#### 致谢

首先感谢刘贵全老师，能够在您的指导下完成我的毕业论文是我莫大的荣幸，您在工作中对我的指导，您对数学敏锐的直觉、您的人格魅力、以及您对学术和人生的执著严谨的态度都给我留下了深刻的印象，再次向您表示最诚挚的感谢和敬意。

感谢实验室项天学长一直以来的支持与鼓励，让我在每个季节都能感受到春天般的温暖。

同时，还要感谢李瑞、杨云帆、张曦文、纪元、彭晗和代映秋同学，你们和我一起学习，分享快乐，共同走过了一段美好的时光。

最后，深深地感谢我的父母，你们一直无条件的支持我，相信我，让我能够一直没有负担地去追求自己的梦想。

#### 目录

[第1章 绪论 5](#_Toc374135063)

[1.1 相关背景介绍 5](#_Toc374135064)

[1.1.1 可配置编程平台的背景介绍 5](#_Toc374135065)

[1.1.2 智能博弈系统的背景介绍 5](#_Toc374135066)

[1.2 可配置智能博弈系统的作用和意义 5](#_Toc374135067)

[1.3 文章的后续框架 6](#_Toc374135068)

[第2章 博弈系统的特征提取和博弈平台的编程框架设计 7](#_Toc374135069)

[2.1 博弈系统的特征提取 7](#_Toc374135070)

[2.1.1 博弈的分类及其各自特点分析 7](#_Toc374135071)

[2.1.2 空间离散回合制棋类游戏的智能博弈分析 9](#_Toc374135072)

[2.2 博弈平台编程框架设计 9](#_Toc374135073)

[2.2.1 棋类游戏的各部件属性模型 9](#_Toc374135074)

[2.2.2 棋类游戏的规则模型 11](#_Toc374135075)

[2.2.3 棋类游戏走法的状态机模型 12](#_Toc374135076)

[2.2.4 棋类游戏的棋盘模型 12](#_Toc374135077)

[2.2.5 棋类游戏的可扩展模型——以和牌类游戏的嵌套模型为例 12](#_Toc374135078)

[2.3 棋类博弈平台的面向对象分析-UML辅助 12](#_Toc374135079)

[2.3.1 12](#_Toc374135080)

[2.4 本章小结 12](#_Toc374135081)

[第3章 多角色敌对搜索内核的构造 13](#_Toc374135082)

[3.1 状态空间搜索算法应用于人工智能的相关背景及其特点 13](#_Toc374135083)

[3.2 状态空间搜索及其理论分支 13](#_Toc374135084)

[3.3 敌对搜索简介 13](#_Toc374135085)

[3.4 敌对搜索的性能优化及其多人多机扩展模型 13](#_Toc374135086)

[3.5 本章小结 13](#_Toc374135087)

[第4章 基于嵌套状态机和图搜索的着法生成器构造 14](#_Toc374135088)

[4.1 可配置的着法搜索模型 14](#_Toc374135089)

[4.2 图搜索算法的深度优先搜索实现 14](#_Toc374135090)

[4.3 本章小结 14](#_Toc374135091)

[第5章 跨平台的GUI接口设计及其MFC实现 15](#_Toc374135092)

[5.1 跨平台GUI接口设计 15](#_Toc374135093)

[5.2 可配置的GUI棋盘模型 15](#_Toc374135094)

[5.3 GUI接口的MFC实现 15](#_Toc374135095)

[5.4 本章小结 15](#_Toc374135096)

[第6章 象棋、五子棋、跳棋的简单配置实践 16](#_Toc374135097)

[6.1 象棋规则和界面的配置 16](#_Toc374135098)

[6.2 五子棋规则和界面的配置 16](#_Toc374135099)

[6.3 六人跳棋规则和界面的配置 16](#_Toc374135100)

[6.4 本章小结 16](#_Toc374135101)

[第7章 论文总结 17](#_Toc374135102)

[第8章 参考文献 18](#_Toc374135103)

[图 1国际象棋走法生成图 14](file:///E:\课题\双学位毕业论文\毕业论文-石守谦.docx#_Toc374210389)

[图 3跳棋走法生图 14](file:///E:\课题\双学位毕业论文\毕业论文-石守谦.docx#_Toc374210390)

[图 4操作与动作的并行包含关系 15](file:///E:\课题\双学位毕业论文\毕业论文-石守谦.docx#_Toc374210391)

#### 内容摘要

随着人工智能领域的日渐发展，棋类游戏中的智能博弈系统已经数见不鲜。棋类游戏因其独特的趣味性和复杂性一直以来都是人们锻炼思维和修身养性的有力工具之一。同时棋类游戏也是人工智能算法研究的发源，更是检验新理论的试验场。历史雄辩地证明，当兴趣与需求结合，人类往往显示出巨大的创造力，这使得智能博弈作为人工智能的一大重要领域经久不衰。然而就当前飞入寻常百姓家的智能棋类游戏而言，一方面编程者水平参差不齐，使得机器智能潜力并未完全发挥；另一方面，编程者需要为每一个独特的棋类游戏重新写智能算法，配置显示界面，这是对社会资源的一种浪费。为了更好地利用计算机资源，调动普通人学习机器智能的积极性，方便编程者快速搭建智能博弈平台，本文介绍了一个通用博弈平台的搭建过程，通过此方式向开发者提供一个方便统一的编程接口，一方面更有效地利用计算资源，方便日后智能游戏的内核统一地随着理论升级，另一方面整合编程资源，将开发者的关注点从开发智能系统转移到扩充现有智能平台上来，有助于为现有智能系统快速引入新空气。

关键字:

人工智能 博弈系统 可配置 最大最小搜索 alpha-beta搜索 扩充搜索

#### Abstract

Undone

Keywords：undone

# 绪论

## 相关背景介绍

### 可配置编程平台的背景介绍

### 智能博弈系统的背景介绍

## 可配置智能博弈系统的作用和意义

随着人工智能领域的日渐发展，棋类游戏中的智能博弈系统已经数见不鲜。棋类游戏因其独特的趣味性和复杂性一直以来都是人们锻炼思维和修身养性的有力工具之一。同时棋类游戏也是人工智能算法研究的发源，更是检验新理论的试验场。历史雄辩地证明，当兴趣与需求结合，人类往往显示出巨大的创造力，这使得智能博弈作为人工智能的一大重要领域经久不衰。然而就当前飞入寻常百姓家的智能棋类游戏而言，一方面编程者水平参差不齐，使得机器智能潜力并未完全发挥；另一方面，编程者需要为每一个独特的棋类游戏重新写智能算法，配置显示界面，这是对社会资源的一种浪费。为了更好地利用计算机资源，调动普通人学习机器智能的积极性，方便编程者快速搭建智能博弈平台，本文介绍了一个通用博弈平台的搭建过程，通过此方式向开发者提供一个方便统一的编程接口，一方面更有效地利用计算资源，方便日后智能游戏的内核统一地随着理论升级，另一方面整合编程资源，将开发者的关注点从开发智能系统转移到扩充现有智能平台上来，有助于为现有智能系统快速引入新空气。

## 文章的后续框架

这篇论文的下面部分将按照这样的顺序进行编排：第二章将详细的介绍本博弈平台的设计框架；第三章将介绍本文采用的敌对搜索算法；第四章介绍通用棋类着法生成器的设计；第五章介绍MFC下GUI接口的跨平台设计与具体实现；第六章使用本平台实地配置象棋、跳棋和五子棋，并分析系统的功能和效率；最后一章将总结整个毕业论文的工作。

# 博弈系统的特征提取和博弈平台的编程框架设计

要设计一个通用的棋类博弈平台，首先要结合各种现有的具体博弈实例分析博弈的分类，并提取出棋类游戏的共同特征，然后以编程概念清晰、性能优越、扩展性强、鲁棒性强等为依据设计编程模型。本章就博弈平台的特征提取和编程框架设计这两个紧密联系的主题给出解决方案。

## 博弈系统的特征提取

### 博弈的分类及其各自特点分析

在此我们不就博弈的数学定义进行深入探讨，仅取定义如下：[2]

设定有N 个“**参与者**”（后文也常称“**玩家**”），对于每一个“参与者” ，都有一个给定的“**策略**”集合. **博弈**（后文也称“**游戏**”）是一个函数， 定义为:

其中R为实数集，“×”表示卡氏积。

也就是说，如果我们知道了参与者的策略集合是什么，那么就可以有一个实数值与之对应。 我们可以把上面的方程拆成两个方程来进一步把它一般化。一个方程是正则形式（Normal form game）的参与者方程，描述策略规定结果的方式。另外一个方程描写参与者对于结果（outcome）集合的偏好（preference）。也就是：

这里  是游戏（博弈）的**结果集合**（outcome set）。对于每一个参与者 都有一个**偏好函数**（preference function）

上文的第二种定义鲜明体现了博弈的特征：玩家可能采取的动作由规则限制，而在规则的限制下，每个玩家都想要得到对自己最有利的结果。（注意这里并不要求像棋类游戏一样，每个玩家的规则和目标都是对称的）

博弈的分类根据不同的基准也有不同的分类。[2]

在当事人之间有没有一个具有约束力的协议意义上，博弈主要可以分为**合作博弈**和**非合作博弈**。

扩展博弈的定义，将博弈“复合”化：每一**阶段**（称为“**元博弈**”）每名玩家均在策略集合中选择策略，并产生单次元博弈的结果（称“**部分结果**”）。在单次结果满足一定条件时，整个博弈才结束。偏好函数也是对部分结果定义的。其实部分结果和最终结果的区别仅仅在于在达到最终结果时停止进入下一元博弈阶段。这称为**即时制博弈**。（将时间连续的博弈划分为细小的时间片，若不同玩家策略产生的结果冲突，由概率决定选择怎样的结果）

定义**回合制博弈**：每一阶段只有一名玩家的策略集合可能不是空集，该阶段称为当前玩家的**回合**。然后回合按照玩家标号顺序地交换的即时制博弈。

按照参与人对其他参与人的了解程度分为**完全信息博弈**和**不完全信息博弈**。完全博弈是指在博弈过程中，每一位参与人对其他参与人的特征、策略空间及收益函数有准确的信息。如果参与人对其他参与人的特征、策略空间及收益函数信息了解的不够准确、或者不是对所有参与人的特征、策略空间及收益函数都有准确的准确信息，在这种情况下进行的博弈就是不完全信息博弈。

定义博弈的**规则**：事先确定的影响产生策略集合和偏好函数的因素。

定义**属性**：为了方便讨论策略集合的生成，将玩家影响策略集合和偏好函数的关键状态分离出来称**属性**。

定义**棋牌类游戏**：

* 对每个玩家而言，策略集合的产生规则绝对交换对称（常义棋）或概率上交换对称（常义牌）；
* 每个玩家的偏好函数交换对称且对特定结果的偏好函数值为2(获胜)，另一些结果为1(和局)，其余结果的偏好函数全为0(失败)；
* 是回合制博弈。

（各种体育赛事、各种常义棋牌类游戏等基本均属于此定义）。

为了讨论方便，将棋牌类游戏**进一步分类**：策略集合与位置相关的棋牌类游戏称**棋类游戏**（如足球、象棋等常义棋类游戏），其余称**牌类游戏**（如常义牌类游戏、“石头剪子布”）。

其中“**位置**”定义为玩家所拥有的一个或多个向量属性。（往往在对“策略”集合的形成有影响时，位置才对博弈有作用）

在位置相关的博弈中，可以根据位置可取值集合的可数性分类为**空间离散的**和**空间连续的**。

### 空间离散回合制棋类游戏的智能博弈分析

空间离散棋类游戏要求方便地表示其位置及各位置间的关系，可以定义一个**棋盘**作为索引。棋盘中每一个位置属性抽象成一个**棋子**，定义为所有与单个位置相关的属性集合。

棋盘可以是无穷大，也可以有有限范围，也可以有周期性边界条件，也可以有任意的相邻关系，并不一定满足实数的运算法则。但所有可能的位置集合必须可数，否则不能表示棋盘局部的全部位置信息。

将与所有与全体玩家有关的影响策略集合的属性归为**棋盘属性**统一管理，将其余属性归为每个**玩家的属性**。

注：

1. 每个玩家的策略集合和偏好函数都是对称的，所以属性内容一致。
2. 与策略集合和偏好函数均无关的属性（如玩家名字，棋子颜色等）对博弈没有任何影响，在定义上放在了玩家上，但是后文在不影响规则的原则下根据方便进行转移。
3. 为讨论方便，后文将“空间离散回合制棋类游戏”简称为“棋类游戏”，有时将玩家称为**棋手**。

## 博弈平台编程框架设计

上文定义了棋类游戏，本节将继续细化各部件的属性模型。我们将逐渐脱离棋类游戏的数学定义，将重点放在框架设计和模型构造上。

首先声明，棋类游戏的形式千差万别，这里只提取一些必要属性和一些常见属性。

### 棋类游戏的各部件属性模型

为了讨论方便，定义几个名词：

**局面**：整个游戏中影响任一玩家策略集合的属性集合。

**着法**：每个回合玩家所有的行动。（后文也称**行动**）

**动作**：玩家当前回合内可以采取的单个策略。

**操作**：泛指玩家影响局面的一类动作。

**走法**：泛指玩家的着法，描述不同操作的顺序关系。

对于着法、动作、走法和操作的区别，在*2.2.3棋类游戏走法的状态机模型*中会有具体说明。往往是着法包含几个操作，而每个操作都以动作的形式具体体现。

下面描述棋类游戏各部件属性构成及必要解释：

* 棋盘的通用属性：
* 当前回合数
* 当前回合操作者（后文称“**当前玩家**”）
* 当前获胜玩家和当前失败玩家（在多人博弈中记录已经确定获胜或失败的玩家，方便确定何时到达博弈最终结果）
* 当前各棋子的位置
* 为了编程方便，加入当前并未在棋盘（玩家没有此向量属性）但实际上仍然存在的棋子（比如为了走一步而刚刚被拿起的棋子）
* 棋盘可能的扩展属性举例：
* 特殊格点及其作用（如军棋中的大本营）
* 当前季节（比如模拟战争游戏）
* 当前游戏总时间
* 历史着法（可能规则允许悔棋，也对如象棋中“不变作和”、“长将判负”等规则给予支持）
* 当前禁止着法（是规则中对每个玩家有同等约束力的限制，如围棋中“打劫”）
* 棋手间的结盟关系
* 棋手通用属性：
* 当前拥有的棋子
* 棋手可能的扩展属性举例：
* 个人已用时间
* 生命力
* 棋手名字
* 棋手颜色
* 占据的格点（例如围棋）
* 当前行动力（有的规则可能限制棋手的总行动能力而并不限制具体动作）
* 棋子通用属性:
* 当前坐标
* 所属棋手
* 棋子可能的扩展属性举例:
* 颜色
* 生命值
* 行动力
* 行动消耗的行动力

### 棋类游戏的规则模型

规则用来限制走法或者判断输赢（其实是决定了偏好函数）。其逻辑模型根据对各部件属性的限制分类如下：

* 全局通用规则：
* 初始局面（包括初始棋子位置等各属性的初始值）
* 玩家轮流顺序（玩家的排序）
* 棋盘范围
* 特殊的空间关系（如象棋九宫格、河界等）
* 使局面陷入循环的惩罚办法
* 局面随着回合数、游戏时间、玩家之间势力评估的变化（比如每过一年，玩家就老一岁，但是会有一年的收入等）
* 可能的全局扩展规则：
* 回合限制
* 总时间限制、玩家时间限制
* 玩家相关的通用规则：
* 每回合产生着法的规定
* 可能的玩家相关扩展规则：
* 行动力增长率
* 生命值增长率
* 棋子相关的通用规则：
* 本类棋子着法产生的具体规则
* 被拿起的条件和效果
* 被放下的条件和效果
* 吃子的条件和效果
* 被吃的条件和效果
* 可能的棋子相关扩展规则：
* 各动作消耗的行动力
* 棋子的行动力增长率
* 棋子的生命值增长率

### 棋类游戏走法的状态机模型

本段讨论走法规则的具体模型，也给出上文所述“着法（Action）、动作（Motion）、走法（Option Graph）、操作（Option）”四个概念的具体关系。

一般地说，走法是着法的模板，操作是动作的模板。也就是说一个着法是一个具体的走法，一个动作是一个具体的操作。

用以操作为节点的有向图表示走法，用走法规定每一回合玩家做能采取的着法的操作构成，既方便制定规则，又方便统一不同棋类游戏着法的产生方式。例如：

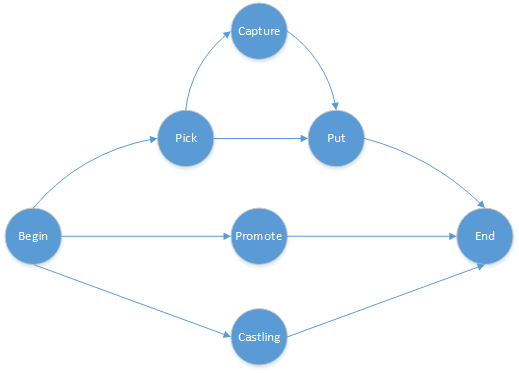
1. 国际象棋中，“一步棋”（一个着法）可以被如下的操作图生成：

图 1国际象棋走法生成图

注：其中，棋子的一次移动是由pick和put两个操作组成的，在put之前，可以吃掉即将到达位置的棋子。promote指升变，castling指王车易位。

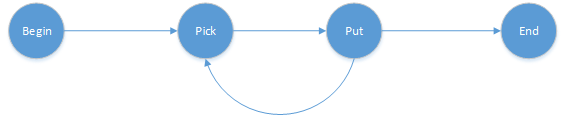
1. 跳棋中，“一步棋”（一个着法）可以被如下的操作图生成：

图 2跳棋走法生图

当我们要产生所有可能的着法时，需要深度优先搜索整个操作图，每次搜索到end节点，就将路径保存起来。但是应该注意，我们要保存的“路径”是动作路径，而非操作路径，而操作路径和动作路径的关系是每个操作路径上可能并行地包含多个动作路径，也可能有一些动作路径在操作路径里中断。

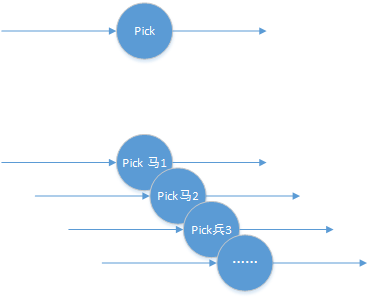
例如，实际由搜索生成的动作图的一部分举例如下：

图 3操作与动作的并行包含关系

即，一个pick操作里可能包含很多具体的动作。

在搜索动作图时采用深度优先搜索，能完全地扫过所有并行的动作，而且采用深度优先搜索可以自然地体现走法的一般性与规则的特殊性的辩证关系。如马1已经在四个方向均被“蹩马腿”，但是在搜索操作图生成走法的过程中，要先pick马1，然后在put或capture的时候调用onCapture或onPut回调函数，发现蹩马腿，onCapture或onPut回调函数返回false，知前方探测失败，此时可以退回pick，发现pick除了capture和put外并无后继节点，则返回pick的上一级——begin，发现已经搜索完马1的所有可能，则退出对马1 的搜索。

另一方面，每当搜索到end，知道一条路径顺利搜索完毕，保存当前路径，返回上一级继续后面的搜索。

整个搜索过程可以用“**运行栈**”和“**记录栈**”管理，“运行栈”是“横向”的，对应本操作包含的平行动作，“记录栈”是“纵向”的，对应已经探测过的操作路径和当前动作路径，以便在探测到end节点时保存整个路径。

上面仅仅是形象的解释，严格的算法描述如下：

1. Begin作为操作节点入操作记录栈，当前记录栈清空，着法记录缓冲清空，开始新一层的搜索。
2. 当开始新一层的搜索，若操作记录栈顶为end，则保存当前动作记录栈, 操作记录栈弹栈，返回上层搜索的f)；否则，操作和动作运行栈清空，若操作记录栈非空，取定操作运行栈顶为当前操作，生成当前操作的后继操作，本层搜索的入操作运行栈。
3. 取定操作运行栈顶为当前操作。
4. 以当前局面为准，生成当前操作所有包含的动作，入动作运行栈。
5. 若当前动作运行栈非空，取定栈顶的元素为当前动作，入动作记录栈，棋盘进行当前动作指定的操作，若成功则跳转到b)，开始深一层的搜索，否则跳转到e)，继续搜索当前操作的其余动作。
6. 若当前动作运行栈空, 说明当前操作没有更多可用动作。若当前操作运行栈非空，操作运行栈弹栈，跳转到c)，继续当前层操作的搜索，否则当前操作运行栈空，说明上层搜索的所有后继节点已经搜索完毕，操作记录栈弹栈，返回上层搜索的f)。否则当前动作运行栈非空，棋盘恢复到深层搜索之前的状态，跳转到e)继续当前操作的搜索。（这里适合用递归函数而不是循环做，可以借助程序的局部变量管理各层搜索的运行栈）

### 棋类游戏的棋盘模型

上段介绍了一个走法的生成，本段介绍常义棋盘模型的构造。

这里“常义棋盘”指棋盘上的点没有特殊的相邻关系，棋盘上的点集完全与一个二维整数向量集合的子集在平面上的相邻关系的意义上同构。即棋盘中任意两点之间的可达路径均可落在用两簇平行且间距相同的直线或线段画出的格子（称为棋盘的**基**）的边界上。

又定义**棋路**：棋子所能行动的方向，这里并不一定都是基，也不一定是直线。特别将直线或线段的棋路称为**直棋路**。

以上定义主要是为了方便棋盘的配置。我们可以只规定棋盘的范围、基的夹角、直棋路的方向和一些附加棋路，就能得到几乎所有的常义棋盘的逻辑结构和显示方式。

如五子棋，棋路两条，夹角90°，范围是一个正方形。

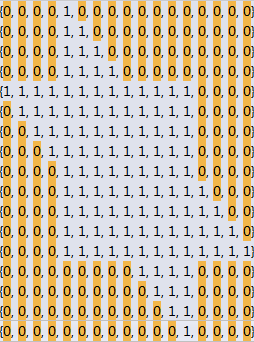
又如跳棋，基的夹角120°，棋路有三条，两个分别为两个基的方向，另一个为两个基的向量和的方向。范围为下面矩阵为1的地方，这样既实现了棋盘范围的任意配置，又实现了跳棋可以行走的三个方向。

图 4跳棋棋盘范围的定义

但对于象棋九宫格等“局部棋路”的情形，应该特别配置，不能作为真正的棋路。

### 棋类游戏的可扩展模型——以和牌类游戏的嵌套模型为例

本章介绍的棋类游戏模型是高度可扩展的。

## 棋类博弈平台的面向对象分析-UML辅助

### 

## 本章小结

# 多角色敌对搜索内核的构造

## 状态空间搜索算法应用于人工智能的相关背景及其特点

## 状态空间搜索及其理论分支

## 敌对搜索简介

## 敌对搜索的性能优化及其多人多机扩展模型

## 本章小结

# 基于嵌套状态机和图搜索的着法生成器构造

## 可配置的着法搜索模型

## 图搜索算法的深度优先搜索实现

## 本章小结

# 跨平台的GUI接口设计及其MFC实现

## 跨平台GUI接口设计

## 可配置的GUI棋盘模型

## GUI接口的MFC实现

## 本章小结

# 象棋、五子棋、跳棋的简单配置实践

## 象棋规则和界面的配置

## 五子棋规则和界面的配置

## 六人跳棋规则和界面的配置

## 本章小结

# 论文总结

本篇论文主要介绍了。围绕着这些核心工作，论文介绍了。为了能让读者对有清晰的认识，论文依次介绍了，随后等，而论文的主要工作放在了最后几章。该论文最后介绍了

但是还有很多有待改进的地方，比如等等。由于我目前自身能力及毕业设计时间的限制，无法将这些一一完成。如果这些工作都能得到完成，。

# 参考文献

[1]