseteq S \times S$ 是状态转换函数，且对任意 $s \in S$，存在 $s' \in S$ 使得 $(s, s') \in R$；
- $L : S \to 2^{\mathscr{A}}$ 是一个标签函数；
- $[\_] : \mathrm{Ind} \to 2^{S \times S}$ 是一个函数，其使得对任意 $ind \in \mathrm{Ind}$，若 $s \in S$，则存在唯一一个 $s \in S$ 使得 $(s, s') \in [ind] \cap R$。

相关概念

令 $\mathscr{M} = (S, R, L)$ 为 Kripke 结构，$\mathscr{M}' = (S, R, L, [\_])$ 为 Ind-Kripke 结构：

- 路径：$\mathscr{M}$ 上的路径是 $\mathscr{M}$ 上的状态构成的无限序列 $\pi = (s_0, s_1, s_2, \dots)$，且满足对任意 $j \geq 0$，$(s_j, s_{j+1}) \in R$；
- $s' \in \pi$：表示 s' 是路径 π 上的一个状态；π_s：表示以 s 为起点的 $\mathscr{M}$ 上的一条路径；
- 初始状态：如果对任意 $s' \in S$，都存在路径 π_s 使得 $s' \in \pi_s$，那么称 s 为初始状态；
- 索引路径：$\mathscr{M}'$ 上的一条索引路径 $\pi_s^{\langle ind \rangle}$（$ind \in \mathrm{Ind}$）是一条路径 $(s_0 (= s)$, s_1, s_2, $\dots)$，且对任意 $j \geq 0$，有 $(s_j, s_{j+1}) \in [ind]$。

Kripke 结构

$\mathscr{A}$：原子命题的集合　　　Ind：索引的集合

Definition (初始 Ind-Kripke 结构)

一个初始 Ind-Kripke 结构是一个五元组 $\mathcal{M} = (S, R, L, [\_], s_0)$，其中：

- S 是状态的非空集合，s_0 是 $\mathcal{M}$ 的初始状态（参见下文）；
- $R \subseteq S \times S$ 是状态转换函数，且对任意 $s \in S$，存在 $s' \in S$ 使得 $(s, s') \in R$；
- $L : S \to 2^{\mathscr{A}}$ 是一个标签函数；
- $[\_] : \mathrm{Ind} \to 2^{S \times S}$ 是一个函数，其使得对任意 $ind \in \mathrm{Ind}$，若 $s \in S$，则存在唯一一个 $s \in S$ 使得 $(s, s') \in [ind] \cap R$。

相关概念

一个(Ind-) 结构是一个二元组 $\mathcal{K} = (\mathcal{M}, s)$，其中 $\mathcal{M}$ 是一个初始 (Ind-)Kripke 结构，s 是 $\mathcal{M}$ 中的一个状态。如果 s 是 $\mathcal{M}$ 的初始状态，则称 $\mathcal{K}$ 是初始 (Ind-) 结构。
在这些结构中，（索引）路径这一概念可以类似地定义。

CTL 的语言符号

- 原子命题集 $\mathscr{A}$；　可数无限索引集合 Ind；　命题常量 **start**；
- 常量符号：$\top$ 和 $\bot$，分别表示“真”和“假”；
- 联结符号：$\vee$ 和 $\neg$，分别表示“析取”和“否定”；
- 路径量词：A、E 和 E_{ind}，分别表示“所有”、“存在”和“存在索引为 $ind \in \mathrm{Ind}$”的路径；
- 时序操作符：X、F、G、U 和 W，分别表示“下一个状态”、“将来某一个状态”、“将来所有状态”、“直到”和“除非”；
- 标点符号：“(”和“)”。

Definition (带索引的 CTL)

带索引的 CTL 公式的存在范式 (existential normal form, ENF)可以用巴科斯范式递归定义如下：

$$\phi ::= \mathbf{start} \mid \bot \mid p \mid \neg\phi \mid \phi \vee \phi \mid \mathrm{EX}\phi \mid \mathrm{EG}\phi \mid \mathrm{E}(\phi\ \mathrm{U}\ \phi) \mid \mathrm{E}_{\langle ind \rangle}\mathrm{X}\phi \mid \mathrm{E}_{\langle ind \rangle}\mathrm{G}\phi \mid \mathrm{E}_{\langle ind \rangle}(\phi \mathrm{U} \phi)$$

其中，$p \in \mathscr{A}$，$ind \in \mathrm{Ind}$。

没有索引和 **start** 的公式称为 CTL 公式。

Definition (带索引的 CTL)

带索引的 CTL 公式的存在范式 (existential normal form, ENF)可以用巴科斯范式递归定义如下：

$$\phi ::= \textsf{start} \mid \bot \mid p \mid \neg\phi \mid \phi\vee\phi \mid \mathrm{EX}\phi \mid \mathrm{EG}\phi \mid \mathrm{E}(\phi\ \mathrm{U}\ \phi) \mid \mathrm{E}_{\langle ind\rangle}\mathrm{X}\phi \mid \mathrm{E}_{\langle ind\rangle}\mathrm{G}\phi \mid \mathrm{E}_{\langle ind\rangle}(\phi\mathrm{U}\phi)$$

其中，$p\in\mathscr{A}$，$ind\in\mathrm{Ind}$。
没有索引和 **start** 的公式称为 CTL 公式。

CTL 中其它形式的公式可以通过如下定义（使用上述定义中的形式）得到：

$$\varphi\wedge\psi \overset{def}{=} \neg(\neg\varphi\vee\neg\psi) \tag{1}$$

$$\varphi\rightarrow\psi \overset{def}{=} \neg\varphi\vee\psi \tag{2}$$

$$\mathrm{A}(\varphi\mathrm{U}\psi) \overset{def}{=} \neg\mathrm{E}(\neg\psi\mathrm{U}(\neg\varphi\wedge\neg\psi))\wedge\neg\mathrm{EG}\neg\psi \tag{3}$$

$$\mathrm{A}(\varphi\mathrm{W}\psi) \overset{def}{=} \neg\mathrm{E}((\varphi\wedge\neg\psi)\mathrm{U}(\neg\varphi\wedge\neg\psi)) \tag{4}$$

$$\mathrm{E}(\varphi\mathrm{W}\psi) \overset{def}{=} \neg\mathrm{A}((\varphi\wedge\neg\psi)\mathrm{U}(\neg\varphi\wedge\neg\psi)) \tag{5}$$

$$\mathrm{AF}\varphi \overset{def}{=} \mathrm{A}(\top\mathrm{U}\psi) \tag{6}$$

$$\mathrm{EF}\varphi \overset{def}{=} \mathrm{E}(\top\mathrm{U}\psi) \tag{7}$$

$$\mathrm{AX}\varphi \overset{def}{=} \neg\mathrm{EX}\neg\varphi \tag{8}$$

$$\mathrm{AG}\varphi \overset{def}{=} \neg\mathrm{EF}\neg\varphi \tag{9}$$

CTL 的语法

Definition (带索引的 CTL)

带索引的 CTL 公式的存在范式 (existential normal form, ENF)可以用巴科斯范式递归定义如下：

$$\phi ::= \textsf{start} \mid \bot \mid p \mid \neg\phi \mid \phi\vee\phi \mid \mathrm{EX}\phi \mid \mathrm{EG}\phi \mid \mathrm{E}(\phi\ \mathrm{U}\ \phi) \mid \mathrm{E}_{\langle ind\rangle}\mathrm{X}\phi \mid \mathrm{E}_{\langle ind\rangle}\mathrm{G}\phi \mid \mathrm{E}_{\langle ind\rangle}(\phi\mathrm{U}\phi)$$

其中，$p\in\mathscr{A}$，$ind\in\mathrm{Ind}$。

没有索引和 **start** 的公式称为 CTL 公式。

符号优先级

带索引的 CTL 中各类符号的优先级如下，且从左到右优先级逐渐降低：

$$\neg, \mathrm{EX}, \mathrm{EF}, \mathrm{EG}, \mathrm{AX}, \mathrm{AF}, \mathrm{AG}, \mathrm{E}_{\langle ind\rangle}\mathrm{X}, \mathrm{E}_{\langle ind\rangle}\mathrm{F}, \mathrm{E}_{\langle ind\rangle}\mathrm{G}, \wedge, \vee, \mathrm{EU}, \mathrm{AU}, \mathrm{EW}, \mathrm{AW}, \mathrm{E}_{\langle ind\rangle}\mathrm{U}, \mathrm{E}_{\langle ind\rangle}\mathrm{W}, \rightarrow .$$

此外，给定一个不包含“$\rightarrow$”的公式 φ 和原子命题 p。在 φ 中，若 p 的前面有偶数个否定 $\neg$，则称 p 在 φ 中的出现为正出现，否则为负出现。若 φ 中所有 p 的出现都为正出现（或负出现），则称 φ 关于 p 是正的（或负的）。

Definition (带索引的 CTL 的语义)

给定公式 φ，初始 Ind-Kripke 结构 $\mathscr{M} = (S, R, L, [\_], s_0)$ 和状态 $s \in S$。$(\mathscr{M}, s)$ 与 φ 之间的可满足关系 $(\mathscr{M}, s) \models \varphi$ 定义如下：

- $(\mathscr{M}, s) \models \textbf{start}$ 当且仅当 $s = s_0$；
- $(\mathscr{M}, s) \not\models \bot$；
- $(\mathscr{M}, s) \models p$ 当且仅当 $p \in L(s)$；
- $(\mathscr{M}, s) \models \varphi_1 \vee \varphi_2$ 当且仅当 $(\mathscr{M}, s) \models \varphi_1$ 或 $(\mathscr{M}, s) \models \varphi_2$；
- $(\mathscr{M}, s) \models \neg\varphi$ 当且仅当 $(\mathscr{M}, s) \not\models \varphi$；
- $(\mathscr{M}, s) \models \mathrm{EX}\varphi$ 当且仅当存在 S 中的一个状态 s_1，使得 $(s, s_1) \in R$ 且 $(\mathscr{M}, s_1) \models \varphi$；
- $(\mathscr{M}, s) \models \mathrm{EG}\varphi$ 当且仅当存在 $\mathscr{M}$ 上的一条路径 $\pi_s = (s_1 = s, s_2, \dots)$，使得对每一个 $i \geq 1$ 都有 $(\mathscr{M}, s_i) \models \varphi$；
- $(\mathscr{M}, s) \models \mathrm{E}(\varphi \mathrm{U} \psi)$ 当且仅当存在 $\mathscr{M}$ 上的一条路径 $\pi_s = (s_1 = s, s_2, \dots)$，使得对某一个 $i \geq 1$ 有 $(\mathscr{M}, s_i) \models \psi$，且对任意 $1 \leq j < i$，有 $(\mathscr{M}, s_j) \models \varphi$；
- $(\mathscr{M}, s) \models \mathrm{E}_{\langle ind \rangle}\mathrm{X}\psi$ 当且仅当对索引路劲 $\pi_s^{\langle ind \rangle} = (s, s', \dots)$，有 $(\mathscr{M}, s') \models \psi$；
- $(\mathscr{M}, s) \models \mathrm{E}_{\langle ind \rangle}\mathrm{G}\psi$ 当且仅当对任意 $s' \in \pi_s^{\langle ind \rangle}$，$(\mathscr{M}, s') \models \psi$；
- $(\mathscr{M}, s) \models \mathrm{E}_{\langle ind \rangle}(\psi_1 \mathrm{U} \psi_2)$ 当且仅当存在 $\pi_s^{\langle ind \rangle} = (s = s_1, s_2, \dots)$ 中的 s_j $(1 \leq j)$ 使得 $(\mathscr{M}, s_j) \models \psi_2$ 且对任意 $s_k \in \pi_s^{\langle ind \rangle}$，若 $1 \leq k < j$，则 $(\mathscr{M}, s_k) \models \psi_1$。

CTL 的语义

记号

令 φ、φ_1 和 φ_2 为公式，这里列出文中出现的一些记号及其含义。

- $Mod(\varphi)$：公式 φ 的所有模型构成的集合；
- 可满足：如果 $Mod(\varphi) \neq \emptyset$，则称 φ 是可满足的；
- 逻辑蕴涵：若 $Mod(\varphi_1) \subseteq Mod(\varphi_2)$，则称 φ_1逻辑地蕴涵φ_2，记为 $\varphi_1 \models \varphi_2$；
- 逻辑等值：当 $\varphi_1 \models \varphi_2$ 且 $\varphi_2 \models \varphi_1$ 时，即 $Mod(\varphi_1) = Mod(\varphi_2)$，则称 φ_1 和 φ_2 为逻辑等值公式（简称为等值公式），记作 $\varphi_1 \equiv \varphi_2$；
- $Var(\varphi)$：出现在 φ 中的原子命题集；
- $\bigvee \Pi$ 和 $\bigwedge \Pi$ 分别表示有限集 Π 中公式的析取和合取；
- V-无关（V-irrelevant）：给定公式 φ 和原子命题集 V，如果存在一个公式 ψ 使得 $Var(\psi) \cap V = \emptyset$ 且 $\varphi \equiv \psi$，那么说 φ 与 V 中的原子命题无关，简称为V-无关（V-irrelevant），写作 $\mathrm{IR}(\varphi, V)$。
- 文字（literal）、子句（clause）、析取范式等跟经典命题情形中的定义一样。

$\mathrm{SNF}^{g}_{\mathrm{CTL}}$ 子句

具有下面几种形式的公式称为 CTL 全局子句分离范式（separated normal form with global clauses for CTL，$\mathrm{SNF}^{g}_{\mathrm{CTL}}$ 子句）[?, ?]：

$$
\begin{array}{ll}
\mathrm{AG}(\mathbf{start} \to \bigvee_{j=1}^{k} m_j) & \text{(初始句，initial clause)} \\
\mathrm{AG}(\top \to \bigvee_{j=1}^{k} m_j) & \text{(全局子句，global clause)} \\
\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \to \mathrm{AX} \bigvee_{j=1}^{k} m_j) & \text{(A-步子句，A-step clause)} \\
\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \to \mathrm{E}_{\langle ind \rangle}\mathrm{X} \bigvee_{j=1}^{k} m_j) & \text{(E-步子句，E-step clause)} \\
\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \to \mathrm{AF} l) & \text{(A-某时子句，A-sometime clause)} \\
\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \to \mathrm{E}_{\langle ind \rangle}\mathrm{F} l) & \text{(E-某时子句，E-sometime clause)}
\end{array}
$$

其中 k 和 n 都是大于 0 的常量，l_i（$1 \le i \le n$）、m_j（$1 \le j \le k$）和 l 都是文字且 $ind \in \mathrm{Ind}$。

转换规则

一个 CTL 公式 φ 可以通过下表中的规则转换为一个 $\mathrm{SNF}^{g}_{\mathrm{CTL}}$ 子句集，记为 T_φ。

表 1: 转换规则

$$\textbf{Trans(1)}\ \frac{q\to \mathrm{E}T\varphi}{q\to \mathrm{E}_{\langle ind\rangle}T\varphi};\qquad \textbf{Trans(2)}\ \frac{q\to \mathrm{E}(\varphi_1\mathrm{U}\varphi_2)}{q\to \mathrm{E}_{\langle ind\rangle}(\varphi_1\mathrm{U}\varphi_2)};\qquad \textbf{Trans(3)}\ \frac{q\to\varphi_1\wedge\varphi_2}{q\to\varphi_1, q\to\varphi_2};$$

$$\textbf{Trans(4)}\ \frac{q\to\varphi_1\vee\varphi_2\ (\text{如果 }\varphi_2\text{ 不是子句})}{q\to\varphi_1\vee p, p\to\varphi_2};\qquad \textbf{Trans(5)}\ \frac{q\to D}{\top\to\neg q\vee D};\ \frac{q\to\bot}{\top\to\neg q};\ \frac{q\to\top}{\{\}}$$

$$\textbf{Trans(6)}\ \frac{q\to Q\mathrm{X}\varphi\ (\text{如果 }\varphi\text{ 不是子句})}{q\to Q\mathrm{X}p, p\to\varphi};\qquad \textbf{Trans(7)}\ \frac{q\to Q\mathrm{F}\varphi\ (\text{如果 }\varphi\text{ 不是文字})}{q\to Q\mathrm{F}p, p\to\varphi};$$

$$\textbf{Trans(8)}\ \frac{q\to Q(\varphi_1\mathrm{U}\varphi_2)\ (\text{如果 }\varphi_2\text{ 不是文字})}{q\to Q(\varphi_1\mathrm{U}p), p\to\varphi_2};\qquad \textbf{Trans(10)}\ \frac{q\to Q\mathrm{G}\varphi}{q\to p, p\to\varphi, p\to Q\mathrm{X}p};$$

$$\textbf{Trans(9)}\ \frac{q\to Q(\varphi_1\mathrm{W}\varphi_2)\ (\text{如果 }\varphi_2\text{ 不是文字})}{q\to Q(\varphi_1\mathrm{W}p), p\to\varphi_2};$$

$$\textbf{Trans(11)}\ \frac{q\to Q(\varphi\mathrm{U}l)}{q\to l\vee p, p\to\varphi, p\to Q\mathrm{X}(l\vee p), q\to Q\mathrm{F}l};\qquad \textbf{Trans(12)}\ \frac{q\to Q(\varphi\mathrm{W}l)}{q\to l\vee p, p\to\varphi, p\to Q\mathrm{X}(l\vee p)}.$$

其中，$T\in\{\mathrm{X},\mathrm{G},\mathrm{F}\}$，$ind$ 是规则中引入的新索引且 $Q\in\{\mathrm{A},\mathrm{E}_{\langle ind\rangle}\}$；$q$ 是一个原子命题，l 是一个文字，D 是文字的析取（即子句），p 是新的原子命题；φ，φ_1，和 φ_2 都是 CTL 公式。

基于遗忘的反应式系统最弱充分条件研究
研究背景和意义
国内外研究现状
研究内容
背景知识
Kripke 结构
CTL 的语法和语义
μ-演算
CTL 和 mu-演算遗忘理论
CTL 遗忘理论
蒙特卡洛树搜索算法
基于手牌拆分的蒙特卡洛树搜索算法
实验比较结果
合作问题分析
算法缺点
结合卷积神经网络的蒙特卡洛树搜索

转换步骤

给定一个 CTL 公式 φ，将其转换为一个 $\mathrm{SNF}^g_{\mathrm{CTL}}$ 字句集合的主要步骤如下：

(1) 将公式 CTL 转换为其 NNF [a] 形式，记为 $nnf(\varphi)$；

(2) 使用等值公式化简 $nnf(\varphi)$，得到 $simp(nnf(\varphi))$；

(3) 使用转换规则将 $\{\mathrm{AG}(\mathbf{start} \to z), \mathrm{AG}(z \to simp(nnf(\varphi)))\}$ 化简为 $\mathrm{SNF}^g_{\mathrm{CTL}}$ 子句集 T_φ，其中 T_φ 由如下导出（derivation）序列生成：

$$T_0 = \{\mathrm{AG}(\mathbf{start} \to p), \mathrm{AG}(p \to \mathbf{simp}(\mathbf{nnf}(\varphi)))\}, T_1, \ldots, T_n = T_\varphi$$

使得

- p 是一个新的原子命题，即：$p \notin \{\mathbf{start}\} \cup Var(\varphi)$；
- $T_{t+1} = (T_t - \{\psi\}) \cup R_t$ $(t \geq 0)$，其中 ψ 为 T_t 中的非 $\mathrm{SNF}^g_{\mathrm{CTL}}$ 子句，且 R_t 是使用一条匹配的归则作用到 ψ 上得到的结果集；
- T_n 中的每个公式都是 $\mathrm{SNF}^g_{\mathrm{CTL}}$ 子句形式。

[a] 对于给定的公式 φ，其否定范式（negation normal form, NNF）是将否定联结词 “$\neg$” 的出现通过上述定义变化到只出现在原子命题之前的形式。

Example

令 $\varphi = \neg\mathrm{AF}p \wedge \mathrm{AF}(p \wedge \top)$，下面给出将 φ 转换为 $\mathrm{SNF}^{g}_{\mathrm{CTL}}$ 子句集的详细步骤。

(1) 将公式 φ 转换为其 NNF 形式：$\mathrm{EG}\neg p \wedge \mathrm{AF}(p \wedge \top)$；

(2) 化简 (1) 中的公式为：$\mathrm{EG}\neg p \wedge \mathrm{AF}p$；

(3) 使用转换规则转换 $\{\mathrm{AG}(\mathbf{start} \to z), \mathrm{AG}(z \to (\mathrm{EG}\neg p \wedge \mathrm{AF}p))\}$，详细步骤如下：

1. $\mathbf{start} \to z$	
2. $z \to \mathrm{EG}\neg p \wedge \mathrm{AF}p$	
3. $z \to \mathrm{EG}\neg p$	$(2, \mathbf{Trans(3)})$
4. $z \to \mathrm{AF}p$	$(2, \mathbf{Trans(3)})$
5. $z \to \mathrm{E}_{\langle 1 \rangle}\mathrm{G}\neg p$	$(3, \mathbf{Trans(1)})$
6. $z \to x$	$(5, \mathbf{Trans(10)})$
7. $x \to \neg l$	$(5, \mathbf{Trans(10)})$
8. $x \to \mathrm{E}_{\langle 1 \rangle}\mathrm{G}x$	$(5, \mathbf{Trans(10)})$
9. $\top \to \neg z \vee x$	$(6, \mathbf{Trans(5)})$
10. $\top \to \neg x \vee \neg p$	$(7, \mathbf{Trans(5)})$

因此，得到的 φ 对应的 $\mathrm{SNF}^{g}_{\mathrm{CTL}}$ 子句集为：

1. $\mathbf{start} \to z$ 2. $z \to \mathrm{AF}p$ 3. $x \to \mathrm{E}_{\langle 1 \rangle}\mathrm{G}x$ 4. $\top \to \neg z \vee x$ 5. $\top \to \neg x \vee \neg p$.

μ-演算的语法

不动点符号：μ 和 ν，分别表示“最小不动点”和“最大不动点”。
$\mathscr{V}$：变元符号的可数集。
各类符号之间的优先级如下（从左到右优先级逐渐变低）：

$$\neg \quad \text{EX} \quad \text{AX} \quad \wedge \quad \vee \quad \mu \quad \nu.$$

Definition (μ-演算公式)

μ-演算公式（简称为 μ-公式或公式）递归定义如下：

$$\varphi ::= p \mid X \mid \neg\varphi \mid \varphi \vee \varphi \mid \text{AX}\varphi \mid \nu X.\varphi$$

其中 $p \in \mathscr{A}$ 且 $X \in \mathscr{V}$。

约定

- 公式 $\nu X.\varphi$ 中的 X 总是正出现在 φ 中，即：φ 中 X 的每一次出现之前都有偶数个否定符号“$\neg$”；
- 称出现在 $\mu X.\varphi$ 和 $\nu X.\varphi$ 中的变元 X 是受约束的（bound），且受约束的变元称为约束变元，不受约束的变元称为自由变元；
- 文字（literal）：原子命题和变元符号及其各自的否定；
- 这里所谈到的公式指的是取名恰当的（well-named）、受保护（guarded）的 μ-公式。

μ-演算的语法

不动点符号：μ 和 ν，分别表示“最小不动点”和“最大不动点”。
$\mathscr{V}$：变元符号的可数集。
各类符号之间的优先级如下（从左到右优先级逐渐变低）：

$$\neg \quad \mathrm{EX} \quad \mathrm{AX} \quad \wedge \quad \vee \quad \mu \quad \nu.$$

Definition (μ-演算公式)

μ-演算公式（简称为 μ-公式或公式）递归定义如下：

$$\varphi ::= p \mid X \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathrm{AX}\varphi \mid \nu X.\varphi$$

其中 $p \in \mathscr{A}$ 且 $X \in \mathscr{V}$。

注意

在 μ-演算公式的定义中，通常考虑动作集 Act 和一组与 $a \in Act$ 相关的模态词“$\langle a \rangle$” [?, ?, ?]。为了方便，本文考虑公式里只有一个动作的情形，但是本文的结论可以扩展到一般的情形。此时，模态词中的动作 a 可以省略，且公式 $\mathrm{EX}\varphi$（或 $\mathrm{AX}\varphi$）与公式 $\langle a \rangle\varphi$（或 $[a]\varphi$）[?] 相同。

μ-演算的语义

Definition

给定 μ-演算公式 φ、Kripke 结构 $\mathscr{M}=(S,R,L,r)$ 和一个从 $\mathscr{V}$ 中的变量到 $\mathscr{M}$ 中状态的赋值函数 $v:\mathscr{V}\rightarrow 2^S$。公式在 $\mathscr{M}$ 和 v 上的解释是 S 的一个子集 $\|\varphi\|_v^{\mathscr{M}}$（如果在上下文中 $\mathscr{M}$ 是明确的，则可以省去上标）：

$$\begin{aligned}
&\|p\|_v=\{s \mid p\in L(s)\},\\
&\|X\|_v=v(X),\\
&\|\varphi_1\vee\varphi_2\|_v=\|\varphi_1\|_v\cup\|\varphi_2\|_v,\\
&\|\mathrm{AX}\varphi\|_v=\{s \mid \forall s'.(s,s')\in R\Rightarrow s'\in\|\varphi\|_v\},\\
&\|\nu X.\varphi\|_v=\bigcup\{S'\subseteq S \mid S'\subseteq\|\varphi\|_{v[X:=S']}\}.
\end{aligned}$$

其中，$v[X:=S']$ 是一个赋值函数，它除了 $v[X:=S'](X)=S'$ 之外，和 v 完全相同。

注意：虽然这里的 Kripke 结构不要求其二元关系是完全的，但是这里的情况更加一般化，其结论也能推广到二元关系是完全的情形。

μ-演算的语义

Definition

给定 μ-演算公式 φ、Kripke 结构 $\mathscr{M}=(S,R,L,r)$ 和一个从 $\mathscr{V}$ 中的变量到 $\mathscr{M}$ 中状态的赋值函数 $v:\mathscr{V}\to 2^S$。公式在 $\mathscr{M}$ 和 v 上的解释是 S 的一个子集 $\|\varphi\|_v^{\mathscr{M}}$（如果在上下文中 $\mathscr{M}$ 是明确的，则可以省去上标）：

$$\|p\|_v=\{s\mid p\in L(s)\},$$
$$\|X\|_v=v(X),$$
$$\|\varphi_1\vee\varphi_2\|_v=\|\varphi_1\|_v\cup\|\varphi_2\|_v,$$
$$\|\mathrm{AX}\varphi\|_v=\{s\mid \forall s'.(s,s')\in R\Rightarrow s'\in\|\varphi\|_v\},$$
$$\|\nu X.\varphi\|_v=\bigcup\{S'\subseteq S\mid S'\subseteq\|\varphi\|_{v[X:=S']}\}.$$

其中，$v[X:=S']$ 是一个赋值函数，它除了 $v[X:=S'](X)=S'$ 之外，和 v 完全相同。

记号和约定

- 赋值：由 $\mathscr{M}$、其赋值函数 v 和 $\mathscr{M}$ 上的状态 s 构成的三元组 $(\mathscr{M},s,v)$ 称为为赋值（当 s 为 $\mathscr{M}$ 的根时，$(\mathscr{M},s,v)$ 简写为 $(\mathscr{M},v)$，也称其为一个赋值）；
- 若 $s\in\|\varphi\|_v$，则称 s “满足” φ，记为 $(\mathscr{M},s,v)\models\varphi$；
- $Mod(\varphi)$：φ 的模型的集合，即 $Mod(\varphi)=\{(\mathscr{M},v)\mid(\mathscr{M},r,v)\models\varphi\}$（当 φ 为 μ-句子时，也可简写为 $Mod(\varphi)=\{\mathscr{M}\mid(\mathscr{M},r,v)\models\varphi\}$）；
- 当公式 φ 为 μ-句子时，可以将赋值函数 v 省略。

μ-公式的析取范式

μ-演算的覆盖 -语法

在覆盖 -语法语法中，用覆盖操作（cover operator）集替换上述 μ-公式的定义中的 EX，且满足

- $Cover(\emptyset)$ 是公式；
- 对任意 $n \geq 1$，若 $\varphi_1, \dots, \varphi_n$ 是公式，则 $Cover(\varphi_1, \dots, \varphi_n)$ 是公式。

Definition

对于给定的初始结构 $\mathscr{M} = (S, R, L, r)$ 和赋值函数 v：

- $(\mathscr{M}, r, v) \models Cover(\emptyset)$ 当且仅当 r 没有任何的后继状态；
- $(\mathscr{M}, s, v) \models Cover(\varphi_1, \dots, \varphi_n)$ 当且仅当
  - 对任意 $i = 1, \dots, n$，存在 $(s, t) \in R$ 使得 $(\mathscr{M}, t, v) \models \varphi_i$；
  - 对任意 $(s, t) \in R$，存在 $i \in \{1, \dots, n\}$ 使得 $(\mathscr{M}, t, v) \models \varphi_i$。

μ-公式的析取范式

μ-演算的覆盖 -语法

在覆盖 -语法语法中，用覆盖操作（cover operator）集替换上述 μ-公式的定义中的 EX，且满足

- $Cover(\emptyset)$ 是公式；
- 对任意 $n \geq 1$，若 $\varphi_1, \ldots, \varphi_n$ 是公式，则 $Cover(\varphi_1, \ldots, \varphi_n)$ 是公式。

等价关系

覆盖 -语法与上述 μ-演算的语法是等价的 [?]，且 $Cover$ 公式与 EX 公式之间可以通过下面的等式转换：

$$Cover(\varphi_1, \ldots, \varphi_n) \Leftrightarrow \mathrm{EX}\varphi_1 \wedge \cdots \wedge \mathrm{EX}\varphi_n \wedge \mathrm{AX}(\varphi \vee \cdots \vee \varphi_n),$$

反之，

$$\mathrm{EX}\varphi \Leftrightarrow Cover(\varphi, \top).$$

μ-公式的析取范式

μ-演算的覆盖 -语法

在覆盖 -语法语法中，用覆盖操作（cover operator）集替换上述 μ-公式的定义中的 EX，且满足

- $Cover(\emptyset)$ 是公式；
- 对任意 $n \geq 1$，若 $\varphi_1, \ldots, \varphi_n$ 是公式，则 $Cover(\varphi_1, \ldots, \varphi_n)$ 是公式。

Definition (析取 μ-公式 [?])

析取 μ-公式集 $\mathscr{F}_d$ 是包含 $\top$、$\bot$ 和不矛盾的文字的合取且封闭于下面几条规则的最小集合:

(1) 析取式（disjunctions）：若 $\alpha, \beta \in \mathscr{F}_d$，则 $\alpha \vee \beta \in \mathscr{F}_d$；

(2) 特殊合取式（special conjunctions）：若 $\varphi_1, \ldots, \varphi_n \in \mathscr{F}_d$ 且 δ 为不矛盾的文字的合取，则 $\delta \wedge Cover(\varphi_1, \ldots, \varphi_n) \in \mathscr{F}_d$；

(3) 不动点操作（fixpoint operators）：若 $\varphi \in \mathscr{F}_d$，且对任意公式 ψ，φ 不含有形如 $X \wedge \psi$ 的子公式，则 $\mu X.\varphi$ 和 $\nu X.\varphi$ 都在 $\mathscr{F}_d$ 中。

目录

CTL 遗忘理论

Definition (V-互模拟)

给定原子命题集 $V \subseteq \mathcal{A}$、索引集合 $I \subseteq \mathrm{Ind}$ 和初始 Ind-结构 $\mathcal{M}_i = (S_i, R_i, L_i, [\_]_i, s_0^i)$ $(i = 1, 2)$。$\mathcal{B}_V \subseteq S_1 \times S_2$ 为二元关系，对任意 $s_1 \in S_1$ 和 $s_2 \in S_2$，若 $(s_1, s_2) \in \mathcal{B}_V$，则：

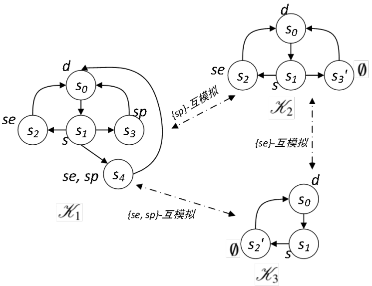
(i) $L_1(s_1) - V = L_2(s_2) - V$;

(ii) $\forall r_1 \in S_1$，若 $(s_1, r_1) \in R_1$，则 $\exists r_2 \in S_2$ 使得 $(s_2, r_2) \in R_2$ 和 $(r_1, r_2) \in \mathcal{B}_V$;

(iii) $\forall r_2 \in S_2$，若 $(s_2, r_2) \in R_2$，则 $\exists r_1 \in S_1$ 使得 $(s_1, r_1) \in R_1$ 和 $(r_1, r_2) \in \mathcal{B}_V$。

那么，称 $\mathcal{B}_V$ 是 $\mathcal{M}_1$ 和 $\mathcal{M}_2$ 之间的一个 V-互模拟关系。

- 结构互模拟：若 $\mathcal{M}_1$ 和 $\mathcal{M}_2$ 之间存在一个 V-互模拟关系 $\mathcal{B}_V$ 使得 $(s_1, s_2) \in \mathcal{B}_V$，则称两个 Ind-结构 $\mathcal{K}_1 = (\mathcal{M}_1, s_1)$ 和 $\mathcal{K}_2 = (\mathcal{M}_2, s_2)$ 是 V-互模拟的，记为 $\mathcal{K}_1 \leftrightarrow_V \mathcal{K}_2$;
- 路径互模拟：令 $i \in \{1, 2\}$，$\pi_i = (s_{i,1}, s_{i,2}, \ldots)$ 为 $\mathcal{M}_i$ 上的路径，若对任意 $j \geq 1$ 都有 $\mathcal{K}_{1,j} \leftrightarrow_V \mathcal{K}_{2,j}$，则称这两条路径是 V-互模拟的，记为 $\pi_1 \leftrightarrow_V \pi_2$，其中 $\mathcal{K}_{i,j} = (\mathcal{M}_i, s_{i,j})$。



Definition (V-互模拟)

给定原子命题集 $V \subseteq \mathscr{A}$、索引集合 $I \subseteq \mathrm{Ind}$ 和初始 Ind-结构 $\mathscr{M}_i = (S_i, R_i, L_i, [\_]_i, s_0^i)$ $(i=1,2)$。$\mathscr{B}_V \subseteq S_1 \times S_2$ 为二元关系，对任意 $s_1 \in S_1$ 和 $s_2 \in S_2$，若 $(s_1, s_2) \in \mathscr{B}_V$，则：

(i) $L_1(s_1) - V = L_2(s_2) - V$;

(ii) $\forall r_1 \in S_1$，若 $(s_1, r_1) \in R_1$，则 $\exists r_2 \in S_2$ 使得 $(s_2, r_2) \in R_2$ 和 $(r_1, r_2) \in \mathscr{B}_V$;

(iii) $\forall r_2 \in S_2$，若 $(s_2, r_2) \in R_2$，则 $\exists r_1 \in S_1$ 使得 $(s_1, r_1) \in R_1$ 和 $(r_1, r_2) \in \mathscr{B}_V$。

那么，称 $\mathscr{B}_V$ 是 $\mathscr{M}_1$ 和 $\mathscr{M}_2$ 之间的一个 V-互模拟关系。

定义

给定集合 $V_i \subseteq \mathscr{A}$、状态 s_i'、路径 π_i' 和 Ind-结构 $\mathscr{K}_j = (\mathscr{M}_j, s_j)$，其中 $i=1,2$，$j=1,2,3$。如果 $\mathscr{K}_1 \leftrightarrow_{V_1} \mathscr{K}_2$ 且 $\mathscr{K}_2 \leftrightarrow_{V_2} \mathscr{K}_3$，则：

(i) $\mathscr{K}_1 \leftrightarrow_{V_1 \cup V_2} \mathscr{K}_3$;

(ii) 若 $V_1 \subseteq V_2$，则 $\mathscr{K}_1 \leftrightarrow_{V_2} \mathscr{K}_2$;

(iii) $s_1' \leftrightarrow_{V_i} s_2'$ $(i=1,2)$ 蕴涵 $s_1' \leftrightarrow_{V_1 \cup V_2} s_2'$;

(iv) $\pi_1' \leftrightarrow_{V_i} \pi_2'$ $(i=1,2)$ 蕴涵 $\pi_1' \leftrightarrow_{V_1 \cup V_2} \pi_2'$;

(v) 对 $\mathscr{M}_1$ 上的每条路径 π_{s_1}，存在 $\mathscr{M}_2$ 上的一条路径 π_{s_2} 使得 $\pi_{s_1} \leftrightarrow_{V_1} \pi_{s_2}$，反之也成立。

Definition (V-互模拟)

给定原子命题集 $V \subseteq \mathscr{A}$、索引集合 $I \subseteq \mathrm{Ind}$ 和初始 Ind-结构 $\mathscr{M}_i = (S_i, R_i, L_i, [\_]_i, s_0^i)$ $(i = 1,2)$。$\mathscr{B}_V \subseteq S_1 \times S_2$ 为二元关系，对任意 $s_1 \in S_1$ 和 $s_2 \in S_2$，若 $(s_1, s_2) \in \mathscr{B}_V$，则：

(i) $L_1(s_1) - V = L_2(s_2) - V$;

(ii) $\forall r_1 \in S_1$，若 $(s_1, r_1) \in R_1$，则 $\exists r_2 \in S_2$ 使得 $(s_2, r_2) \in R_2$ 和 $(r_1, r_2) \in \mathscr{B}_V$;

(iii) $\forall r_2 \in S_2$，若 $(s_2, r_2) \in R_2$，则 $\exists r_1 \in S_1$ 使得 $(s_1, r_1) \in R_1$ 和 $(r_1, r_2) \in \mathscr{B}_V$。

那么，称 $\mathscr{B}_V$ 是 $\mathscr{M}_1$ 和 $\mathscr{M}_2$ 之间的一个 V-互模拟关系。

Theorem

令 $V \subseteq \mathscr{A}$ 是原子命题集，$\mathscr{K}_i$ $(i = 1,2)$ 是两个具有 V-互模拟关系的 Ind-结构，即：$\mathscr{K}_1 \leftrightarrow_V \mathscr{K}_2$。若 Φ 是一个 CTL 公式且 $\mathrm{IR}(\Phi, V)$，则有 $\mathscr{K}_1 \models \Phi$ 当且仅当 $\mathscr{K}_2 \models \Phi$。

Definition (互模拟等价，bisimilar equivalence)

给定原子命题集 $V \subseteq \mathscr{A}$，公式 φ 和 ψ。若对任意 $\mathscr{K} \models \varphi$，都存在一个 $\mathscr{K}' \models \psi$，使得 $\mathscr{K} \leftrightarrow_V \mathscr{K}'$；且对任意 $\mathscr{K}' \models \psi$，都存在一个 $\mathscr{K} \models \varphi$，使得 $\mathscr{K} \leftrightarrow_V \mathscr{K}'$，则称公式 φ 和 ψ 是 V-互模拟等价的（bisimilar equivalence），记为 $\varphi \equiv_V \psi$。

Lemma

对任意 $V \subseteq \mathscr{A}$，$\leftrightarrow_V$ 和 $\equiv_V$ 为等价关系。

互模拟等价

Definition (互模拟等价，bisimilar equivalence)

给定原子命题集 $V \subseteq \mathcal{A}$，公式 φ 和 ψ。若对任意 $\mathcal{K} \models \varphi$，都存在一个 $\mathcal{K}' \models \psi$，使得 $\mathcal{K} \leftrightarrow_V \mathcal{K}'$；且对任意 $\mathcal{K}' \models \psi$，都存在一个 $\mathcal{K} \models \varphi$，使得 $\mathcal{K} \leftrightarrow_V \mathcal{K}'$，则称公式 φ 和 ψ 是<u>V-互模拟等价的（bisimilar equivalence）</u>，记为 $\varphi \equiv_V \psi$。

Corollary

令 V、V_1、V_2 为 $\mathcal{A}$ 的子集，φ 和 ψ 为公式。

(i) 若 $\varphi \equiv \psi$，则 $\varphi \equiv_V \psi$。

(ii) 若 φ 和 ψ 不包括索引，且 $\varphi \equiv_\emptyset \psi$，则 $\varphi \equiv \psi$。

(iii) 若 $\varphi \equiv_{V_i} \psi$ $(i = 1,2)$，则 $\varphi \equiv_{V_1 \cup V_2} \psi$。

(iv) 若 $\varphi \equiv_{V_1} \psi$ 和 $V_1 \subseteq V_2$，则 $\varphi \equiv_{V_2} \psi$。

互模拟等价

Definition (互模拟等价，bisimilar equivalence)

给定原子命题集 $V \subseteq \mathscr{A}$，公式 φ 和 ψ。若对任意 $\mathscr{K} \models \varphi$，都存在一个 $\mathscr{K}' \models \psi$，使得 $\mathscr{K} \leftrightarrow_V \mathscr{K}'$；且对任意 $\mathscr{K}' \models \psi$，都存在一个 $\mathscr{K} \models \varphi$，使得 $\mathscr{K} \leftrightarrow_V \mathscr{K}'$，则称公式 φ 和 ψ 是<u>V-互模拟等价的（bisimilar equivalence）</u>，记为 $\varphi \equiv_V \psi$。

定义

令 φ 为一个 CTL 公式。则 $\varphi \equiv_U T_\varphi$，其中 $T_\varphi = \mathrm{SNF}^g_{\mathrm{CTL}}(\varphi)$ 和 $U = Var(T_\varphi) - Var(\varphi)$。

基于“斗地主”规则的拆分 Split 算法

function Split(F,X)

  if $X=\varnothing$

    then return F;

  else

    令 $\Gamma(X)=\{Y_1,Y_2,\ldots,Y_n\}-\{\varnothing\}$;

    Return $(Split(F+Y_1,X-Y_1)\bigcup Split(F+Y_2,X-Y_2)\bigcup\cdots\bigcup Split(F+Y_n,X-Y_n))$

    ,其中 $F+Y_i=\{z\bigcup\{Y_i\}\mid z\in F\}$;

基于“斗地主”规则的手牌较小拆分算法 (LessSplit)

Input: handpoker X

Output: handpoker smaller split set MS

function LessSplit(X)

$F = \{\varnothing\}$， $MS = \varnothing$;

S=Split(F,X);

$L_{min} = \mathrm{Min}(\{\mathrm{len}(z) \mid z \in S\})$;

for each s in S:

if $\mathrm{len}(s) \le (L_{min} + 3)$

then $MS = MS \cup s$;

return MS;

- 玩家的手牌为：34556789LB。对玩家手牌进行拆分，所有拆分结果为：
  - s_1={3，4，5，5，6，7，8，9，L，B}
  - s_2={3，4，5，5，6，7，8，9，LB}
  - s_3={3，4，55，6，7，8，9，L，B}
  - s_4={3，4，55，6，7，8，9，LB}
  - s_5={3，4，5，LB，56789}
  - s_6={3，4，5，L，B，56789}
  - s_7={3，5，9，LB，45678}
  - s_8={3，5，9，L，B，45678}
  - s_9={3，5，LB，456789}
  - s_{10}={3，5，L，B，456789}
  - s_{11}={5，8，9，LB，34567}
  - s_{12}={5，8，9，L，B，34567}
  - s_{13}={5，9，LB，345678}
  - s_{14}={5，9，L，B，345678}
  - s_{15}={5，LB，3456789}
  - s_{16}={5，L，B，3456789}

手牌拆分算法实例

- 玩家的手牌为：34556789LB。对玩家手牌进行拆分，所有拆分结果为：
  - s_1={3，4，5，5，6，7，8，9，L，B}
  - s_2={3，4，5，5，6，7，8，9，LB}
  - s_3={3，4，55，6，7，8，9，L，B}
  - s_4={3，4，55，6，7，8，9，LB}
  - s_5={3，4，5，LB，56789}
  - s_6={3，4，5，L，B，56789}
  - s_7={3，5，9，LB，45678}
  - s_8={3，5，9，L，B，45678}
  - s_9={3，5，LB，456789}
  - s_{10}={3，5，L，B，456789}
  - s_{11}={5，8，9，LB，34567}
  - s_{12}={5，8，9，L，B，34567}
  - s_{13}={5，9，LB，345678}
  - s_{14}={5，9，L，B，345678}
  - s_{15}={5，LB，3456789}
  - s_{16}={5，L，B，3456789}

手牌拆分算法实例

- 玩家的手牌为：34556789LB。对玩家手牌进行拆分，所有拆分结果为：
  - s_1={3，4，5，5，6，7，8，9，L，B}
  - s_2={3，4，5，5，6，7，8，9，LB}
  - s_3={3，4，55，6，7，8，9，L，B}
  - s_4={3，4，55，6，7，8，9，LB}
  - s_5={3，4，5，LB，56789}
  - s_6={3，4，5，L，B，56789}
  - s_7={3，5，9，LB，45678}
  - s_8={3，5，9，L，B，45678}
  - s_9={3，5，LB，456789}
  - s_{10}={3，5，L，B，456789}
  - s_{11}={5，8，9，LB，34567}
  - s_{12}={5，8，9，L，B，34567}
  - s_{13}={5，9，LB，345678}
  - s_{14}={5，9，L，B，345678}
  - s_{15}={5，LB，3456789}
  - s_{16}={5，L，B，3456789}

手牌拆分算法实例

- 玩家的手牌为：34556789LB。对玩家手牌进行拆分，所有拆分结果为：
  - s_1={3，4，5，5，6，7，8，9，L，B}
  - s_2={3，4，5，5，6，7，8，9，LB}
  - s_3={3，4，55，6，7，8，9，L，B}
  - s_4={3，4，55，6，7，8，9，LB}
  - s_5={3，4，5，LB，56789}
  - s_6={3，4，5，L，B，56789}
  - s_7={3，5，9，LB，45678}
  - s_8={3，5，9，L，B，45678}
  - s_9={3，5，LB，456789}
  - s_{10}={3，5，L，B，456789}
  - s_{11}={5，8，9，LB，34567}
  - s_{12}={5，8，9，L，B，34567}
  - s_{13}={5，9，LB，345678}
  - s_{14}={5，9，L，B，345678}
  - s_{15}={5，LB，3456789}
  - s_{16}={5，L，B，3456789}

手牌拆分算法实例

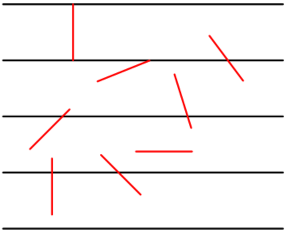
- 玩家的手牌为：34556789LB。对玩家手牌进行拆分，所有拆分结果为：
  - s_1={3，4，5，5，**6**，**7**，8，9，L，B}
  - s_2={3，4，5，5，**6**，**7**，8，9，LB}
  - s_3={3，4，**55**，**6**，**7**，8，9，L，B}
  - s_4={3，4，**55**，**6**，**7**，8，9，LB}
  - s_5={3，4，5，LB，56789}
  - s_6={3，4，5，L，B，56789}
  - s_7={3，5，9，LB，45678}
  - s_8={3，5，9，L，B，45678}
  - s_9={3，5，LB，456789}
  - s_{10}={3，5，L，B，456789}
  - s_{11}={5，8，9，LB，34567}
  - s_{12}={5，8，9，L，B，34567}
  - s_{13}={5，9，LB，345678}
  - s_{14}={5，9，L，B，345678}
  - s_{15}={5，LB，3456789}
  - s_{16}={5，L，B，3456789}

蒙特卡洛树搜索算法

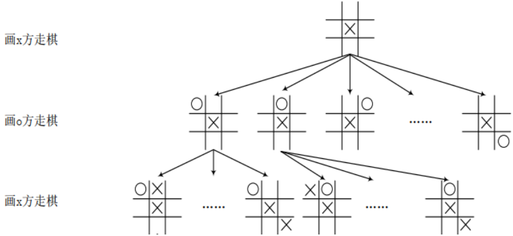
蒙特卡洛抽样法

为了求解问题，首先建立一个概率模型或随机过程，使它的参数或数字特征等于问题的解, 然后通过对模型、过程的观察或者抽样试验来计算这些参数、数字特征，最后给出所求解的近似值。

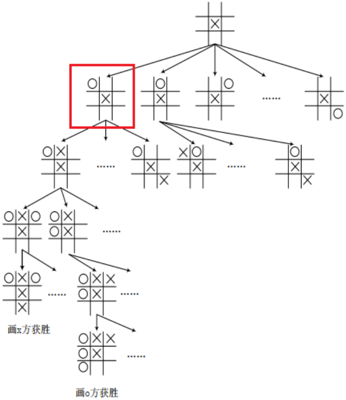
如:Buffon's needle problem



博弈树搜索算法: 将初始状态和所有可能的后续状态通过直接先后关系连接在一起形成博弈树



蒙特卡洛树搜索算法



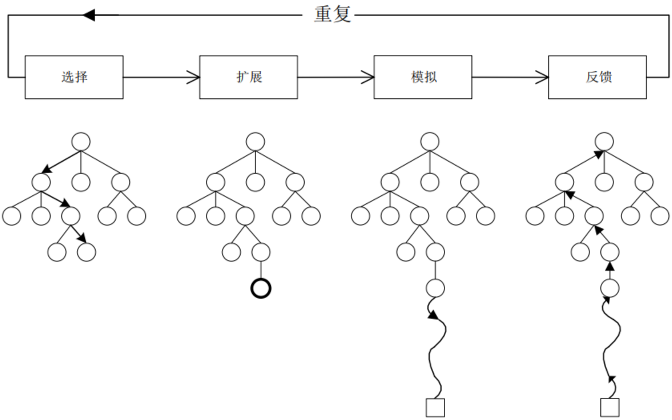
思想

利用经验平均来代替随机变量的期望。如在博弈状态 s 时期望值为 $v_\pi(s)$，一般难以通过计算直接求出该值，但是可以通过蒙特卡洛方法获得一系列收益 $G_1(s),\cdots,G_n(s)$. 根据大数定律，当 n 趋于无穷大时，抽样收益的均值趋近于期望值。定义 $v(s)$ 为系列收益的平均值，即

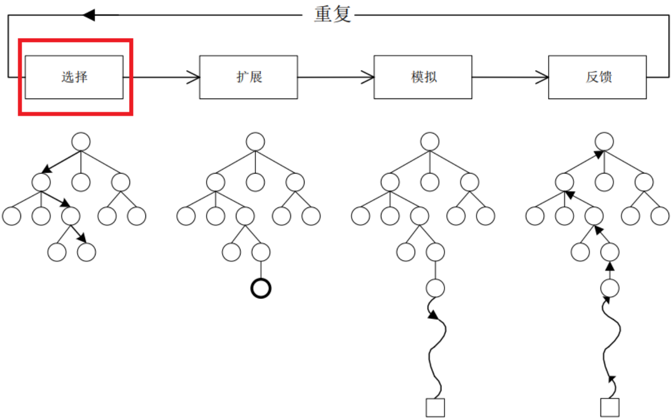
$$v(s)=\frac{G_1(s)+\cdots+G_n(s)}{n}$$

当 $n\to\infty$ 时，$v(s)\to v_\pi(s)$

蒙特卡洛树搜索算法过程



基于手牌拆分的蒙特卡洛树搜索算法过程



与规则算法 (RB) 比较

规则算法 (RB)

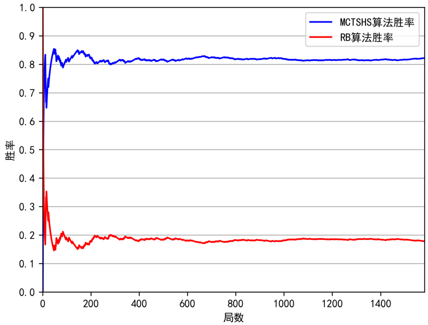
该算法分为主动策略和被动策略两种。主动策略中若上轮玩家取得主动权，那本轮该玩家可根据自己手牌主动选择出牌类型，而不需要考虑其他玩家的出牌类型；被动策略中玩家需要考虑本轮其他玩家的出牌，被动选择跟牌类型。

不区分角色比较结果：

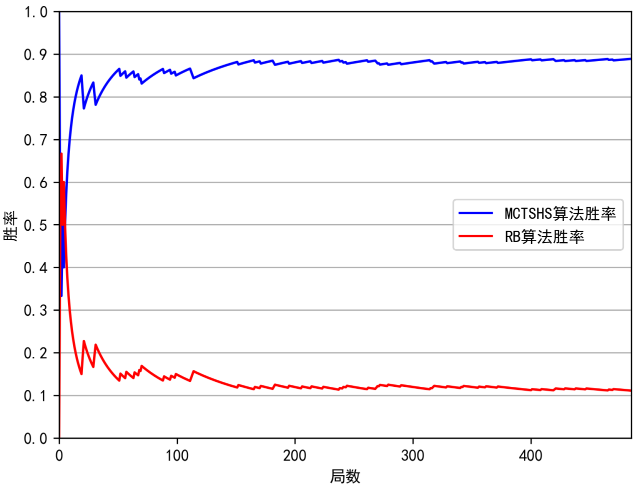
规则算法 (RB)

该算法分为主动策略和被动策略两种。主动策略中若上轮玩家取得主动权，那本轮该玩家可根据自己手牌主动选择出牌类型，而不需要考虑其他玩家的出牌类型；被动策略中玩家需要考虑本轮其他玩家的出牌，被动选择跟牌类型。

不区分角色比较结果：

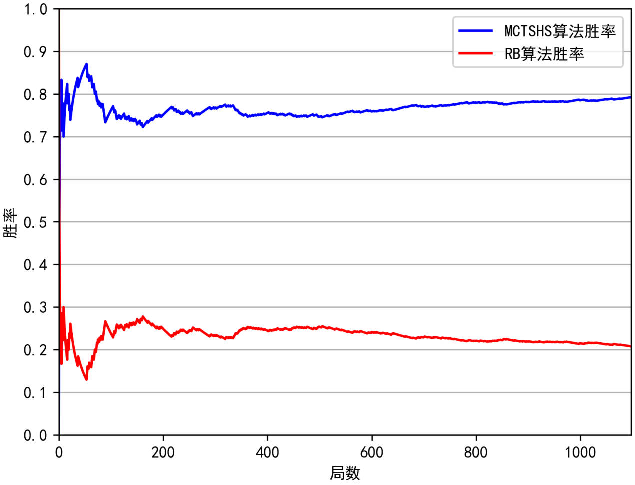


地主 MCTSHS 对农民 RB：



与规则算法 (RB) 比较

农民 MCTSHS 对地主 RB：

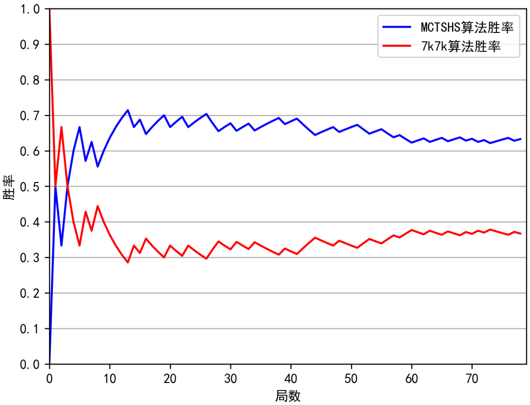


基于遗忘的反应式系统最弱充分条件研究
研究背景和意义
国内外研究现状
研究内容
背景知识
Kripke 结构
CTL 的语法和语义
μ-演算
CTL 和 mu-演算遗忘理论
CTL 遗忘理论
蒙特卡洛树搜索算法
基于手牌拆分的蒙特卡洛树搜索算法
实验比较结果
合作问题分析
算法缺点
结合卷积神经网络的蒙特卡洛树搜索

7k7k 算法

该算法为北京迦游网络科技有限公司开发的“斗地主”智能算法。

不区分角色比较结果：



合作问题分析：

算法事例	合作问题算法事例
当前玩家手牌	334567QQKKA2B
当前玩家位置	1（其中 0 表示地主，1 表示农民一，2 表示农民二）
当前玩家角色	农民一
地主已出牌	339922789JQK666JJL
当前玩家已出牌	55TTB
农民二已出牌	77AA89TJQKA44488
本轮中地主出牌	L
博弈过程	0|33, 1|55, 2|77 ; 0|99, 1|TT, 2|AA ; 0|22, 1|pass, 2|pass ; 0|7 89TJQK, 1|pass, 2|89TJQKA ; 0|pass, 1|pass, 2|4448 ; 0|66 6JJ, 1|pass, 2|pass ; **0|L, 1|B**, 2|pass ; 0|pass, **1|3,** 2|2.

算法缺点

算法缺点：

- 每次决策思考时间过长
- 已搜索到的决策未能充分利用

算法缺点

算法缺点：

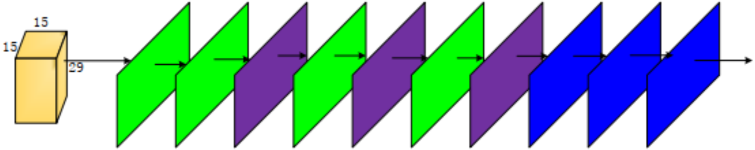
- 每次决策思考时间过长
- 已搜索到的决策未能充分利用

改进算法!!

目录

MCM 模型
- 手牌拆分的蒙特卡洛树搜索模块
- CNN 策略学习模块
- 改善策略模块

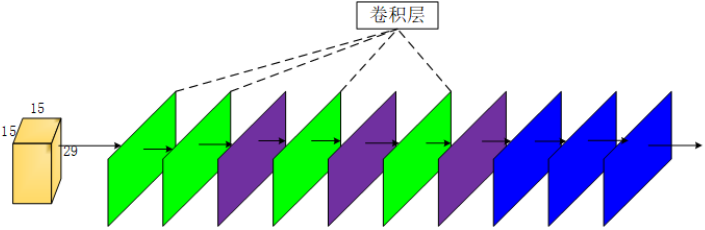
CNN 策略学习模块:



结合卷积神经网络的蒙特卡洛树搜索模型 MCM

MCM 模型
- 手牌拆分的蒙特卡洛树搜索模块
- CNN 策略学习模块
- 改善策略模块

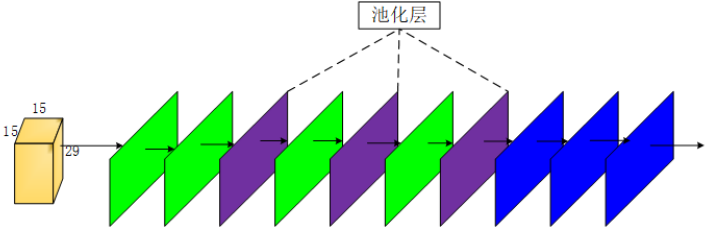
CNN 策略学习模块:



MCM 模型

- 手牌拆分的蒙特卡洛树搜索模块
- CNN 策略学习模块
- 改善策略模块

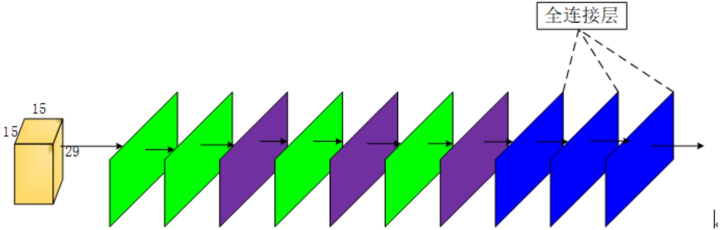
CNN 策略学习模块:



MCM 模型

- 手牌拆分的蒙特卡洛树搜索模块
- CNN 策略学习模块
- 改善策略模块

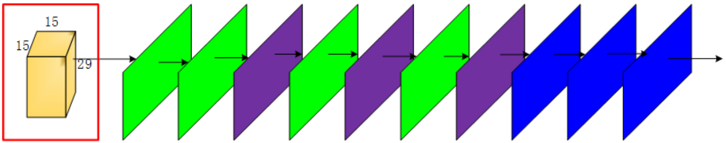
CNN 策略学习模块:



结合卷积神经网络的蒙特卡洛树搜索模型 MCM

MCM 模型
- 手牌拆分的蒙特卡洛树搜索模块
- CNN 策略学习模块
- 改善策略模块

CNN 策略学习模块:



CNN 策略学习模块 ——输入表示

- X 维度：表示 15 种扑克
- Y 维度：
  - 0-3(下标) 表示扑克的张数
  - 4-13(下标) 表示扑克是否参与组成出牌类型
  - 14(下标) 表示该出牌是否为地主玩家。
- Z 维度:

- X 维度：表示 15 种扑克
- Y 维度：
  - 0-3(下标) 表示扑克的张数
  - 4-13(下标) 表示扑克是否参与组成出牌类型
  - 14(下标) 表示该出牌是否为地主玩家。
- Z 维度:

CNN 策略学习模块 ——输入表示

- X 维度：表示 15 种扑克
- Y 维度：
  - 0-3(下标) 表示扑克的张数
  - 4-13(下标) 表示扑克是否参与组成出牌类型
  - 14(下标) 表示该出牌是否为地主玩家。
- Z 维度:

CNN 策略学习模块 —输入表示

- X 维度：表示 15 种扑克
- Y 维度：
  - 0-3(下标) 表示扑克的张数
  - 4-13(下标) 表示扑克是否参与组成出牌类型
  - 14(下标) 表示该出牌是否为地主玩家。
- Z 维度:

CNN 策略学习模块 ——输入表示

- X 维度：表示 15 种扑克
- Y 维度：
  - 0-3(下标) 表示扑克的张数
  - 4-13(下标) 表示扑克是否参与组成出牌类型
  - 14(下标) 表示该出牌是否为地主玩家。
- Z 维度:

- X 维度：表示 15 种扑克
- Y 维度：
  - 0-3(下标) 表示扑克的张数
  - 4-13(下标) 表示扑克是否参与组成出牌类型
  - 14(下标) 表示该出牌是否为地主玩家。
- Z 维度:

维度	意义
0-2	分别表示当前玩家、下家、上家 8 轮之前所有出牌的张数，不区分每次出牌类型。
3-5	分别表示当前玩家、下家、上家前第 8 轮的出牌
6-8	分别表示当前玩家、下家、上家前第 7 轮的出牌
9-11	分别表示当前玩家、下家、上家前第 6 轮的出牌
12-14	分别表示当前玩家、下家、上家前第 5 轮的出牌
15-17	分别表示当前玩家、下家、上家前第 4 轮的出牌
18-20	分别表示当前玩家、下家、上家前第 3 轮的出牌
21-23	分别表示当前玩家、下家、上家前第 2 轮的出牌
24-26	分别表示当前玩家、下家、上家前第 1 轮的出牌
27	表示本轮玩家出牌
28	表示当前玩家的手牌，只记录牌张数，不区分牌型

CNN 策略学习模块

- 学习样本处理
  - 将 MCTSHS 决策结果的数据进行去重
  - 随机打乱去重后的样本顺序，并从打乱的样本中，随机选择 90% 的样本组成训练集，10% 作为测试集

- 学习样本处理
  - 将 MCTSHS 决策结果的数据进行去重
  - 随机打乱去重后的样本顺序，并从打乱的样本中，随机选择 90% 的样本组成训练集，10% 作为测试集

MCTSHS 学习的部分历史数据记录:

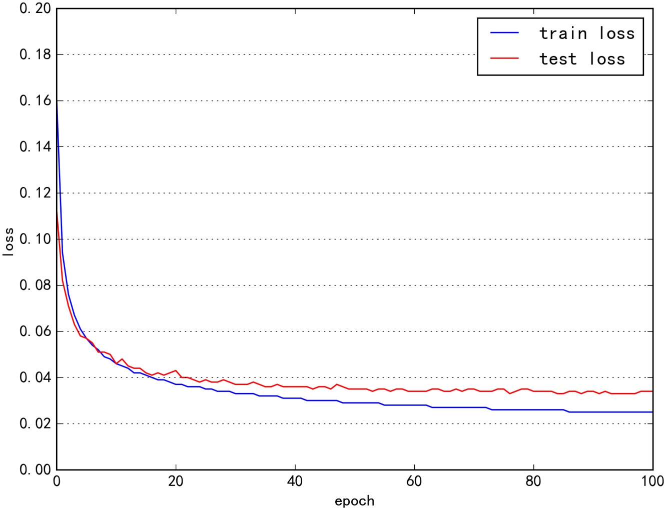
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 456789 789TJQ 9TJQKA 0 0 556677 **TTJJQQ** 46TTJJQQKKAAA2 15874 11639 0.733259

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 456789 789TJQ 9TJQKA 0 0 556677 **JJQQKK** 46TTJJQQKKAAA2 2087 1393 0.667613

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 456789 789TJQ 9TJQKA 0 0 556677 **QQKKAA** 46TTJJQQKKAAA2 997 395 0.397019

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 456789 789TJQ 9TJQKA 0 0 556677 **0** 46TTJJQQKKAAA2 1042 413 0.396792

CNN 网络学习策略损失变化图:



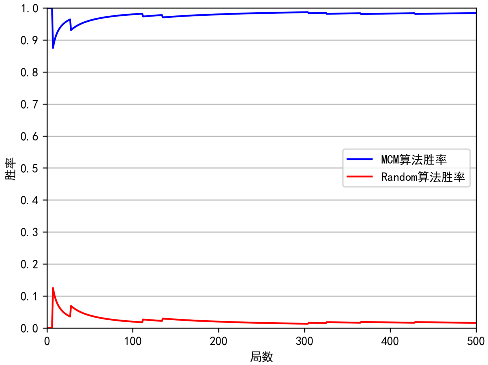
实验比较设定

- 地主、农民使用不同的决策算法，其中农民一、农民二均使用农民的决策算法
- 地主、农民一、农民二使用不同的决策算法
- 地主、农民一、农民二使用不同的决策算法进行相同牌局比较

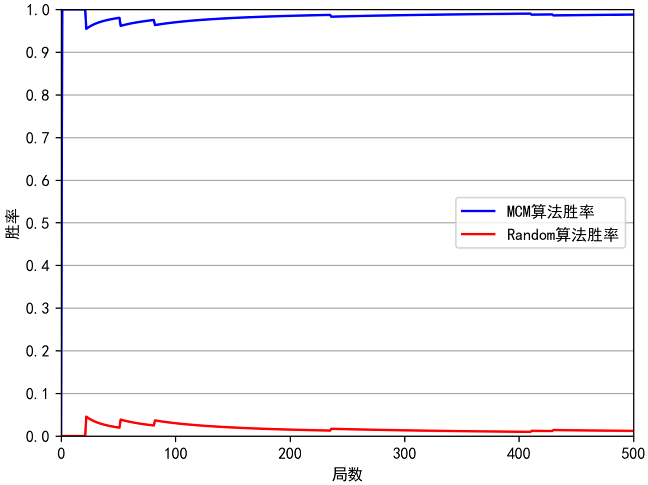
随机算法 (Random) 介绍

思路为：根据玩家的手牌、本轮其它玩家出牌等信息按照博弈规则计算出当前状态下玩家可能的所有出牌，并从中随机选择一种可能出牌作为本轮的最终出牌。

地主 MCM 对农民 Random 的胜率变化图:



农民 MCM 对地主 Random 的胜率变化图:

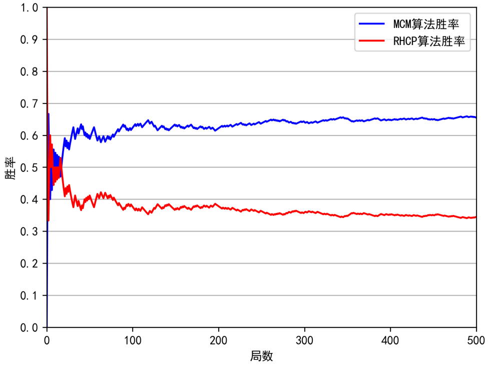


实验结果 ——与 RHCP 算法比较

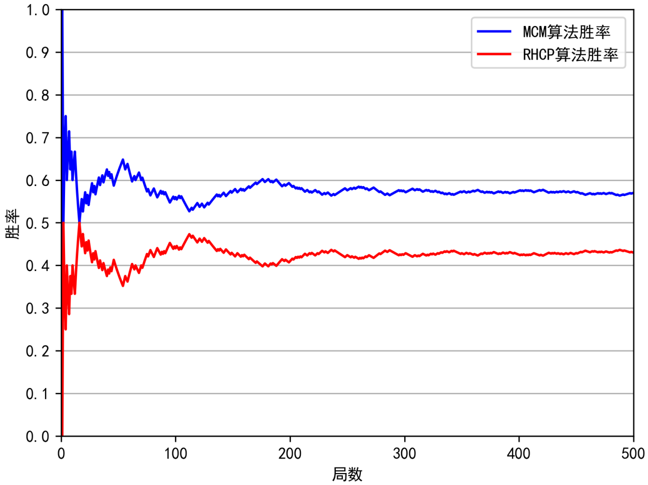
RHCP 算法介绍

该算法引入手牌剩余价值的概念，其总体思路是将手牌按照“斗地主”规则进行不同的组合，并选择使得出牌后手牌价值较高的出牌作为本轮最佳出牌。

地主 MCM 对农民 RHCP 的胜率变化图:



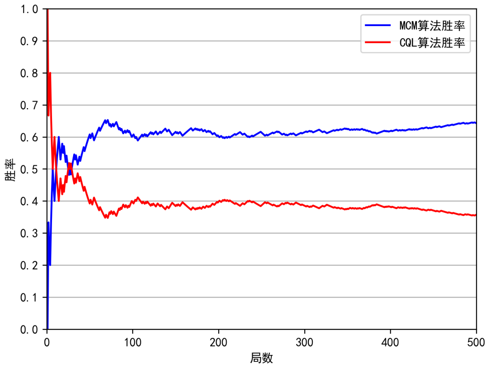
农民 MCM 对地主 RHCP 的胜率变化图:



CQL 算法介绍

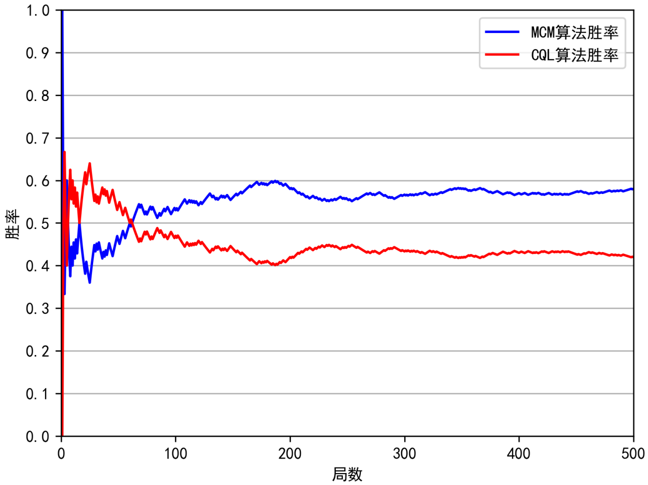
该算法由上海交通大学 You Y 等人提出。You Y 等人针对“斗地主”博弈中，每次出牌时存在较多可能组合牌型的情况，提出一种处理组合动作的新方法组合 Q 学习 (CQL)。

地主 MCM 对农民 CQL 的胜率变化图:



实验结果 ——与 CQL 算法比较

农民 MCM 对地主 CQL 的胜率变化图:



CQL、RHCP 以及 MCM 算法相互比较:

地主		农民一		农民二	
决策算法	胜率	决策算法	胜率	决策算法	胜率
CQL	44.4%	RHCP	21.6%	MCM	34%
CQL	44.8%	MCM	21.6%	RHCP	33.6%
RHCP	52.6%	CQL	6.4%	MCM	41%
RHCP	46.4%	MCM	28%	CQL	25.6%
MCM	63%	CQL	6%	RHCP	31%
MCM	59.2%	RHCP	26.6%	CQL	14.2%
MCM	56%	MCM	22.4%	MCM	21.6%

实验结果 ——CQL、RHCP、MCM 相互比较

CQL、RHCP 以及 MCM 算法相互比较:

地主		农民一		农民二	
决策算法	胜率	决策算法	胜率	决策算法	胜率
CQL	44.4%	RHCP	21.6%	MCM	34%
CQL	44.8%	MCM	21.6%	RHCP	33.6%
RHCP	52.6%	CQL	6.4%	MCM	41%
RHCP	46.4%	MCM	28%	CQL	25.6%
MCM	63%	CQL	6%	RHCP	31%
MCM	59.2%	RHCP	26.6%	CQL	14.2%
MCM	56%	MCM	22.4%	MCM	21.6%

CQL、RHCP 以及 MCM 算法相互比较:

地主		农民一		农民二	
决策算法	胜率	决策算法	胜率	决策算法	胜率
CQL	42%	RHCP	19%	MCM	40%
CQL	47%	MCM	19%	RHCP	34%
RHCP	57%	CQL	10%	MCM	32%
RHCP	52%	MCM	31%	CQL	17%
MCM	66%	CQL	8%	RHCP	36%
MCM	67%	RHCP	20%	CQL	13%

- 上述实验结果详见:
  https://github.com/StarrySky3/experimental-result-/tree/master/experimental-result

目录

总结:

- 论文提出 MCTSHS 算法对“斗地主”进行研究。实验表明该算法针对“斗地主”博弈能做出不错的决策。
- 针对基于 MCTSHS 算法的思考时间过长且已搜索策略未能充分利用的缺点，论文提出 MCM 算法。实验表明，MCM 算法相较于其它智能决策算法具有一定优势。

展望:

- 后续研究对玩家手牌信息进行预测处理。
- 在后续的工作中，可以对玩家进行对手建模。通过预测玩家手牌以实现对手当前状态下的可能决策，从而找到最佳的应对之策以取得游戏胜利。

作者在攻读硕士学位期间参与项目及成果

- 发表了一篇中文核心
- 申请了一项国家发明专利 (在审)
- 参加国家自然科学基金 1 项

敬请各位老师批评指正

谢谢!

References I