人机混合智能规划平台

V1.0

卓汉逵、李运聪、实验室全体同学

智能规划实验室

中山大学 计算机学院

摘 要

为使得设计规划领域和问题的描述这项工作对于非智能规划领域专家更加容易理解，近年来，规划领域的可视化表示已经被应用于智能规划与调度的知识工程中。但是目前专注于这方面的知识工程应用普遍存在着规划领域建模的效率不高、不能根据规划求解结果对规划领域进行微调的问题。针对这些问题，中山大学数据科学与计算机学院智能规划实验室（XPlan-Lab）设计并实现了基于人机混合智能的规划领域模型获取平台（下面简称“平台”）。

平台基于MVC设计模式实现对规划领域和问题的描述的可视化建模，可以协助设计人员即使在不熟悉智能规划领域定义语言（Planning Domain Definition Language, PDDL）相关语法的情况下，以一种直观的方式进行规划领域建模，并且支持 PDDL 基本语法约束检查。为提高规划领域设计人员进行规划领域可视化建模的效率，平台在规划领域设计层次上构建了领域知识库抽象层，作为规划领域描述的依赖基础。设计人员可以直接实例化领域知识库中的领域知识模板，而不是从原子性的领域知识元素开始进行规划领域建模，这明显地提高了规划领域可视化建模的效率。

同时为协助设计人员针对特定领域和问题对规划领域进行微调，平台基于人机混合智能中的人在回路智能规划（Human-in-the-Loop Planning, HILP），实现了规划的交互式可视化以及规划验证，支持设计人员根据规划验证结果直接对规划领域进行微调，以修正规划动作序列里的缺陷。

目 录

第 1 章　引言

本章主要介绍人机混合智能规划平台的背景以及研究意义，介绍分析了国内外智能规划与调度知识工程、规划领域建模、规划分析的发展现状，并阐述了平台的主要内容和贡献，最后给出了平台的主要结构。

1.1 平台研究背景及意义

1.1.1 研究背景

智能规划是人工智能（Artificial Intelligence）科学的一个具体分支。一个具体的规划问题通常由规划领域描述、规划问题描述（包括初始状态以及目标状态）组成。在智能规划应用中，世界的状态被描述为一系列确定事实的集合，假设给定世界的一个初始状态，智能规划可以描述成为了找出可以引导初始状态转移到所需的目标状态的动作序列的一个决策的过程[1]。而智能规划过程的结果就是规划（Plan），一个有序的动作序列，规划里的每个动作都可以连续地修改相关的世界状态。在完整的规划也就是动作序列被应用后，世界最终的状态会转换为智能规划应用问题里所希望的目标状态。

一个待求解的规划问题必须要用形式化的语言（比如目前主流的规划领域定义语言 PDDL）来进行描述，也就是规划领域和问题描述，才能被智能规划与调度系统（也就是狭义上的规划器）作为输入，经过计算求解得到规划问题的规划结果作为输出。近年来，尽管智能规划与调度系统取得了长足的发展，但是这些系统仍然需要依赖于经过精心设计的规划领域和问题的描述作为系统的输入，并针对特定的规划领域和问题进行微调。人工智能规划与调度的知识工程便涉及到领域模型的获取、设计、验证和维护，以及选择和优化适当的机制来处理这些过程，这些过程直接影响到实际规划与调度应用的成功与否。

规划领域和问题的描述无疑是在使用智能规划技术时的第一道关键步骤，为了使得设计规划领域和问题的描述这项工作对于非智能规划领域专家更加容易理解，近年来，规划领域的可视化表示已经被应用于智能规划与调度的知识

工程中。通过对规划领域的可视化建模，使得规划领域模型的获取、设计、验证和维护更加容易理解，但是同时也带来了一些不可避免的问题。一方面，相比于传统的由 PDDL 专家使用文本编辑进行领域建模，规划领域的可视化建模其实是以效率来换取易理解性，虽然降低了领域建模的门槛，但是也导致了建模的效率不高的问题[2]。另一方面，目前专注于规划领域可视化建模的知识工程应用普遍不能根据规划结果的反馈对规划领域进行微调，从而修正规划领域中可能存在的不合理的领域知识。

人机混合智能（Human-Machine Mixed Intelligence）中的人在回路智能规划（Human-in-the-Loop Planning, HILP）旨在将人的作用引入到智能系统中，在这种智能系统设计范式中，人始终作为智能系统的一部分，当智能系统的结果输出的置信度低时，人可以主动地介入其中从而调整智能系统并给出科学合理的正确的问题求解，构成提升智能水平的反馈回路[3]。

1.1.2 研究意义

尽管近年来智能规划与调度系统一直在快速地发展，但是这些系统仍然需要依赖于经过精心设计的规划领域和问题的描述作为系统的输入，所以规划领域和问题的描述无疑是在使用智能规划技术时的第一道关键步骤，为了使得设计规划领域和问题的描述这项工作对于非智能规划领域专家更加容易理解，近年来，规划领域的可视化表示已经被应用于智能规划与调度的知识工程中。

通过对规划领域的可视化建模，即使是不熟悉 PDDL 语言相关语法的领域设计人员都可以借助用户图形界面中图形化的元素进行领域模型建模。图形化的建模工具无疑大大降低了智能规划领域建模的门槛和难度。

规划领域的可视化建模降低了规划领域建模的门槛和难度，那么平台设计的领域知识库就是在此基础上提升了可视化建模的效率。因为在基于领域知识库的前提下，领域设计人员在进行领域建模的时候，可以避免每次都要从繁杂的底层的领域知识元素开始进行建模，而是可以通过实例化领域知识库中已存在的领域知识模板来直接生成部分的所需的领域模型。领域知识库把设计人员从繁杂的底层的领域知识元素建模工作中解放出来，从而可以专注于高层的领域模型设计，这无疑大大提升了可视化建模的效率。

虽然有了规划领域的可视化建模可以降低领域建模的难度，但是即使是对于经验丰富的智能规划领域专家来说，设计科学合理的领域模型都是非常复杂和耗时的过程。而要验证领域模型设计的科学合理性，那么把领域模型输入到实际的规划问题求解中无疑是最直接有效的途径，然后根据规划领域以及问题描述、求解得到的规划进行规划的验证。基于人在回路智能规划 HILP 的思想，通过规划以及规划验证的可视化，识别出规划动作序列中存在的缺陷动作，设计人员据此进而对规划领域进行修正，提高领域模型设计的科学合理性。

综上所述，通过综合规划领域可视化建模、规划领域知识库、集成规划求解器、集成规划验证器、规划领域模型微调等功能，为建模智能规划领域模型提供一系列完整的解决方案，平台可以大大降低领域设计人员进行领域建模的难度和提升领域可视化建模的效率，以及可以协助设计人员设计出科学合理的领域模型。

1.2 研究现状

1.2.1 智能规划与调度知识工程现状

自从在 2008 年国际智能规划与调度会议上（International Conference on Automated Planning and Scheduling, ICAPS）举办了第一届智能规划与调度知识工程研讨会（Knowledge Engineering for Planning and Scheduling, KEPS），近十年来围绕着智能规划与调度知识工程的研究一直在持续地发展。

尽管智能规划与调度系统取得了快速的发展，但是这些系统仍然需要依赖于经过精心设计的规划领域和问题的描述作为系统的输入，并针对特定的规划领域和问题进行微调。人工智能规划与调度的知识工程涉及到领域模型的获取、设计、验证和维护，以及选择和优化适当的机制来处理这些过程，这些过程直接影响到实际规划与调度应用的成功与否。领域无关的规划器与依赖领域知识的规划器之间的性能差距，清楚地表明了智能规划与调度中知识工程技术的重要性。

智能规划与调度知识工程研讨会继承了已举办多届的国际智能规划与调度

知识工程竞赛（International Competitions on Knowledge Engineering for Planning and Scheduling, ICKEPS）的传统与使命，不同于后者的主题仅仅聚焦于规划领域编码技术和软件工具，智能规划与调度知识工程研讨会涉及到人工智能规划与调度知识工程的所有方面，包括但不限于：

1) 规划领域和问题描述的公式化
2) 规划领域知识的获取方法以及工具
3) 规划器和调度器的前置以及后置处理技术
4) 控制知识的获取以及改进
5) 描述规划领域的形式化语言
6) 规划领域知识的重复利用
7) 从特定应用领域语言到可以直接输入规划器的领域模型语言（例如PDDL）的转换
8) 规划器的启发法、参数和控制知识的格式化规范
9) 从泛型知识本体导入规划领域知识
10) 描述规划器和调度器的功能的知识本体
11) 规划问题的自动化再形成
12) 领域知识的自动化提取过程
13) 规划领域模型、问题以及规划的验证
14) 规划领域模型、搜索空间以及规划的可视化方法
15) 映射规划领域属性和规划技术
16) 规划的表示以及重复利用
17) 知识工程层面的规划分析

1.2.2 规划领域建模现状

在实际的智能规划与调度应用中，规划领域建模是一项非常复杂的任务。它包含了规划应用的需求分析、形式化的领域模型描述、使用合适的规划器对领域模型进行测试[2]。一般地，在规划领域建模中有三种不同的领域知识建模策略[2]，分别是：

1) 传统的由 PDDL 专家使用文本编辑器的方法，基于文本的、结构化的描述来进行规划领域建模。

2) 基于 GUI 的规划领域可视化建模。

3) 分层的、基于对象标记的方法。

对于这三种不同的规划领域建模策略，近年来不断涌现了一些非常优秀的的知识工程应用系统。

PDDL Studio 是一个支持领域设计人员编写和编辑 PDDL 规划领域以及问题文件的 PDDL 编辑器[4]。PDDL Studio 的主要目的是向知识工程师提供编辑和检验 PDDL 代码的功能，而不管 PDDL 代码是如何产生的。PDDL Studio 支持语法错误识别、PDDL 代码高亮和集成规划求解器，就像使用集成开发环境（IDE）编写传统的编程语言一样。

美国宇航局艾姆斯研究中心（NASA Ames Research Center）主导开发了人工智能规划与调度、约束编程及优化一体化平台，可扩展通用远程操作规划体系结构（Extensible Universal Remote Operations Planning Architecture, EUROPA）[5]。EUROPA 的主要目标是处理真实世界中的复杂问题，能够处理两种形式化表示语言，新领域定义语言（New Domain Definition Language, NDDL）[6][7]以及动作标记建模语言（Action Notation Modeling Language, ANML）[8]，后者已经被应用于 NASA 的众多任务中。EUROPA 支持领域建模、结果可视化和一个交互式的规划过程，但是并不支持目前在智能规划与调度中占据主流地位的 PDDL。

itSIMPLE 提供了一个允许领域知识工程师使用统一建模语言（UML）进行规划领域建模的开发环境[9][10][11]。该系统可以收集规划问题的需求，然后使用

UML，以面向对象的方式，对规划问题进行建模、可视化以及修改等等。itSIMPLE 支持导出 PDDL 以及集成规划求解器。

基于对象的智能规划图形界面（Graphical Interface for Planning with Objects, GIPO）是一个支持智能规划与调度知识工程的集成图形工具，基于它独有的面向对象语言（object-centred languages, OCL, OCL_h）[12][13]。

VIZ 是一个受到 GIPO 和 itSIMPLE 设计思想影响的知识工程工具，支持非知识工程师用户使用简易的、用户友好的 GUI 进行领域建模并导出 PDDL[14]。该工具以一种非常直观的方式，利用简单且显而易见的图形来建模 PDDL 领域模型。但是 VIZ 并不支持集成第三方规划求解器。

相比于传统的由 PDDL 专家使用文本编辑进行领域建模，规划领域可视化建模不要求规划领域设计人员同时也必须是个 PDDL 专家，这无疑降低了领域建模的门槛。但是事实上规划领域的可视化建模耗费大部分时间用于建模部分[2]，是以效率来换取易理解性的，虽然降低了领域建模的门槛，但是在领域建模的效率上比不上传统的由 PDDL 专家使用文本编辑进行领域建模。

1.2.3 规划分析现状

规划分析是完整的智能规划与调度系统中不可或缺的一部分。在智能规划与调度研究领域中，规划分析的发展方向主要集中在用于规划验证、规划可视化、动画、规划查询和规划总结的工具或者技术[15]。

随着规划动作序列中动作的数量以及动作之间因果关系的增多，规划分析这项工作变得越来越复杂和耗费时间。事实上，包含数量众多甚至数百个动作的规划对于人类来说几乎是不可读的。为了使得大规模的规划对于人类更加具有透明性和可读性，对规划分析的研究具有非常重要的意义。

VisPlan 是一个提供规划交互式可视化和验证的应用程序[16]。VisPlan 可以帮助使用者分析以及可视化规划，寻找和展示规划动作序列中动作之间的因果关系，验证规划的正确性，识别出规划中可能存在的缺陷并给予高亮提示，最终允许使用者交互式地修改规划的动作序列从而修复规划中的缺陷。

InLPG 是一个支持规划可视化、检验和产生的交互式工具[17]。InLPG 集成

了领域无关规划求解器 LPG，提供了非常丰富的规划信息的可视化，包括动作序列中某个动作成功被执行或者执行失败的理由、规划所需的资源消耗的趋势、规划动作的时序调度等等。更进一步，该工具支持使用者在规划过程中干涉规划器的行为，或者修改规划。

VAL 系统是一个基于 PDDL 的规划验证工具[18]。VAL 的系统输入包括符合 PDDL 语法的规划文件、以及相关联的规划领域和问题描述文件，然后 VAL 可以验证规划里动作序列的执行，可以识别规划动作序列的变迁是否能满足目标状态或者规划动作序列中是否存在不合理的动作。

目前关于规划验证方面的研究对于规划验证后的进一步处理方法较为单一，例如，如果被验证的规划动作序列中存在不合理的动作，那么允许使用者可以交互式的修改当前正在被验证的规划本身，但尚缺乏对相关的规划领域作进一步处理的研究。

1.3 平台的研究内容与主要工作

平台设计并实现人机混合智能规划平台，提供规划领域模型可视化建模、领域知识库、语法约束检查、数据转换、规划、规划验证、规划领域模型微调等功能。

平台包含 7 大功能模块：

1) 可视化建模模块，采用 MVC 设计模式，负责规划领域模型建模，可视化图形信息支持 XML 配置化。

2) 领域知识库模块，负责维护、管理领域知识，包含两类基本领域知识模板，支持 XML 配置化。

3) 语法约束检查模块，负责 PDDL 基本语法约束检查。

4) 数据转换模块，负责语言和相关输入输出文件的转换，支持输入输出 XML 文件，支持输出 PDDL 文件，支持 XML 转换为 PDDL。

5) 规划模块，集成多种规划器，规划器的设置支持 XML 配置化，负责执

行规划并展示规划结果，规划结果支持 XML 配置化。

6) 规划验证模块，集成规划验证器，验证器的设置支持 XML 配置化，负责执行规划验证并展示详细的验证信息。

7) 领域模型微调模块，对于有缺陷的规划提供修复建议并支持用户交互式的修改规划领域模型，包含可视化模型修改、文本编辑修改。

平台的工作主要集中在以下几个方面：

1) 分析知识工程中智能规划领域与问题描述、HILP 的背景，阐述为建模智能规划领域模型提供一系列完整解决方案的平台的优势与意义。然后介绍智能规划与调度知识工程、规划领域建模以及规划分析的相关研究工作，并基于面向对象的思想、统一建模语言 UML 以及规范化的软件开发流程，对平台进行需求分析、设计与实现，开发了人机混合智能规划平台，该系统具有功能完整、简单易用等优点。

2) 研究可以提升规划领域可视化建模的效率的方法。在可视化建模的设计层次中构建抽象的领域知识库层，使得依赖于领域知识库的可视化建模的效率大大提升。

3) 研究可以帮助领域设计人员建模科学合理的规划领域模型的方法。基于人机混合智能中人在回路智能规划 HILP 的思想，通过执行规划、执行规划验证、提供修复建议一系列步骤，交互式的帮助设计人员对规划领域模型进行微调，以不断修正缺陷。

平台的创新点主要体现在以下几个方面：

1) 在规划领域设计层次中构建了领域知识库抽象层，作为规划领域和问题描述的依赖基础。依赖于领域知识库，设计人员在建模规划领域的时候，平台可以提供智能提示以及自动补全，从而直接实例化领域知识库中的领域知识模板来进行规划领域建模，而不是从原子性的领域知识元素开始进行规划领域建模，这明显地提高了规划领域建模的效率。

2) 基于人机混合智能的人在回路智能规划 HILP，集成规划求解器、集成规划验证器，实现了规划的交互式可视化以及规划的执行和验证，可以识别出规划动作序列里可能存在的缺陷，提供修复建议并支持设计人员根据修复建议直接对规划领域进行可视化微调，从而修复规划领域中可能存在的不合理的领域知识，最终帮助设计人员设计出科学合理的规划领域模型并提升实际的智能规划与调度应用的成功率。

3) 作为人机混合智能规划平台，平台的主要目的是为普通的领域设计人员提供一个建模智能规划领域模型的完整解决方案，一个完整的领域模型开发流程，包括领域知识库、可视化建模、约束检查、数据转换、文本编辑、规划求解、规划验证、领域模型微调等功能，方便设计人员在统一的集成开发环境中建模规划领域模型。

第 2 章　相关理论和技术综述

本章对人机混合智能规划平台开发过程中所用到的相关理论和关键技术进行详细的介绍。

2.1　智能规划

智能规划是人工智能科学的一个具体分支。假设给定世界的一个初始状态，智能规划的目标就是找出可以引导初始状态转移到目标状态的动作序列。所以智能规划过程的结果就是规划（Plan），一个有序的动作序列，规划里的每个动作都可以连续地修改世界的状态。在完整的规划也就是动作序列被应用到世界后，世界最终的状态会转换为所希望的目标状态。

2.1.1　描述规划问题的语言

为表示现实世界中的真实规划问题，简化智能规划模型是非常普遍的做法[1]。下面的内容只考虑经典智能规划模型，它确保规划世界是确定性的、有限的、完全可观察的（能掌握世界的所有信息）和静止的（除了规划代理，世界中没有其他的动作）。如果在同一时间里，模型是非连续的（时间以及动作的应用），那么该模型描述的是 STRIPS（Stanford Research Institute Problem Solver）型问题[19]；否则，该模型描述的是时序规划（temporal planning），考虑持续动作（durative action），也就是动作从开始发生到产生动作的效果需要经过一段时间，而不是瞬间完成的。

为了能够描述各种不同类型的规划问题，描述规划领域和问题的语言应该具有足够的表达能力。同时为了让规划器被设计得能高效地解决通用的规划问题，应该对描述规划领域和问题的语言进行某种程度的限制。

描述规划领域和问题的语言具有以下 5 个基本的语言组成成分：

1)　状态（States）

世界的任意状态都可以表示为字面量（literals）的结合物

（conjunction）。假设世界是封闭的，也就意味着如果一个字面量没有在世界状态中被明确地显式地声明，那么这个字面量也就不存在于世界状态中。基于这个封闭世界的假设，世界状态中只包含了值为真的字面量。

2) 操作符（Operators）

一个操作符就是一个动作的模式。从操作符实例化一个动作，就是将操作符的所有变量替换为实际的参数。每一个操作符都要声明它的前提条件（preconditions）和后置效果（effects）。

3) 规划领域（Planning domain）

规划领域 Σ 是一个三元式（S, A, γ）。S 是所有可能的世界状态的集合；A 是所有可能的动作的集合；γ 是转移函数，描述了对指定的世界状态应用指定的动作后的结果世界状态。

4) 目标（Goals）

目标 g 就是规划问题中所希望得到的世界状态的集合。

5) 规划问题（Planning problem）

规划问题 p 是一个三元式（Σ, S_0, g）。Σ =（S, A, γ）是规划领域；S_0 是世界状态 S 集合中的初始状态；g 是目标状态集合。

2.1.2 规划

规划是智能规划决策过程的目标。规划 π 是一个动作的序列$<a_1, a_2, \ldots, a_k>$，$k = |\pi|$ 是规划的长度。

2.2 规划领域定义语言 PDDL

规划领域定义语言 PDDL 是人工智能规划的标准定义语言。最初的 PDDL 1.2 版本是由 Drew McDermott 等人在参加 1998 年第一届国际智能规划竞赛（International planning competition, IPC）时为了统一规划器的需求和输入格式

而发展出来的，PDDL 以 STRIPS 和动作描述语言（Action description language, ADL）[20][21][22]作为基础[23]。多年来，为了使 PDDL 能描述尽可能多的规划领域，伴随着 PDDL 版本的迭代，越来越多的新特性被添加到 PDDL 中，PDDL 也逐渐成为用于表示智能规划问题的标准语言。

PDDL 的基本组成成分包括：

1) 对象（object），规划应用所关注的现实世界的对象。

2) 谓词（predicate），规划应用所关注的属于对象的属性，可以为真或者假。

3) 初始状态（initial state），规划应用开始的世界状态。

4) 目标状态（goal state），规划应用所希望的世界状态。

5) 动作/操作（action），可以改变世界状态的方式。

一个完整的 PDDL 定义应该包含两部分，分别是领域（domain）定义和问题（problem）定义。尽管 PDDL 语言标准本身不要求这样做，但是大多数规划器都要求将规划定义和问题定义划分为两种独立的文件：

1) 领域定义文件，描述了领域的谓词和动作。

2) 问题定义文件，描述了问题的对象、初始状态和目标状态。

在语法结构上，PDDL 通过关键字（keyword）组织其基本组成成分，从而形式化领域定义和问题定义，可参考 2.2.1 以及 2.2.2 中的样例。

2.2.1 领域定义

基于 2.2 所提到的领域定义，领域定义中包括了领域谓词和操作符（在 PDDL 中称为动作），还可能包含类型定义、常量和静态事实等等。一般的领域定义格式如下示例：

```
(define (domain DOMAIN_NAME)
  (:requirements [:strips] [:equality] [:typing] [:adl])
  (:predicates (PREDICATE_1_NAME [?A1 ?A2 ... ?AN])
               (PREDICATE_2_NAME [?A1 ?A2 ... ?AN])
           ...)

  (:action ACTION_1_NAME
```

```
    [:parameters (?P1 ?P2 ... ?PN)]
    [:precondition PRECOND_FORMULA]
    [:effect EFFECT_FORMULA]
   )

  (:action ACTION_2_NAME
    ...)

  ...)
```

使用（“[]”）限定的元素是可选的。领域、谓词和动作等等的标志符通常包含字母数字、连字符（“-”）和下划线（“_”）。谓词和动作的参数会被参数前面的问号（“?”）所区分。谓词声明部分（:predicate）中使用的参数仅仅用于指定谓词的参数个数，参数的名字并不重要，谓词甚至可以没有参数。同时谓词支持重载，但是所有重载的谓词的参数个数必须相同。

如果领域定义声明了支持类型（:typing），那么所有的类型名字必须要在被使用之前进行声明：

```
(:types NAME1 ... NAME_N)
```

声明谓词或者动作的参数类型的格式如下：（连接参数名字和类型名字的连字符前后必须都含有空格）

```
?X - TYPE_OF_X
```

对于 STRIPS 型领域，一个动作前提条件的公式可以是：

1) 原子式：（谓词的参数必须是动作的参数）

```
(PREDICATE_NAME ARG1 ... ARG_N)
```

2) 原子式的联合

```
(and ATOM1 ... ATOM_N)
```

在 PDDL 中，动作的后置效果没有被明确的划分为肯定的效果（增加）和否定的效果（删除），但是否定的效果（删除）会被（“not”）明确的指示。对于 STRIPS 型领域，一个动作后置效果的公式可能包含：

1) 增加原子式：（谓词的参数必须是动作的参数）

```
(PREDICATE_NAME ARG1 ... ARG_N)
```

2) 删除原子式：

```
(not (PREDICATE_NAME ARG1 ... ARG_N))
```

3) 原子式的联合：

```
(and ATOM1 ... ATOM_N)
```

2.2.2 问题定义

基于 2.2 所提到的问题定义，问题定义中包括了在问题实例中出现的对象、初始状态的描述和目标状态的描述。一般的问题定义格式如下示例：

```
(define (problem PROBLEM_NAME)
  (:domain DOMAIN_NAME)
  (:objects OBJ1 OBJ2 ... OBJ_N)
  (:init ATOM1 ATOM2 ... ATOM_N)
  (:goal CONDITION_FORMULA)
  )
```

原子的实例化，就是谓词的参数变量都被替换为问题实例中的对象或者常量。

初始状态描述（:init）就是一个所有在初始状态中被实例化为真的原子式的列表，除此之外所有其他的原子式均为假。目标状态描述（:goal）的格式和动作前提条件的格式一样。在初始状态描述和目标状态描述中所有用到的谓词都必须是相关领域定义中已经定义的谓词。

2.2.3 PDDL 版本迭代

① PDDL 1.2

PDDL 1.2 是 1998 年第 1 届以及 2000 年第 2 届国际智能规划竞赛 IPC 的官方语言[23]。作为 PDDL 的最初版本，PDDL 1.2 介绍了该语言的基本概念，PDDL 拥有类似 Lisp 语言的语法结构。

② PDDL 2.1

PDDL 2.1 是 2002 年第 3 届国际智能规划竞赛 IPC 的官方语言[24]。基于 PDDL 1.2， PDDL 2.1 增加了对时序规划（temporal planning）、度量规划（plan-metrics）、数值变量（numeric fluent）以及持续动作（durative action）的支持[25]。

动作的持续意味着动作从开始发生到产生动作效果需要经过一段时间，而不是瞬间完成的。

③ PDDL 2.2

PDDL 2.2 是 2004 年第 4 届国际智能规划竞赛 IPC 中确定性轨迹的官方语言[26]。在 PDDL 2.2 版本中，增加了对派生谓词（derived predicate）、时间初始文字（timed initial literal）的支持[27]。

④ PDDL 3.0

PDDL 3.0 是 2006 年第 5 届国际智能规划竞赛 IPC 中确定性轨迹的官方语言，引入了状态轨迹约束（state-trajectory constraint）、偏好（preference）[28][29][30]。

⑤ PDDL 3.1

PDDL 3.1 是 2008 年第 6 届和 2011 年第 7 届国际智能规划竞赛 IPC 中确定性轨迹的官方语言，在 PDDL 3.1 中，引入了对象变量（object-fluent）[31][32]。

2.3 QT 框架

人机混合智能规划平台是一个 GUI 应用程序，基于 QT 5 开发，开发环境为 Qt Creator。

QT 是一个跨平台的桌面端、嵌入式和移动端应用程序开发框架。支持的平台包括 Linux、OS X、Windows、VxWorks、QNX、Android、IOS、BlackBerry 以及 Sailfish OS 等等。

就其本身而言，QT 并不是一门编程语言，它是一个使用 C++编写的框架。在 QT 中，为了扩充 C++的语言特性，例如信号和槽（Signals & Slots），QT 使用了预处理器 MOC（Meta-Object Compiler）。在编译步骤之前，MOC 首先解析使用 QT 扩展的 C++（Qt-extended C++）编写的源文件，然后基于这些被解析的源文件产生标准的 C++源文件。因此 QT 框架本身以及任意使用了 QT 框架的应用程序或者库都可以被标准的 C++编译器编译，例如 Clang、GCC、ICC、

MinGW 以及 MSVC。

2.4 VIZ 可视化建模

由 Jindřich Vodrážka 等人设计开发的 VIZ 是一个规划领域可视化建模工具，支持非知识工程师用户使用简易的、用户友好的 GUI 进行领域建模并直接导出 PDDL 文件[14]，能够以一种非常直观的方式，利用简单且显而易见的图形来建模 PDDL 领域模型。

2.4.1 模型元素分类

VIZ 的模型元素分类基于面向对象编程（object oriented programming）和一阶逻辑式（formalism of first order logic, FOL），划分为以下 4 类元素：

1) 类（class）：表示包含所有属于该类的对象（object）的公共属性的基类，可以被理解为对象的集合，支持类的继承。

2) 对象（object）：表示类的一种特定实例。

3) 变量（variable）：可以指向任意的特定类的实例对象。

4) 谓词（predicate）：标识一个给定的规划领域的一阶逻辑原子声明。

2.4.2 模型设计层次

VIZ 把规划领域模型设计划分为 3 个有递进次序的抽象层：

① 声明类（class）和谓词（predicate）。

② 使用变量（variable）和预先声明的谓词（predicate）定义规划的动作。

③ 使用对象（object）和预先声明的谓词（predicate）定义规划的问题。

规划领域模型设计层次的第 1 层、第 2 层用于描述领域定义，第 3 层用于描述问题定义。其中第 2 层和第 3 层均依赖于在第 1 层中声明的类（class）和谓词（predicate）。

2.4.3 声明表达语言的建模

声明表达语言对应于 2.4.2 中模型设计层次的第 1 层，声明类（class）和谓词（predicate）。

在这部分建模中，使用 QT 的图形元素（QGraphicsItem）表示 PDDL 领域模型元素，两者的映射关系如表 2-1。其中线条用于关联模型元素，模型元素允许的关联如表 2-2。

表 2-1 QT 图形元素与 PDDL 领域模型元素的映射关系

QT 图形元素	PDDL 领域模型元素
矩形（QGraphicsRectItem）	类（class）
椭圆（QGraphicsEllipseItem）	谓词（predicate）
线条（QGraphicsLineItem）	模型元素关联（purpose）
文本（QGraphicsTextItem）	模型元素的名字（name）

表 2-2 QT 线条与 PDDL 领域模型元素关联的映射关系

QT 线条起点	QT 线条终点	PDDL 领域模型元素关联
矩形（QGraphicsRectItem）	矩形（QGraphicsRectItem）	类的继承，线条起点对应元素为父类，线条终点对应元素为子类，不允许循环继承和多重继承
椭圆（QGraphicsEllipseItem）	矩形（QGraphicsRectItem）	谓词的参数关联，参数在参数列表中的次序对应线条关联的顺序

2.4.4 定义规划动作的建模

定义规划动作对应于 2.4.2 中模型设计层次的第 2 层，使用变量（variable）和预先声明的谓词（predicate）定义规划的动作。

在这部分建模中，QT 的图形元素（QGraphicsItem）与 PDDL 领域模型元素之间的映射关系如表 2-3。其中线条用于关联模型元素，模型元素允许的关联如表 2-4。

表 2-3 QT 图形元素与 PDDL 领域模型元素的映射关系

QT 图形元素	PDDL 领域模型元素
矩形（QGraphicsRectItem）	变量（variable）
椭圆（QGraphicsEllipseItem）	谓词（predicate）
线条（QGraphicsLineItem）	模型元素关联（purpose）
文本（QGraphicsTextItem）	模型元素的名字（name）

表 2-4 QT 线条与 PDDL 领域模型元素关联的映射关系

QT 线条起点	QT 线条终点	PDDL 领域模型元素关联
椭圆（QGraphicsEllipseItem）	矩形（QGraphicsRectItem）	谓词的参数关联，参数在参数列表中的次序对应线条关联的顺序

不同于 2.4.3 声明表达语言的建模，在定义规划动作的建模中，谓词被划分到 3 个不同的集合中，分别对应动作的前提条件、动作的正效果、动作的负效果，而且对应的椭圆图形元素的外圈轮廓也使用不一样的颜色标识，对应关系如表 2-5。

表 2-5 椭圆元素颜色和谓词集合的对应关系

QT 椭圆元素颜色代码	谓词集合
#3CDC84	动作前提（precond）
#0F74DC	动作正效果（effect^{+}）
#FF5753	动作负效果（effect^{-}）

同时，三个谓词集合之间存在互斥关系，不互斥的谓词集合可以同时对应同一个椭圆图形元素（即包含多个不同颜色的外圈轮廓），互斥关系如表 2-6。

表 2-6 谓词集合互斥关系

谓词集合	谓词集合	互斥原因
动作前提（precond）	动作正效果（effect^{+}）	无意义
动作正效果（effect^{+}）	动作负效果（effect^{-}）	矛盾

2.4.5 定义规划问题的建模

定义规划问题对应于 2.4.2 中模型设计层次的第 3 层，使用对象（object）和预先声明的谓词（predicate）定义规划的问题。

在这部分建模中，QT 的图形元素（QGraphicsItem）与 PDDL 领域模型元素的映射关系如表 2-7。其中线条用于关联模型元素，模型元素允许的关联如表 2-8。

表 2-7 QT 图形元素与 PDDL 领域模型元素的映射关系

QT 图形元素	PDDL 领域模型元素
矩形（QGraphicsRectItem）	对象（object）
椭圆（QGraphicsEllipseItem）	谓词（predicate）
线条（QGraphicsLineItem）	模型元素关联（purpose）
文本（QGraphicsTextItem）	模型元素的名字（name）

表 2-8 QT 线条与 PDDL 领域模型元素关联的映射关系

QT 线条起点	QT 线条终点	PDDL 领域模型元素关联
椭圆（QGraphicsEllipseItem）	矩形（QGraphicsRectItem）	谓词的参数关联，参数在参数列表中的次序对应线条关联的顺序

在定义规划问题的建模中，谓词被划分到 2 个互斥的集合，分别对应初始状态、目标状态，对应的椭圆图形元素也用不一样的颜色标识，对应关系如表 2-9。

表 2-9 椭圆元素颜色和谓词所属状态的对应关系

QT 椭圆元素颜色代码	谓词所属状态
#75F4AB	初始状态（initial state）
#FA8C80	目标状态（goal state）

2.5 规划器

智能规划的目标是找出可以引导初始状态转移到目标状态的动作序列。智能规划与调度系统需要可以实现智能规划目标的机制，换句话说，也就是它需要规划器（Planner）。

在平台中，给定领域定义 PDDL 文件和问题定义 PDDL 文件作为规划器的输入，使用规划器对规划问题进行求解，并对规划器的输出进行解析产生规划

文件，然后展示详细的规划结果报告。下面介绍在人机混合智能规划平台的开发过程中所用到的规划器。

2.5.1 Metric-FF

Metric-FF 是由 Joerg Hoffmann 开发的一个领域无关智能规划系统[33]。该系统是 FF 规划器（结合动作描述语言 ADL）的扩展[34]，增加了对数值状态变量（numerical state variables）的支持，更加准确的说是增加了对 PDDL 2.1 第 2 层特性的子集的支持[35]。Metric-FF 使用 C 语言开发，它参加了 2002 年第 3 届国际智能规划竞赛 IPC，具有非常有竞争力的性能表现。

平台使用的 Metric-FF 版本是 Metric-FF-v2.1，发布时间为 2002 年，发布平台为 Windows。

2.5.2 Blackbox

Blackbox 是由 Henry Kautz 开发的一个可满足性智能规划系统，该系统首先将 STRIPS 型问题转化为布尔可满足性问题（Boolean satisfiability problems），然后使用各种先进的可满足性引擎（satisfiability engines）来求解问题[36]。Blackbox 的前端采用的是系统 graphplan 的修改版本。Blackbox 非常具有灵活性，可以指定多种规划求解引擎来进行问题求解，所以 Blackbox 具有可以在非常大范围种类的规划问题上有效运行的能力。

平台使用的 Blackbox 版本是 Blackbox 42，发布时间为 2006 年，发布平台为 Windows。

2.5.3 LPG

LPG （Local search for Planning Graphs）是一个基于局部搜索和规划图的规划器，它兼容 PDDL 2.1[37]。该规划器既能处理规划产生（plan generation）问题，也能处理规划修改（plan adaptation）问题。LPG 的搜索空间包含动作图（action graphs）、规划图中表示局部规划的特定子图。

平台使用的 LPG 版本是 LPG-td，发布时间为 2004 年，发布平台为 Windows。LPG-td 参加了 2004 年第 4 届国际智能规划竞赛 IPC。LPG-td 是 LPG 的扩展版

本，增加了对 PDDL 2.2 中派生谓词（derived predicate）和时间初始文字（timed initial literal）的支持[38]。

2.6 规划验证器

和规划器一样，规划分析是完整的智能规划系统中不可或缺的一部分。在智能规划与调度研究领域中，规划验证是规划分析的主要发展方向之一。

在平台中，给定领域定义 PDDL 文件、问题定义 PDDL 文件和对应的规划文件作为规划验证器（Validator）的输入，对规划的动作序列进行验证，并对规划验证器的输出进行解析，获得并展示验证结果，如果验证失败，还可以提供关于有缺陷的动作的修复建议。下面介绍在人机混合智能规划平台的开发过程中所用到的规划验证器。

2.6.1 VAL

VAL 是由 Richard Howey 等人开发的一个规划验证工具[18]，支持 PDDL 3 和 PDDL+[39]。VAL 支持的基本特性包括：

1) 连续的效果（Continuous Effects）

2) 派生谓词（Derived Predicates）

3) 时间初始文字（Timed Initial Literals）

4) 过程（Processes）

5) 外生事件（Exogenous Events）

6) 规划修复建议（Plan Repair Advice）

7) LaTeX 报告

第 3 章　算法与理论

本章将详细介绍人机混合智能规划平台中涉及到的主要算法与理论，包括领域知识一致性检查中的领域约束、领域模型微调中的 HILP 智能规划。

3.1 领域知识一致性检查

基于 STRIPS 型规划领域模型，本节将针对领域知识一致性检查建立约束，这些约束为实现语法约束检查模块提供理论支持。

为更加清晰地描述本节的内容，首先给出本节将使用到的概念定义及描述，然后分别介绍平台所涉及到的各种约束并建立约束公式。

3.1.1 基本概念定义及描述

一个规划领域被定义为：$\Sigma = (S, A, \Gamma)$，其中 S 为世界状态的集合，A 是动作模型的集合，Γ 是确定性转移函数 $S \times A \rightarrow S$。A 中的每一个动作模型，都包含 3 部分：包含 0 个或者多个参数的动作名字、在动作执行之前应该被满足的前提条件的集合、动作执行后所产生的后置效果的集合。

一个规划问题可以被定义为：$\mathcal{P} = (\Sigma, s_0, g)$，其中 s_0 是初始状态，g 是目标状态。规划问题的一个解是一个动作序列：$(a_0, a_1, \ldots, a_n)$，也就是规划，是将 s_0 映射为 g。每个 a_i 是一个动作模式，包含一个动作名字、0 个或者多个参数，例如（*board-truck ?d - driver ?t - truck ?loc – location*）。

作为扩展，一个规划轨迹被定义为：$t = (s_0, a_0, s_1, a_1, \ldots, s_n, a_n, g)$，其中 $s_1, \ldots, s_n$ 是部分可观察的中间状态。如果一个状态的部分命题是缺失的那么这个状态就被称为是部分的，当所有的命题都缺失了，那么这个状态就是空的。

定义动作模式集合 A，谓词集合 P，类型集合 C，规划轨迹集合 T。定义谓词 p，谓词 p 的名字为 Id(p)，参数集合为 Para(p)。定义动作 a，动作 a 的参数集合为 Para(a)，正前提条件集合 Pos(a)，负前提条件集合 Neg(a)，增加效果集合 Add(a)，删除效果结合 Del(a)。定义类 c，类 c 的祖先类集合为 Family(c)，父类

集合为 Parent(c)。

3.1.2 谓词约束

在平台的可视化建模中，可以声明重载谓词，任意两个名字相同的谓词为重载谓词，它们的参数类型和参数次序可以不相同，但是参数个数必须相同，针对该规则建立约束公式如 3-1 所示。

$$\mathrm{Id}(p_1) = \mathrm{Id}(p_2) \Rightarrow |\mathrm{Para}(p_1)| = |\mathrm{Para}(p_2)|.\ \forall p_1, p_2 \in P \qquad (3\text{-}1)$$

3.1.3 类型约束

在可视化建模中声明类时，允许类之间的继承关系，但是不允许存在循环继承以及多重继承，针对该规则建立约束公式如 3-2 所示。

$$|\mathrm{Parent}(c)| = \{1, 0\}, c \notin \mathrm{Family}(c).\ \forall c \in C \qquad (3\text{-}2)$$

3.1.4 状态约束

如果一个谓词 p 频繁地刚好出现在动作 a 执行之前，那么 p 很有可能属于 a 的前提条件，如式 3-3 所示。

$$\mathrm{Para}(p) \subseteq \mathrm{Para}(a) \Rightarrow p \in \mathrm{Pos}(a) \qquad (3\text{-}3)$$

条件 Para(p)⊆Para(a)是 STRIPS 型模型所要求的，动作的参数必须包含其前提条件以及后置效果的所有参数。

同样地，如果一个谓词 p 频繁地刚好出现在动作 a 执行之后，那么 p 很有可能属于 a 的后置效果，如式 3-4 所示。

$$\mathrm{Para}(p) \subseteq \mathrm{Para}(a) \Rightarrow p \in \mathrm{Add}(a) \qquad (3\text{-}4)$$

3.1.5 动作约束

在可视化建模中，必须保证被建模的动作模型是一致的不矛盾的。

一个动作 a 不应该同时产生两个冲突的后置效果 p 以及¬p，对此建立约束公式如 3-5 所示。

$$p \in \mathrm{Add}(a) \Rightarrow p \notin \mathrm{Del}(a) \qquad (3\text{-}5)$$

一个动作 a 的前提条件中不应该同时包含两个冲突的条件 p 以及¬p，对此建立约束公式如 3-6 所示。

$$p \in \mathrm{Pos}(a) \Rightarrow p \notin \mathrm{Neg}(a) \tag{3-6}$$

一个谓词 p 不应该同时出现在动作 a 的前提条件和后置效果中，因为这样的定义是无意义的，如式 3-7、3-8 所示。

$$p \in \mathrm{Pos}(a) \Rightarrow p \notin \mathrm{Add}(a) \tag{3-7}$$

$$p \in \mathrm{Neg}(a) \Rightarrow p \notin \mathrm{Del}(a) \tag{3-8}$$

动作 a 的所有参数类型必须属于声明的类型集合 C，如式 3-9 所示。

$$\mathrm{Para}(a) \subseteq C. \forall a \in A \tag{3-9}$$

同理动作 a 的前提条件和后置效果中的所有谓词 p 必须是声明的谓词集合 P 中的实例，包括谓词的参数列表、参数类型等，如式 3-10 所示。

$$p \in \{\mathrm{Pos}(a), \mathrm{Neg}(a), \mathrm{Add}(a), \mathrm{Del}(a)\} \Rightarrow p \in P \tag{3-10}$$

3.1.6 规划问题约束

在可视化建模中，必须保证被建模的问题模型是一致的不矛盾的。

一个谓词 p 不应该同时出现在规划问题的初始状态和目标状态中，因为这样的定义是无意义的，如式 3-11 所示。

$$p \in s_0 \Rightarrow p \notin g \tag{3-11}$$

与 3.1.5 同理，规划问题的所有参数类型必须属于声明的类型集合 C，所有的谓词 p 必须是声明的谓词集合 P 的实例。

3.1.7 规划约束

每一个规划轨迹中的动作 a_i 都是可执行的，也就是说动作 a_i 在执行之前它的所有前提条件都能够得到满足，表示为式 3-12。

$$p \in \mathrm{Pos}(a_i) \Rightarrow p \in \mathrm{Exe}(i-1) \tag{3-12}$$

其中 $p \in \mathrm{Exe}(i\text{-}1)$意味着 p 要么存在于初始状态中，而且不被动作序列所删

除，要么是被某个在 a_i 之前的动作 a'添加的，而且不被动作 a'与 a_i 之间的任意动作删除。Exe(i-1)意味着执行动作序列(a_1, …, a_{i-1})后的中间状态。

同理，考虑任意一个可观察的中间状态 o_i，o_i 的每一个谓词 p 要么存在于初始状态，要么被某个动作新添加的，如式 3-13 所示。

$$p \in o_i \Rightarrow p \in \text{Exe}(i-1) \tag{3-13}$$

3.2 HILP 人在回路智能规划

平台的领域模型微调模块充分利用 HILP 的思想，允许用户根据规划验证的结果人为地修改规划领域模型，达到生成更加准确的规划解的目的。

结合 HILP，可以从以下 3 个方面描述领域模型微调模块根据规划验证的结果对领域模型进行微调的算法[40]：

1) 合作模式（Cooperation Modality）

   交互（Interaction），根据规划验证的结果，向用户提供修改建议，用户选择相应的修改选项，系统执行修改选项。

2) 通信模式（Communication Modality）

   通过自定义接口（Through custom interfaces），直接对规划领域进行修改，而不是对规划动作序列进行修改。

3) 通信内容（What is Communicated）

   新模型（New model），对规划领域进行修改，允许用户创建新动作，或者修改已有动作。

第 4 章　系统需求建模

为了更加清晰地描述人机混合智能规划平台的业务需求，本章首先描述平台的系统概述，对系统的功能性需求和非功能性需求进行分析和描述，然后采用用例图对功能性需求进行用例析取，并采用用例规约对关键用例进行详细描述。

4.1 需求分析

4.1.1 系统概述

平台是一个人机混合智能规划平台，实现基于领域知识库的规划领域设计可视化建模，提高可视化建模的效率，实现利用规划结果微调领域模型。

平台包含 7 大功能模块，分别是可视化建模模块、领域知识库模块、语法约束检查模块、数据转换模块、规划模块、规划验证模块、领域模型微调模块：

1) 可视化建模模块，采用 MVC 设计模式，负责规划领域模型建模，可视化图形信息支持 XML 配置化。

2) 领域知识库模块，负责维护、管理领域知识，包含两类基本领域知识模板，支持 XML 配置化。

3) 语法约束检查模块，负责 PDDL 基本语法约束检查。

4) 数据转换模块，负责语言和相关输入输出文件的转换，支持输入输出 XML 文件，支持输出 PDDL 文件，支持 XML 转换为 PDDL。

5) 规划模块，集成多种规划器，规划器的设置支持 XML 配置化，负责执行规划并展示规划结果，规划结果支持 XML 配置化。

6) 规划验证模块，集成规划验证器，验证器的设置支持 XML 配置化，负责执行规划验证并展示详细的验证信息。

7) 领域模型微调模块，对于有缺陷的规划提供修复建议并支持用户交互

式的修改规划领域模型，包含可视化模型修改、文本编辑修改。

4.1.2 功能性需求

本节分析并描述平台的主要功能性需求，包括规划领域可视化建模、领域知识库、约束检查、数据转换、规划求解、规划验证、领域模型微调：

1) 规划领域可视化建模

基于 2.4 中介绍的可视化建模方法，用户可以在图形用户界面中编辑、管理 QT 的图形元素，图形用户界面包含 3 个编辑界面，分别是声明表达语言的界面、定义规划动作的界面和定义规划问题的界面，其中声明表达语言是必须的，定义规划动作和定义规划问题是可选的。此外，用户还可以在编辑界面外单独编辑领域模型的属性信息（领域的名字、领域模型的描述性注释）。用户可以使用平台的数据转换功能，导出领域模型的属性信息、3 个编辑界面的领域模型信息为一个汇总的 XML 文件。

2) 领域知识库

领域知识库中包含了两类领域知识模板，分别是类模板、谓词模板，类模板就是类型的标记符，谓词模板的格式为谓词标记符及其参数，以空格分开（[*identifier*]([*spaces*][*parameter's type*])*）。领域知识库仅与可视化建模中的声明表达语言编辑界面进行交互，为在该编辑界面中声明类或者声明谓词提供智能提示和自动补全。在声明表达语言编辑界面中，在用户声明类时会根据用户的输入筛选出领域知识库中符合条件的类模板，提供智能提示，用户可以通过实例化类模板或者自定义类来声明类。在用户声明谓词时也会根据用户的输入筛选出领域知识库中符合条件的谓词模板，提供智能提示，用户可以通过实例化谓词模板或者自定义谓词来声明谓词，如果被声明的谓词包含了谓词参数（谓词名字与谓词参数之间以空格分割），那么编辑界面中会一并生成所有的谓词参数对应的矩形图形元素以及关联的线条图形元素。领域知识库初始化是空的，在声明类或者声明谓词时，可以自动添加新

的类或者谓词到知识库中，更新知识库的模板。另外，用户也可以直接在开发者选项中编辑领域知识库，包括添加、修改或者删除领域知识模板。

3) 约束检查

基于 3.1 领域知识一致性检查中的领域约束，用户在编辑界面中进行规划领域建模时，可以进行规划领域语法约束检查。在声明表达语言编辑界面中，可以检查类和谓词的定义，对于谓词，可以检查所有重载的谓词的参数个数是否相同。在定义规划动作和定义规划问题的编辑界面中，用户可以检查动作定义、问题定义与声明表达语言界面中的领域定义的一致性，包括所有的类实例（动作定义中的变量、问题定义中的对象）和谓词实例，其中对于谓词实例可以检查谓词的参数列表。约束检查可以输出完备的检查信息、领域统计信息。

4) 数据转换

用户在进行规划领域建模时，可以使用平台的数据转换功能，可以导出领域模型为 XML 文件。用户也可以导入领域模型 XML 文件，并在领域模型编辑界面中进行展示和编辑。用户可以将编辑界面中的领域模型导出为 PDDL 文件，包括领域定义 PDDL 文件和问题定义 PDDL 文件。

5) 规划求解

用户可以使用领域定义 PDDL 文件和问题定义 PDDL 文件进行规划。在规划界面中，用户可以选择用于进行规划的 PDDL 文件，可以是默认的由编辑界面可视化建模生成的 PDDL 文件或者自定义的已存在的 PDDL 文件，而且还可以对自定义的已存在的 PDDL 文件进行文本编辑。用户可以选择用于进行规划的规划器，进行规划后，在规划界面中可以显示详细的规划报告。另外，用户可以在开发者选项中对规划器进行管理和设置。

6) 规划验证

用户可以使用领域定义 PDDL 文件、问题定义 PDDL 文件和相应的规划文件（txt 后缀格式）进行规划验证。在规划界面中，用户可以选择用于进行规划验证的文件，对于领域定义 PDDL 文件和问题定义 PDDL 文件可以是默认的由编辑界面可视化建模生成的 PDDL 文件或者自定义的已存在的 PDDL 文件；对于规划文件可以是默认的由规划求解模块生成的规划文件或者规划求解失败后用户按照设想自定义的规划文件，还可以是自定义的已存在的规划文件，而且还可以对自定义的已存在的文件进行文本编辑。用户进行规划验证后，会展示规划验证界面，在规划验证界面中会展示详细的规划验证报告，包括规划的动作序列信息、动作序列中动作的信息（前提条件、后置效果等）、世界状态的改变（动作应用前的状态、动作应用后的状态）。如果规划验证为不合理，还可以在相应的有缺陷的动作信息中展示产生缺陷的原因，提供并展示相关的修复建议。

7) 领域模型微调

基于 3.2 的 HILP 智能规划，用户在进行规划验证后，如果规划被验证为不合理的，可以提供并展示相关的修复建议。用户可以根据系统提供的规划修复建议，自主地选择修复选项来执行修复，对于规划动作序列中存在缺陷的动作中不满足的前提条件，规划验证模块提供了 2 个可选的修复选项。一个是自动地直接在规划领域中定义一个新的动作，该动作没有任何的前提条件，只有唯一的一个后置效果，那就是修复建议中提示的有缺陷的动作所不满足的前提条件；另一个是允许用户自主的修改规划领域定义。另外，如果输入规划验证的领域定义 PDDL 文件是默认的，那么用户的修复选项会作用于可视化建模的定义规划动作编辑界面；如果输入规划验证的领域定义 PDDL 文件是自定义的已存在的，那么用户的修复选项会直接作用于领域定义 PDDL 文件，以文本编辑的方式。

4.1.3 非功能性需求

本节分析并描述平台的非功能性需求，其中包括易用性、可维护性、可扩

展性。

1) 易用性

作为一个人机混合智能规划平台，易用性关系到用户使用系统进行开发的难易程度，一个易用的系统应该具有用户友好的图形界面、人性化的操作方式。平台充分考虑到了易用性，在规划领域可视化建模中，规划领域的编辑界面简洁友好，操作直观简单，有完备的容易理解的说明性提示，同时随着领域模型的增大，用户可以缩放旋转编辑界面，以获得更好的对领域模型的概观，另外为了方便用户展示可视化的领域模型，还可以导出编辑界面的截图。

2) 可维护性

可维护性关系到开发人员对平台进行维护、功能更改的难易程度。平台总体上采用模块化的架构，包含可视化建模模块、领域知识库模块、语法约束检查模块、数据转换模块、规划模块、规划验证模块、领域模型微调模块。其中在可视化建模模块中采用 MVC 设计模式，包含模型（Model）、视图（View）、控制器（Controller）。

3) 可扩展性

在平台中，一方面，可视化建模的领域模型信息支持 XML 标签化，对于添加新的 PDDL 语言特性非常方便。另一方面，平台集成了多种规划器和验证器的可执行程序，规划器和验证器的配置都支持 XML 配置化，对于新添加、设置规划器和验证器都非常方便，具有良好的可扩展性。

4.2 用例分析

4.2.1 系统用例

本节结合 4.1.2 的功能性需求分析和描述，分别对平台的主要功能模块进行用例析取，包括可视化建模模块、领域知识库模块、语法约束检

查模块、数据转换模块、规划模块、规划验证模块、领域模型微调模块，并给出用例图。

1) 可视化建模模块

用户可以分别在声明表达语言的界面、定义规划动作的界面和定义规划问题的界面中编辑、管理 QT 的图形元素，从而声明表达语言、定义规划动作以及定义规划问题，每一个相应的界面中，用户可以添加谓词、添加关联线条、删除元素，不同的是在声明表达语言界面中可以添加类，在定义规划动作界面中可以添加变量，在定义规划问题界面中可以添加对象。此外，用户还可以在编辑界面外编辑领域属性（领域的名字、领域模型的描述性注释）。如图 4-1 所示可视化建模模块用例图。

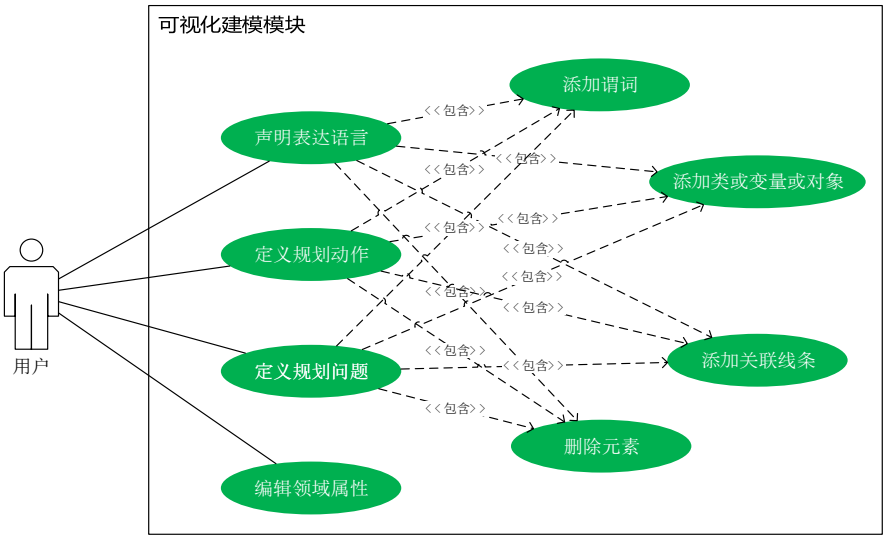

图 4-1 可视化建模模块用例图

2) 领域知识库模块

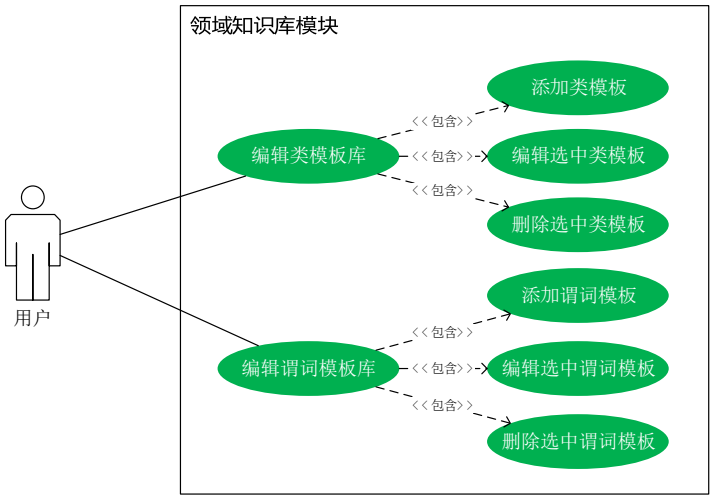

图 4-2 领域知识库模块用例图

用户可以在开发者选项中编辑领域知识库，包括编辑类模板库和谓词模板库。可以添加新模板、编辑以及删除选中模板。如图 4-2 所示领域知识库模块用例图。

3) 语法约束检查模块

用户在编辑界面中进行规划领域建模时，可以检查规划领域语法约束以及总结领域语法。如图 4-3 所示语法约束检查模块用例图。

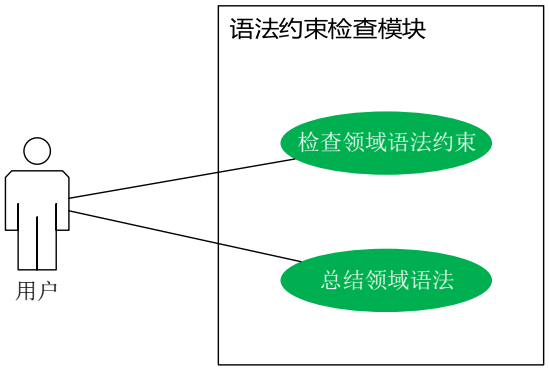

图 4-3 语法约束检查模块用例图

4) 数据转换模块

用户在进行规划领域建模时，可以使用数据转换功能，导出领域模型为 XML 文件，用户也可以导入领域模型 XML 文件。用户可以将编辑界面中的领域模型导出为 PDDL 文件，包括领域定义 PDDL 文件和问题定义 PDDL 文件。如图 4-4 所示数据转换模块用例图。

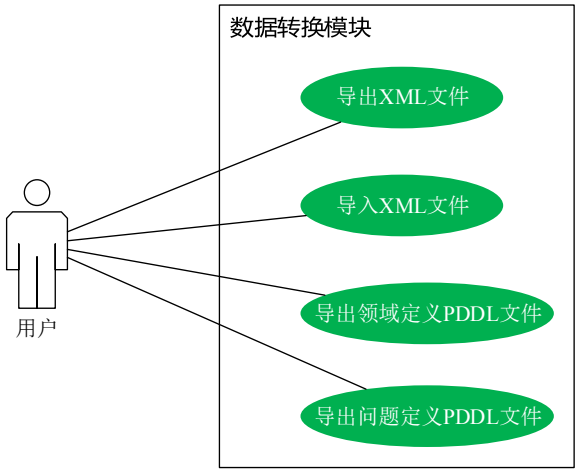

图 4-4 数据转换模块用例图

5) 规划模块

在规划界面中，用户可以配置用于执行规划的 PDDL 文件（包括领域定义 PDDL 文件和问题定义 PDDL 文件），可以是默认的由编辑界面可视化建模生成的 PDDL 文件或者自定义的已存在的 PDDL 文件，而且还可以对自定义的已存在的 PDDL 文件进行文本编辑。用户可以选择规划器，然后执行规划。另外，用户可以在开发者选项中设置规划器。如图 4-5 所示规划模块用例图。

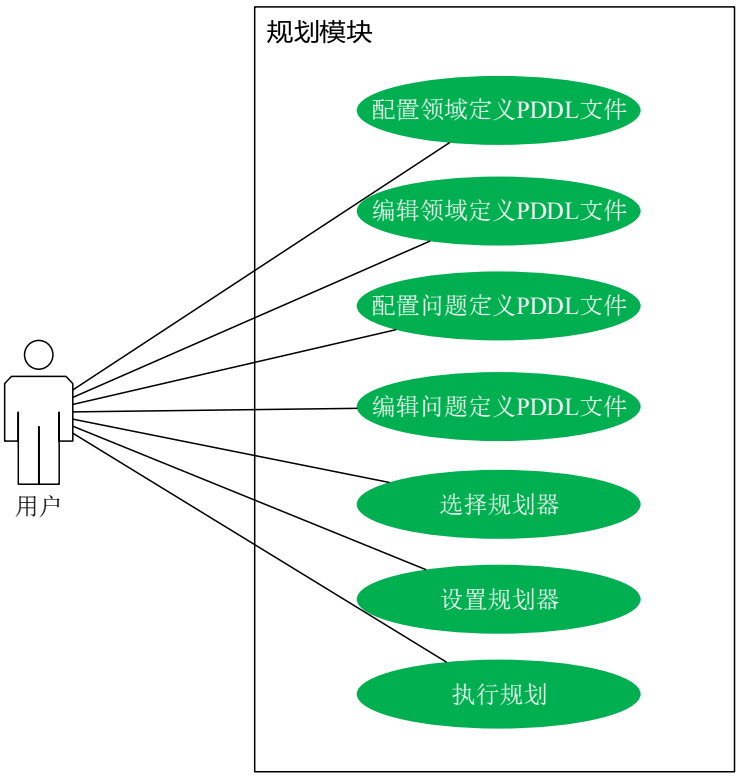

图 4-5 规划模块用例图

6) 规划验证模块

在规划界面中，用户可以配置进行规划验证的文件（包括包括领域定义 PDDL 文件、问题定义 PDDL 文件和规划文件），对于领域定义 PDDL 文件和问题定义 PDDL 文件可以是默认的由编辑界面可视化建模生成的 PDDL 文件或者自定义的已存在的 PDDL 文件；对于规划文件可以是默认的由规划求解模块生成的规划文件或者规划求解失败后用户按照设想自定义的规划文件，还可以是自定义的已存在的规划文件，而且还可以对自定义的已存在的文件进行文本编辑。配置好相关文件后，用户可以执行规划验证。如图 4-6 所示规划验证模块用例图。

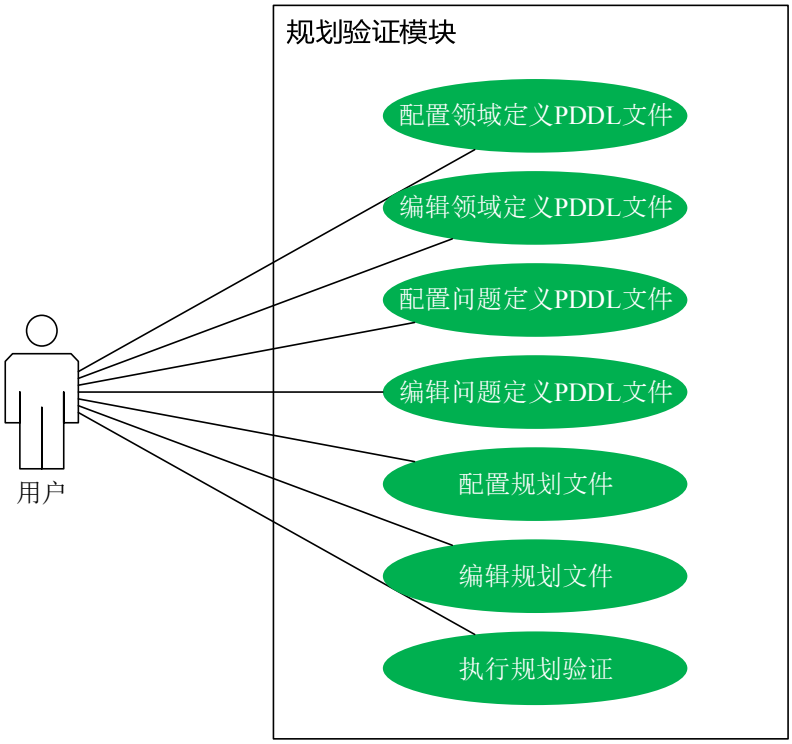

图 4-6 规划验证模块用例图

7) 领域模型微调

在规划验证界面，如果规划被验证为不合理的，用户可以根据修复建议修正领域模型。对于有缺陷的动作中不满足的前提条件，规划验证模块提供了 2 个可选的修复选项，其中一个选项是自动地直接在规划领域中创建新的动作；另一个选项是允许用户自主的修改已有动作。如图 4-7 所示领域模型微调模块用例图。

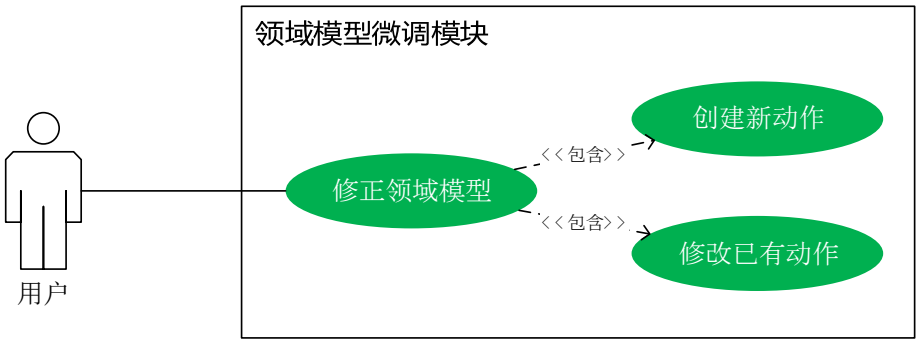

图 4-7 领域模型微调模块用例图

4.2.2 关键用例详述

本节使用用例规约的方式详述几个关键用例的交互方式，包括添加谓词用例、添加关联线条用例、执行规划用例、执行规划验证用例、创建新动作用例、修改已有动作用例。

1) 添加谓词用例

表 4-1 添加谓词用例规约

用例名称	添加谓词
参与者	用户
用例描述	用户在可视化建模中通过创建椭圆图形元素来添加谓词
前置条件	用户将系统的编辑界面切换到添加谓词模式
基本事件流	1.用户单击编辑界面的空白处，进行如下相应流程： 1) 若系统处于声明表达语言编辑界面，则需要用户输入谓词，包括谓词名称、谓词的参数列表，均以空格隔开，用户可以点击输入提示框中的谓词进行补全，点击确定。 2) 若系统处于定义规划动作编辑界面，则需要用户在下拉选择框中选择一个谓词以及在复选框中选择谓词的集合，点击确定。 3) 若系统处于定义规划问题编辑界面，则需要用户在下拉选择框中选择一个谓词以及在复选框中选择谓词的所属状态。点击确定。
备选事件流	1a.如果系统处于声明表达语言编辑界面，用户输入谓词为空，用户点击确定，系统提示谓词名称不能为空。 1b.如果系统处于声明表达语言编辑界面，用户输入谓词不符合输入规范，用户点击确定，系统提示谓词名称格式错误。
后置条件	添加谓词成功

2) 添加关联线条用例

表 4-2 添加关联线条用例规约

用例名称	添加关联线条
参与者	用户
用例描述	用户在可视化建模中通过创建线条图形元素来添加关联，允许的关联参考 2.4 中 VIZ 可视化建模
前置条件	用户将系统的编辑界面切换到添加线条模式

基本事件流	1.用户单击编辑界面的空白处，持续按住单击按钮并拖拽， 释放点击按钮，进行如下相应流程： 1) 若线条两端均为矩形图形元素，线条关联为类的继承。 2) 如线条两端分别为矩形图形元素和椭圆图形元素，线条关联为谓词参数的声明。
备选事件流	1a.若线条两端均为矩形图形元素，两矩形之间已存在类继承关联且与新建线条关联方向相反，系统提示不允许循环继承。 1b.若线条两端均为矩阵图形元素，线条终点所指矩形已存在类继承线条关联且为线条终点，系统提示不允许多重继承。 1c.若线条的两端均为椭圆图形元素，系统提示不允许该类型线条关联。 1d.若线条两端中的任意一端指向空白处，则系统将不会执行任何操作。
后置条件	添加关联线条成功

3) 执行规划用例

表 4-3 执行规划用例规约

用例名称	执行规划
参与者	用户
用例描述	用户在规划界面中执行规划
前置条件	用户已配置领域定义 PDDL 文件，已选择规划器
基本事件流	1.用户点击执行规划按钮。
备选事件流	1a.若用户尚未配置问题定义 PDDL 文件，系统提示问题定义 PDDL 文件不能为空。
后置条件	执行规划成功，系统展示规划报告

4) 执行规划验证用例

表 4-4 执行规划验证用例规约

用例名称	执行规划验证
参与者	用户
用例描述	用户在规划界面中执行规划验证
前置条件	用户已在复选框中选择规划验证
基本事件流	1.用户点击执行规划验证按钮。
备选事件流	1a.若用户尚未配置领域定义 PDDL 文件，系统提示领域定义

	PDDL 文件不能为空。 1b.若用户尚未配置问题定义 PDDL 文件，系统提示问题定义 PDDL 文件不能为空。 1c.若用户尚未配置规划验证所需的规划文件，系统提示规划文件不能为空。
后置条件	执行规划验证成功，系统打开规划验证界面并展示详细的规划验证报告

5) 创建新动作用例

表 4-5 创建新动作用例规约

用例名称	创建新动作
参与者	用户
用例描述	用户在规划验证界面，执行修复选项创建新动作
前置条件	用户单击选中规划动作序列中用红色标记的有缺陷的动作，然后单击选中或者双击修复建议窗口中的任意一条修复建议并执行修复，系统展示修复界面
基本事件流	1.用户在修复界面的下拉选择框中选择直接创建新动作，用户点击确定，进行如下相应流程： 1) 若当前用于规划验证的领域定义 PDDL 文件为默认的由可视化建模生成的 PDDL 文件，系统跳转到可视化建模中的定义规划动作编辑界面，并打开创建新动作窗口，用户输入新动作名称，点击确定。 2) 若当前用于规划验证的领域定义 PDDL 文件为自定义的已存在的 PDDL 文件，系统跳转到领域定义 PDDL 文件的文本编辑界面，点击确定。
备选事件流	1a.如果用户在创建新动作窗口输入的动作名称为空，系统提示动作名称不能为空。 1b.如果用户在创建新动作窗口输入的动作名称不符合输入规范，系统提示动作名称格式错误。
后置条件	创建新动作成功

6) 修改已有动作用例

表 4-6 修改已有动作用例规约

用例名称	修改已有动作
参与者	用户
用例描述	用户在规划验证界面，执行修复选项修改已有动作
前置条件	用户单击选中规划动作序列中用红色标记的有缺陷的动作，然后

	单击选中或者双击修复建议窗口中的任意一条修复建议并执行修复，系统展示修复界面
基本事件流	1.用户在修复界面的下拉选择框中选择修改已有动作，用户点击确定，进行如下相应流程： 1) 若当前用于规划验证的领域定义 PDDL 文件为默认的由可视化建模生成的 PDDL 文件，系统跳转到可视化建模中的定义规划动作编辑界面。 2) 若当前用于规划验证的领域定义 PDDL 文件为自定义的已存在的 PDDL 文件，系统跳转到领域定义 PDDL 文件的文本编辑界面。
后置条件	修改已有动作成功

第 5 章　系统总体设计

本章阐述人机混合智能规划平台的系统总体设计，包括系统的架构设计、可视化建模模块架构设计以及关键用例时序图。

5.1 架构设计

5.1.1 系统架构设计

人机混合智能规划平台总体上采用模块化的架构设计，系统架构设计如图 5-1 所示。

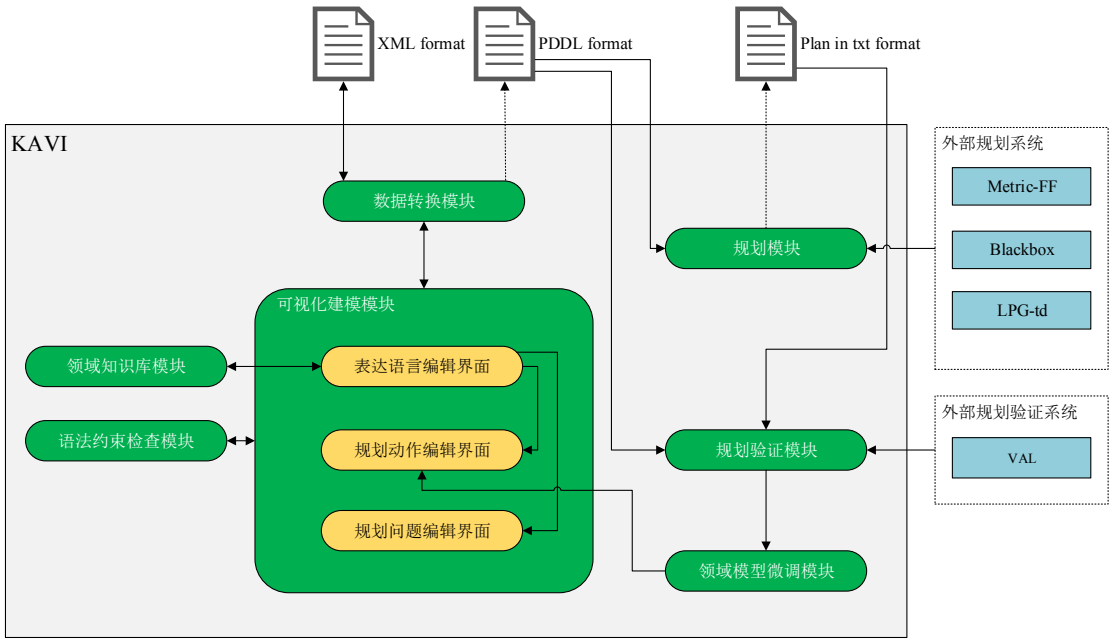

图 5-1 人机混合智能规划平台架构图

系统架构设计中包含了 7 大功能模块，包含可视化建模模块、领域知识库模块、语法约束检查模块、数据转换模块、规划模块、规划验证模块、领域模型微调模块：

1) 可视化建模模块，采用 MVC 设计模式，负责规划领域模型建模，可视化图形信息支持 XML 配置化。

2) 领域知识库模块，负责维护、管理领域知识，包含两类基本领域知识模板，支持 XML 配置化。

3) 语法约束检查模块，负责 PDDL 基本语法约束检查。

4) 数据转换模块，负责语言和相关输入输出文件的转换，支持输入输出 XML 文件，支持输出 PDDL 文件，支持 XML 转换为 PDDL。

5) 规划模块，集成多种规划器（Metric-FF、Blackbox、LPG-td），规划器的设置支持 XML 配置化，负责执行规划并展示规划结果，规划结果支持 XML 配置化。

6) 规划验证模块，集成规划验证器（VAL），验证器的设置支持 XML 配置化，负责执行规划验证并展示详细的验证信息。

7) 领域模型微调模块，对于有缺陷的规划提供修复建议并支持用户交互式的修改规划领域模型，包含可视化模型修改、文本编辑修改。

5.1.2 可视化建模模块架构设计

在 5.1.1 中给出了系统的总体架构设计，其中的可视化建模模块采用了 MVC 设计模式，包含模型（Model）、视图（View）、控制器（Controller）。

1) 视图（View）

在可视化建模中，视图主要是规划领域的编辑界面，包括声明表达语言编辑界面、定义规划动作编辑界面、定义规划问题编辑界面。基于 QT 的图形场景（QGraphicsScene），通过 QT 的事件机制把视图的更新信息传播到控制器，用户可以在编辑界面中编辑、管理 QT 的图形元素，另外还可以导出当前编辑界面的截图。

2) 控制器（Controller）

在可视化建模中，控制器主要负责处理 3 个编辑界面中的业务逻辑。基于 QT 的图形视图（QGraphicsView），负责来自视图（View）中的各种事件的分发处理，并把来自视图（View）中的图形更新信息同步到模型（Model）以及视图（View）。

3) 模型（Model）

在可视化建模中，模型主要用于维护编辑界面关联的 XML 数据，以及把模型的更新同步回视图（View）中的数据缓存。

可视化建模模块的架构设计如图 5-2 所示。

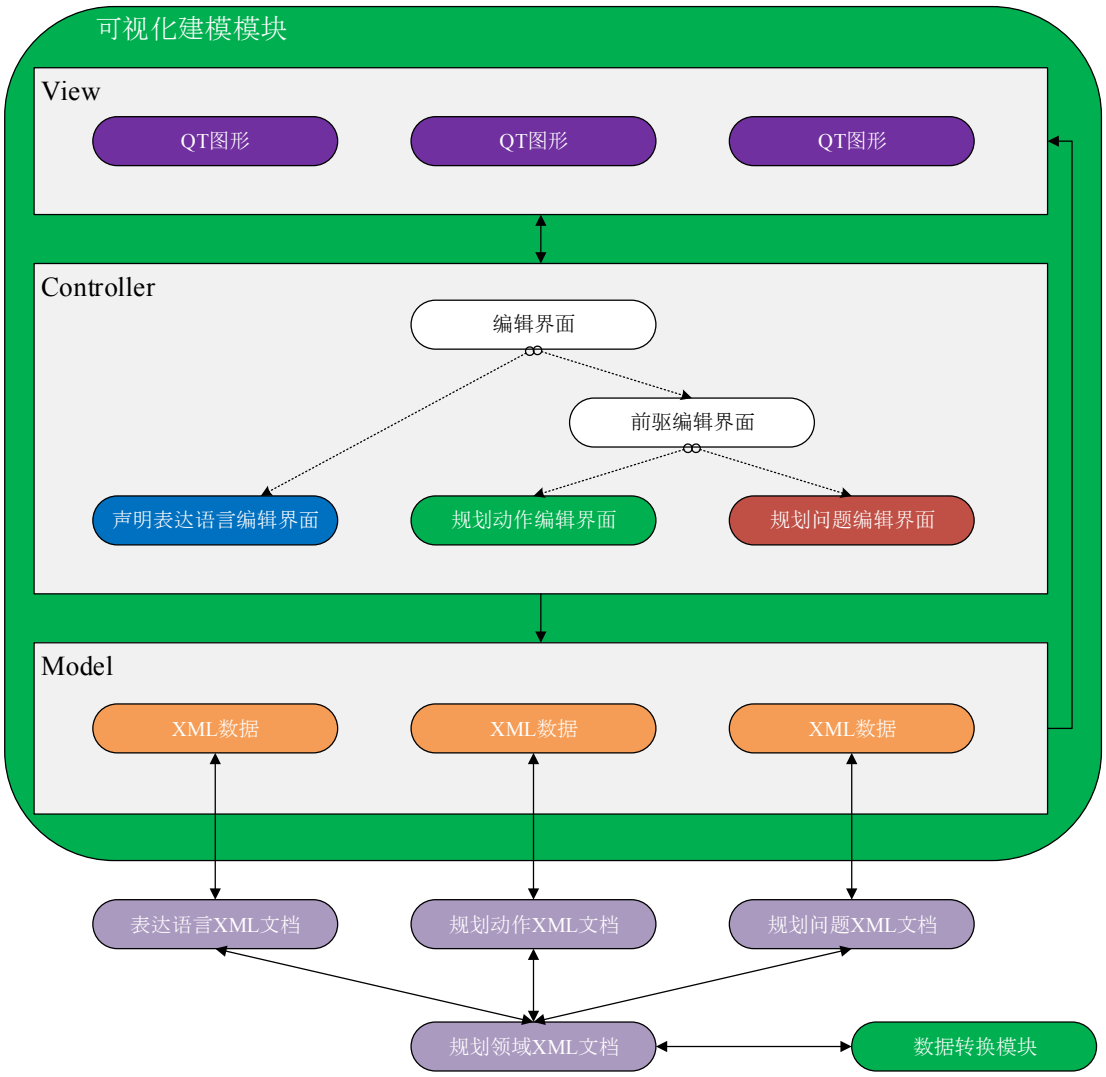

图 5-2 可视化建模模块架构图

5.2 关键用例实现

本节对 4.2.2 中的关键用例进行详细的分析，用时序图的方式来描述用例的业务功能的交互流程，并对相应的时序图进行实现描述。

5.2.1 添加谓词用例

添加谓词是可视化建模模块中声明领域模型元素的非常具有代表性的用例，本节以在声明表达语言编辑界面中添加谓词为例，其他编辑界面与此类似。添加谓词用例的时序图如图 5-3 所示。

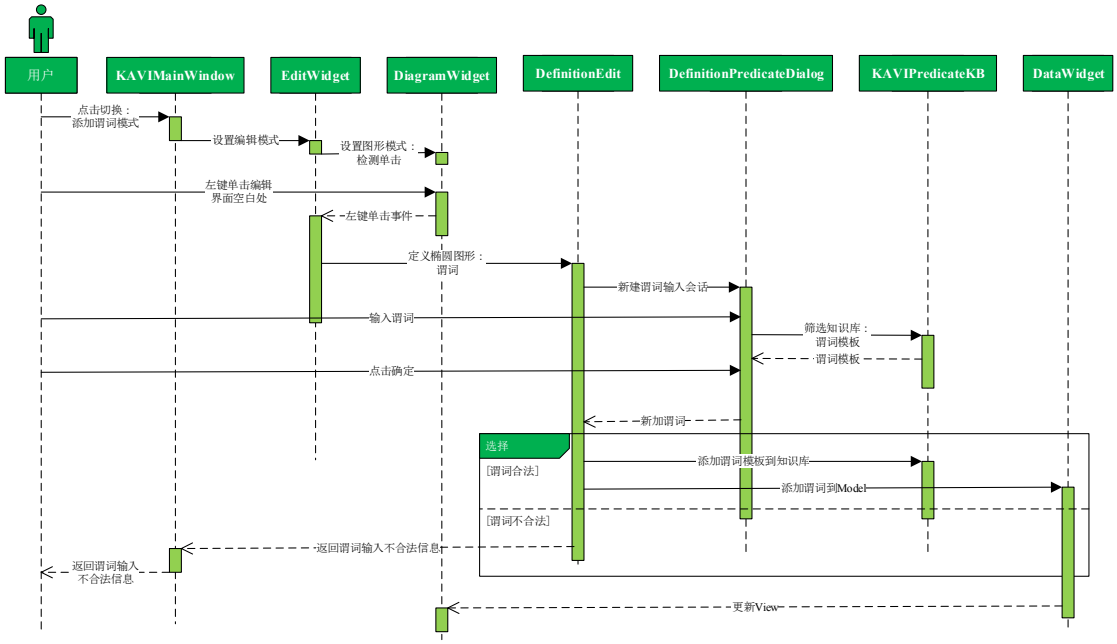

图 5-3 声明表达语言编辑界面的添加谓词用例时序图

添加谓词的实现描述如下：

1) 用户点击 KAVIMainWindow 工具栏的椭圆按钮，设置 EditWidget 的编辑模式为添加谓词模式，并设置 DiagramWidget 的图形模式为检测单击。

2) 用户左键单击编辑界面的空白处，DiagramWidget 捕获鼠标左键事件，封装点击坐标等信息到事件中并传播到 EditWidget。

3) EditWidget 分发处理鼠标左键事件，调用 DefinitionEdit 定义用椭圆图形表示的谓词，并打开输入新建谓词的会话 DefinitionPredicateDialog。

4) 用户在 DefinitionPredicateDialog 中输入谓词，根据用户的输入筛选出领域知识库的谓词模板库 KAVIPredicateKB 中符合模糊匹配的谓词模板，并返回在下拉选择框中智能提示。

5) 用户输入完毕点击确定，返回 DefinitionPredicateDialog 中所输入的谓词

到 DefinitionEdit。

6) DefinitionEdit 判断新建谓词是否合法。如果判断谓词合法，则添加新谓词模板到 KAVIPredicateKB 中，并添加谓词到 DataWidget，最后 DataWidget 会同步更新到 DiagramWidget，从而更新编辑界面。如果判断谓词不合法，则通过 KAVIMainWindow 反馈给用户谓词输入不合法的信息。

5.2.2 添加关联线条用例

添加关联线条是可视化建模模块中用于添加领域模型元素之间的关联关系的用例，本节以在声明表达语言编辑界面中添加关联线条为例，其他编辑界面与此类似。添加关联线条用例的时序图如图 5-4 所示。

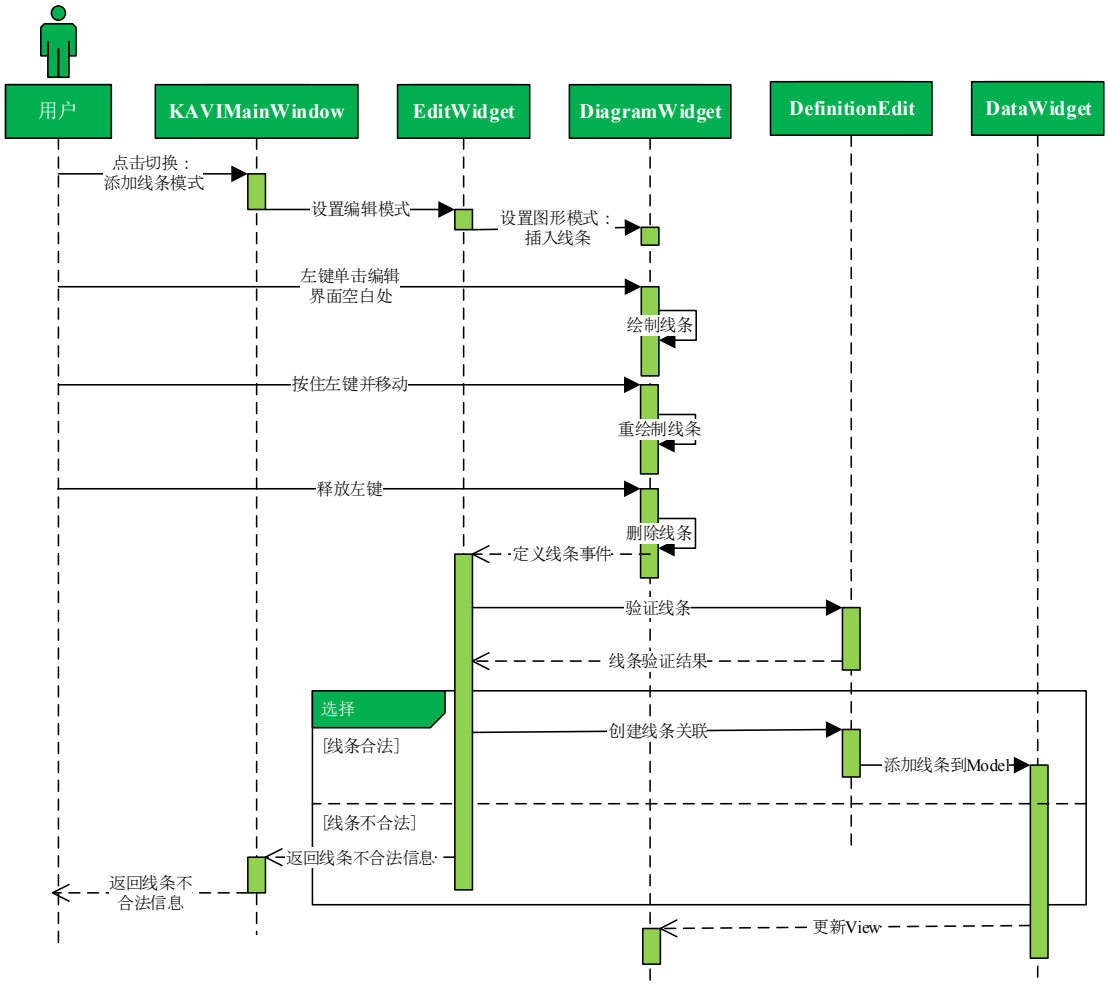

图 5-4 声明表达语言编辑界面的添加关联线条用例时序图

添加关联线条的实现描述如下：

1) 用户点击 KAVIMainWindow 工具栏的线条按钮，设置 EditWidget 的编辑模式为添加线条模式，并设置 DiagramWidget 的图形模式为插入线条。

2) 用户左键单击编辑界面的空白处，DiagramWidget 捕获鼠标左键事件，绘制临时线条。

3) 用户在编辑界面中按住左键并移动，DiagramWidget 捕获鼠标移动事件，重新绘制临时线条。

4) 用户在编辑界面中释放左键，DiagramWidget 捕获鼠标释放事件，删除临时线条，封装线条起点终点等信息到事件中并传播到 EditWidget。

5) EditWidget 分发处理定义线条事件，定义线条，调用 DefinitionEdit 验证线条的合法性。

6) DefinitionEdit 判断线条是否合法。如果判断线条合法，则在 DefinitionEdit 中创建线条关联，并添加线条到 DataWidget，最后 DataWidget 会同步更新到 DiagramWidget，从而更新编辑界面。如果判断线条不合法，则通过 KAVIMainWindow 反馈给用户线条不合法的信息。

5.2.3 执行规划用例

在规划界面中，在执行规划之前，用户需要配置领域定义 PDDL 文件、问题定义 PDDL 文件，在配置领域定义 PDDL 文件后，选择执行规划所需的规划器，然后执行规划，规划界面展示规划结果。执行规划用例的时序图如图 5-5 所示。

执行规划的实现描述如下：

1) 用户点击 KAVIMainWindow 工具栏的规划按钮，打开执行规划功能的界面 PlanningDialog。

2) 新建的 PlanningDialog 从规划器的 XML 配置文件中初始化规划器的集合，新建规划器建议类 PlannerSuggestion，初始化领域定义 PDDL 文件、

问题定义 PDDL 文件为通过可视化建模模块默认生成的 PDDL 文件，根据初始化的领域定义 PDDL 文件从 PlannerSuggestion 中获取合适的可用的规划器集合，并使用可用规划器集合初始化规划器下拉选择框。

3) 如果用户勾选自定义文件，则需要用户手动配置领域定义 PDDL 文件、问题定义 PDDL 文件为文件系统中已存在的 PDDL 文件，与 2)同理根据自定义的领域定义 PDDL 文件设置规划器下拉选择框。

4) 用户在规划器下拉选择框中选择规划器，点击执行规划，PlanningDialog 新建一个规划执行器 ExecPlanner，执行规划求解并生成、返回规划文件。

5) PlanningDialog 调用 PlanAnalyzer 获取规划文件的 HTML 报告，并渲染展示规划的 HTML 报告。

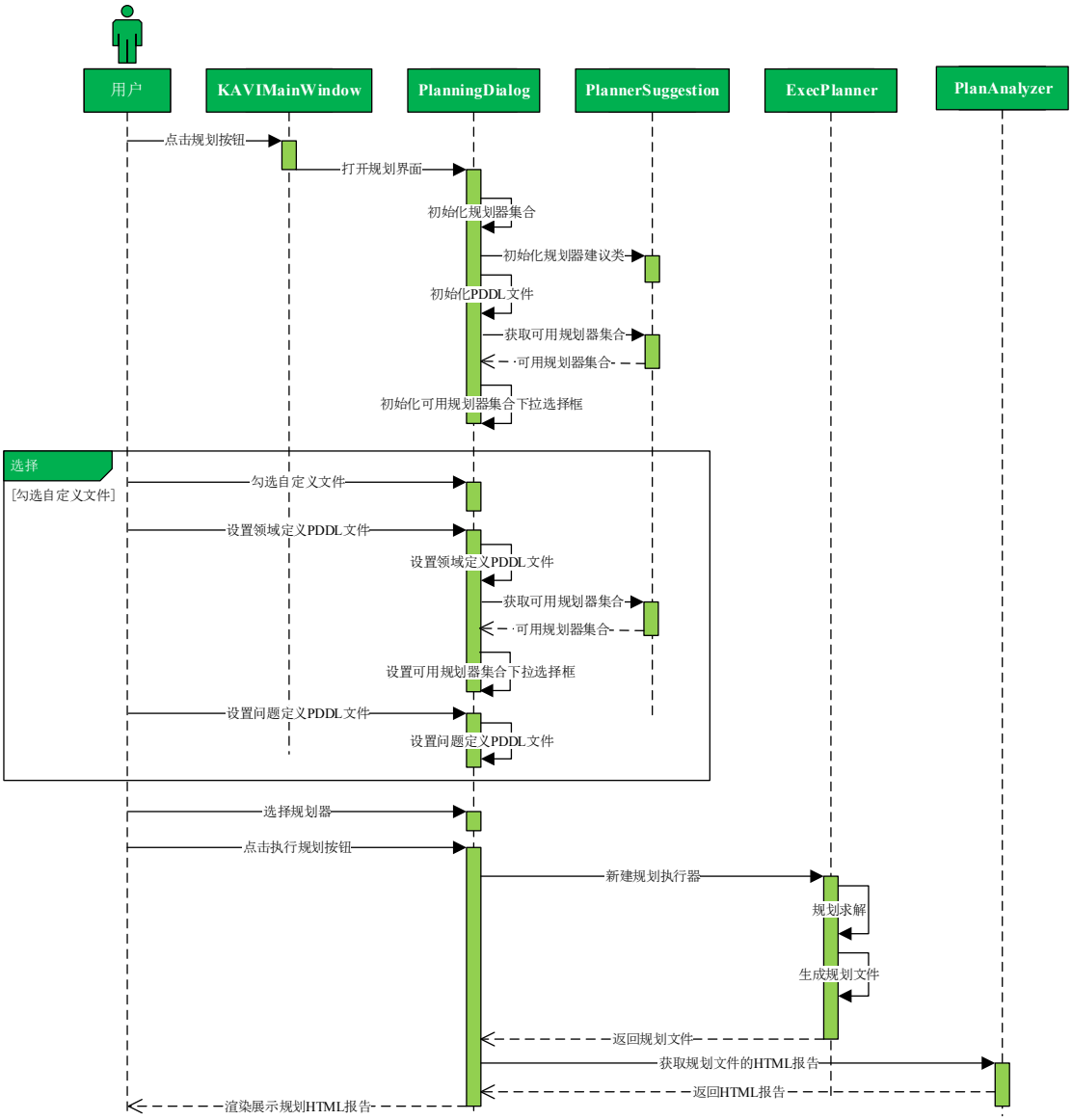

图 5-5 执行规划用例时序图

5.2.4 执行规划验证用例

在规划界面中，在执行规划验证之前，用户需要勾选规划验证，配置领域定义 PDDL 文件、问题定义 PDDL 文件、规划文件，然后执行规划验证，打开规划验证界面展示规划验证的详细报告。执行规划验证用例的时序图如图 5-6 所示。

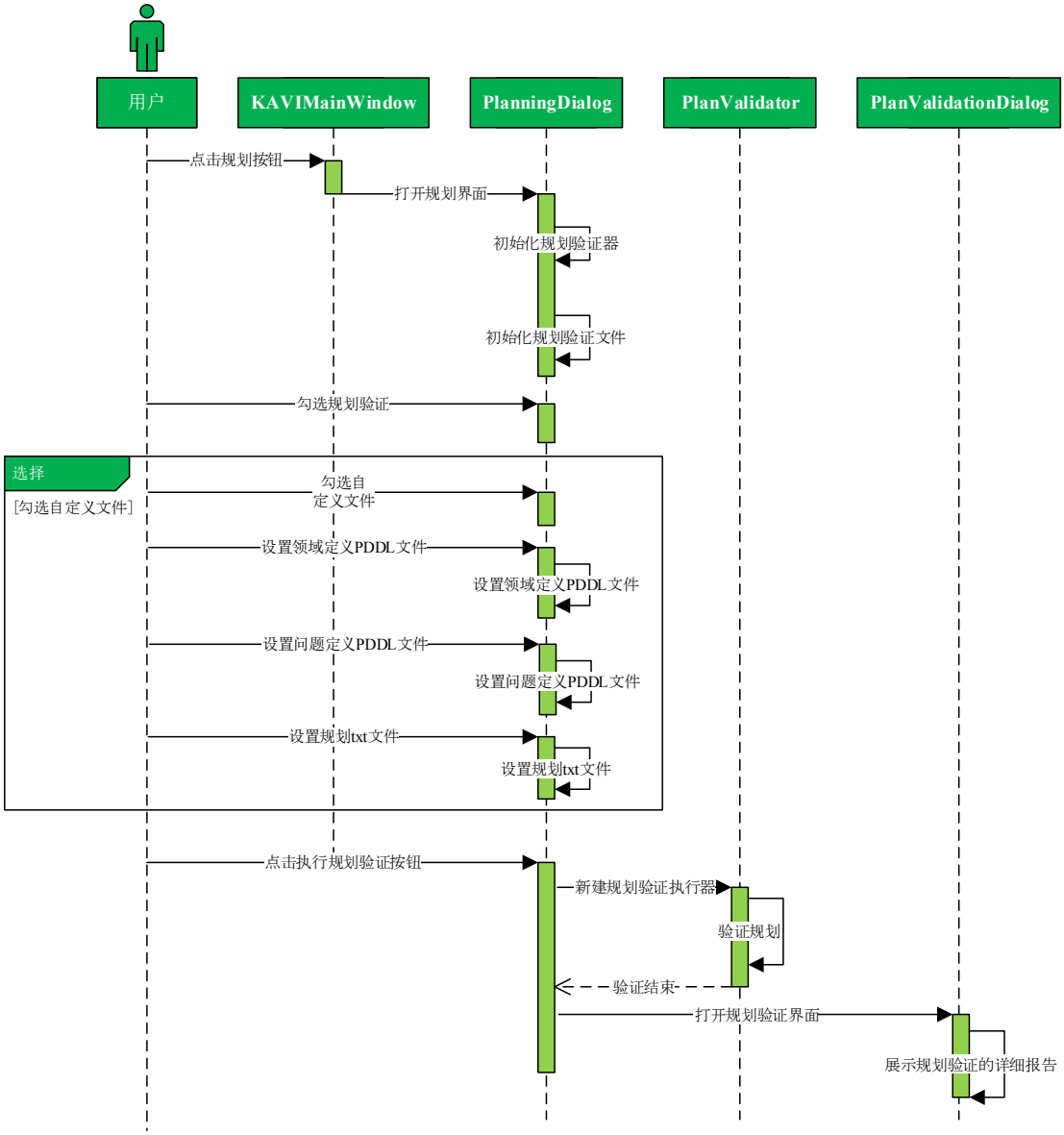

图 5-6 执行规划验证用例时序图

执行规划验证的实现描述如下：

1) 用户点击 KAVIMainWindow 工具栏的规划按钮，打开执行规划验证功能的界面 PlanningDialog。

2) 新建的 PlanningDialog 从规划验证器的 XML 配置文件中初始化规划验证器，初始化领域定义 PDDL 文件、问题定义 PDDL 文件为通过可视化建模模块默认生成的 PDDL 文件，初始化规划文件为通过规划模块默认生成的规划文件，用户勾选规划验证。

3) 如果用户勾选自定义文件，则需要用户手动配置领域定义PDDL文件、问题定义PDDL文件、规划文件为文件系统中已存在的PDDL文件。

4) 用户点击执行规划验证按钮，PlanningDialog 新建一个规划验证执行器PlanValidator，根据领域定义 PDDL 文件和问题定义 PDDL 文件验证规划文件中的规划。

5) PlanningDialog 打开规划验证界面 PlanValidationDialog，展示规划验证的详细报告，包括动作序列信息、动作详细信息、世界状态转移信息等等。

5.2.5 创建新动作用例

基于 5.2.4，执行规划验证之后，在规划验证界面，如果规划被验证为不合理的，用户可以根据修复建议修正领域模型。用户单击选中规划动作序列中用红色标记的有缺陷的动作，对于该动作中不满足的前提条件，用户选中修复建议窗口中的任意一条修复建议并执行修复，规划验证模块可以提供 2 个可选的修复选项，其中一个选项是自动地直接在规划领域中创建新动作。创建新动作用例的时序图如图 5-7 所示。

创建新动作的实现描述如下：

1) 在规划验证界面PlanValidationDialog中，用户单击选中动作序列中用红色标记的有缺陷的动作，PlanValidationDialog 展示选中动作详细信息、世界状态转移信息、选中动作的修复建议。

2) 用户单击选中 PlanValidationDialog 修复建议窗口中的任意一条修复建议，点击修复按钮，打开修复界面 RepairDialog。

3) 在修复界面 RepairDialog 中，用户在修复选项下拉选择框中选择创建新动作选项，点击确定，返回所选的创建新动作选项到PlanValidationDialog。

4) PlanValidationDialog 封装创建新动作选项的信息，返回封装的创建新动作选项信息到 PlanningDialog。

5) PlanningDialog 判断当前被验证的规划文件是否自定义文件。如果是自定义文件，则构建新动作的 PDDL 文本，并用领域定义 PDDL 文件和新动作 PDDL 文本打开文件文本编辑界面 EditFileDialog，对领域定义 PDDL 文件进行编辑，添加新动作 PDDL 文本到领域定义 PDDL 文件中。如果不是自定义文件，返回创建新动作选项到 KAVIMainWindow，系统跳转到可视化建模中的定义规划动作编辑界面，用户输入新动作名称，KAVIMainWindow 封装新动作的信息，并调用 OperatorsEdit 根据新动作信息创建新动作，OperatorsEdit 添加新动作到 DataWidget，最后 DataWidget 会同步更新到 DiagramWidget，从而更新编辑界面。

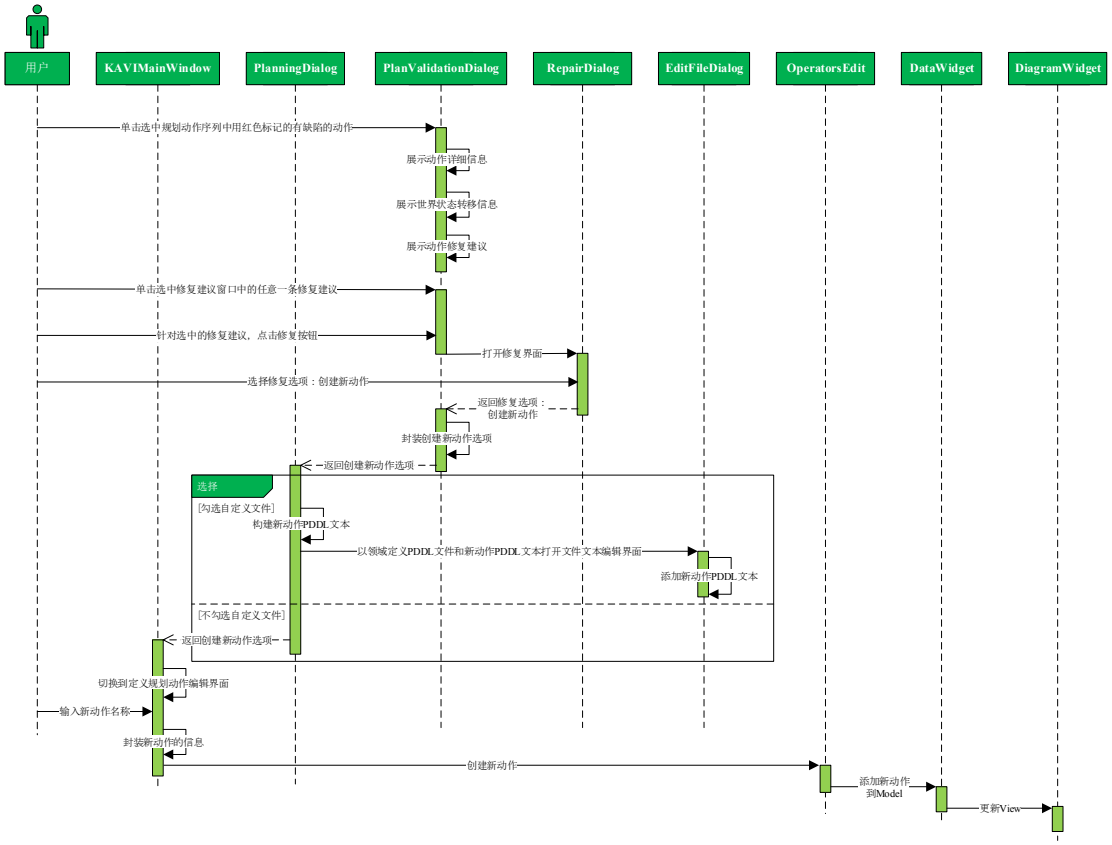

图 5-7 创建新动作用例时序图

5.2.6 修改已有动作用例

基于 5.2.4，执行规划验证之后，在规划验证界面，如果规划被验证为不合理的，用户可以根据修复建议修正领域模型。用户单击选中规划动作序列中用红色标记的有缺陷的动作，对于该动作中不满足的前提条件，用户选中修复建

议窗口中的任意一条修复建议并执行修复，规划验证模块可以提供 2 个可选的修复选项，另外一个选项是修改规划领域中已有的动作。修改已有动作用例的时序图如图 5-8 所示。

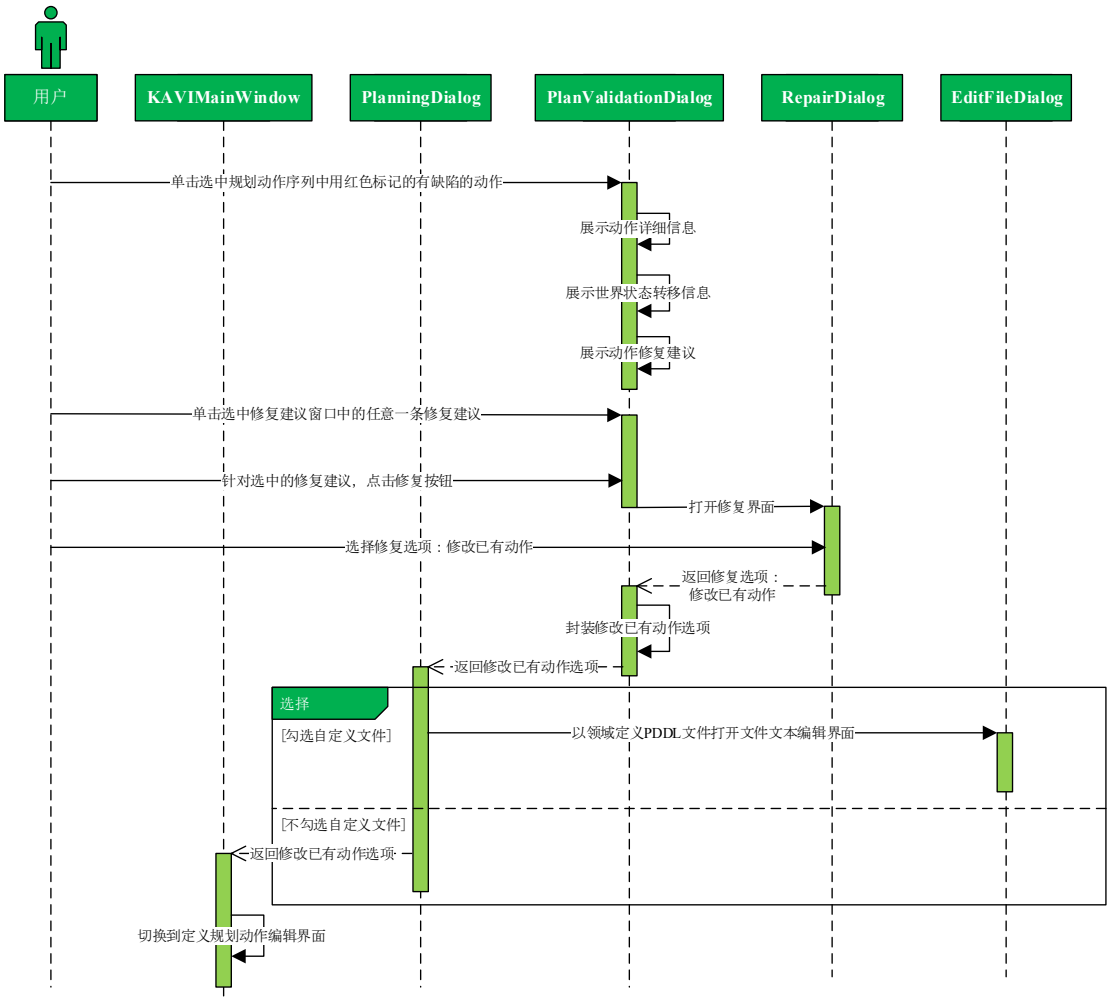

图 5-8 修改已有动作用例时序图

修改已有动作的实现描述如下：

1) 在规划验证界面 PlanValidationDialog 中，用户单击选中动作序列中用红色标记的有缺陷的动作，PlanValidationDialog 展示选中动作详细信息、世界状态转移信息、选中动作的修复建议。

2) 用户单击选中 PlanValidationDialog 修复建议窗口中的任意一条修复建议，点击修复按钮，打开修复界面 RepairDialog。

3) 在修复界面 RepairDialog 中，用户在修复选项下拉选择框中选择修改已

有动作选项，用户点击确定，RepairDialog 返回所选的修改已有动作选项到 PlanValidationDialog。

4) PlanValidationDialog 封装修改已有动作选项的信息，返回封装的修改已有动作选项信息到 PlanningDialog。

5) PlanningDialog 判断当前被验证的规划文件是否自定义文件。如果当前被验证的规划文件是自定义文件，则用领域定义 PDDL 文件打开文件文本编辑界面 EditFileDialog，由用户自主地对领域定义 PDDL 文件进行编辑。如果不是自定义文件，返回修改已有动作选项到 KAVIMainWindow，系统跳转到可视化建模中的定义规划动作编辑界面，由用户自主地修改规划领域模型中的已有动作。

第 6 章　系统详细设计

平台第 5 章从系统架构的角度描述人机混合智能规划平台的总体设计，本章描述平台的各个模块的详细设计，包括类图、函数功能、流程图、详细描述等。

6.1 领域知识库模块

领域知识库模块负责维护、管理领域知识。在领域知识库中包含两类基本领域知识模板，包括类模板和谓词模板，分别由不同 XML 配置文件维护。领域知识库模块的主要类图如图 6-1 所示。

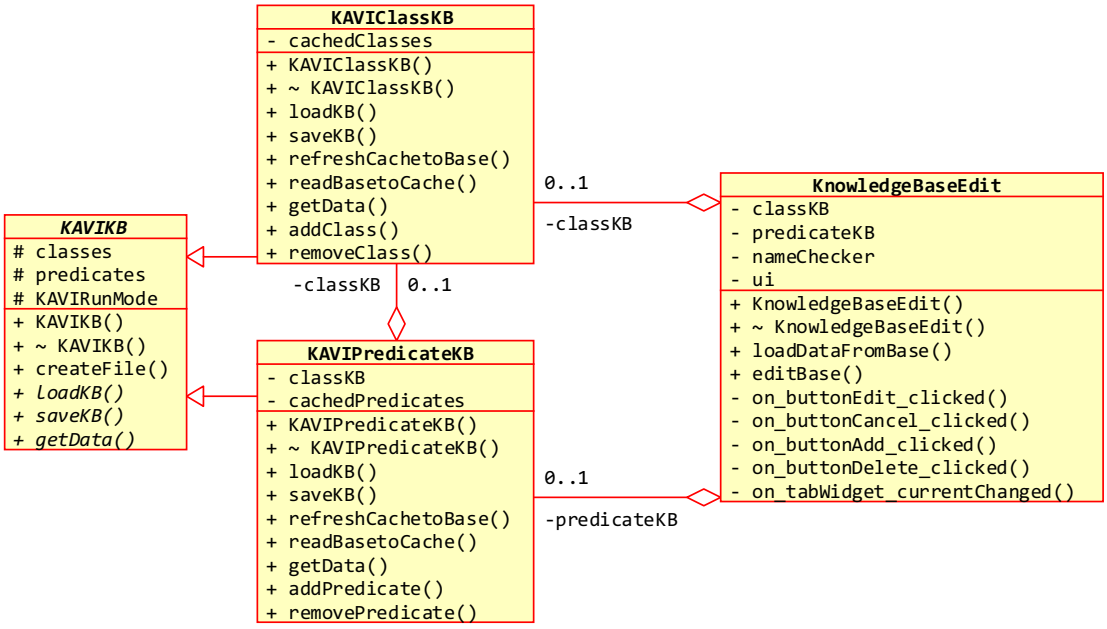

图 6-1 领域知识库模块类图

1) KAVIKB

KAVIKB 类是领域知识库，负责维护持久化的类模板和谓词模板，与相应的缓存类模板库和谓词模板库进行交互。

2) KAVIClassKB, KAVIPredicateKB

KAVIClassKB 类、KAVIPredicateKB 类分别是继承于 KAVIKB 的类模板库和谓词模板库，负责维护、管理缓存的类模板和谓词模板。

系统通过 loadKB()函数加载领域知识库时，首先从文件系统中加载领域知识库 XML 配置文件到 KAVIKB 中，然后通过函数 readBasetoCache()读取持久化的知识模板到 KAVIClassKB 缓存类模板库和 KAVIPredicateKB 缓存谓词模板库中。

系统运行过程中，可以对领域知识库进行 3 种操作，分别是添加知识模板(addClass(), addPredicate())、删除指定的领域知识模板(removeClass(), removePredicate())、获取知识模板(getData())。系统对领域知识库的所有更改会先刷写到缓存领域知识模板库 KAVIClassKB 和 KAVIPredicateKB 中。

当系统调用 saveKB()函数进行保存领域知识库时，首先通过函数 refreshCachetoBase()将缓存领域知识模板库 KAVIClassKB 和 KAVIPredicateKB 中的类模板和谓词模板刷写到 KAVIKB 中的持久化模板库中，然后刷新到文件系统中的领域知识库 XML 配置文件。

3) KnowledgeBaseEdit

KnowledgeBaseEdit 类负责在平台的开发者选项中对领域知识库进行管理和编辑，直接作用于其所依赖的 KAVIClassKB 和 KAVIPredicateKB。可以对类模板库和谓词模板库进行添加知识模板、删除知识模板、修改知识模板。

6.2 可视化建模模块

本节基于 2.4 的 VIZ 可视化建模、4.1.2 功能性需求中的规划领域可视化建模以及 5.1.2 可视化建模模块架构设计，描述平台的可视化建模模块的详细设计。

2.4 的 VIZ 可视化建模描述了平台的可视化建模的设计思想，通过这种可视化建模方法，可以用一种非常直观的方式，利用简单且显而易见的图形来建模 PDDL 领域模型

5.1.2 给出了可视化建模模块的架构设计，该模块整体上采用 MVC 设计模式，包含模型（Model）、 视图（View）、 控制器（Controller），如图 6-2 所示。

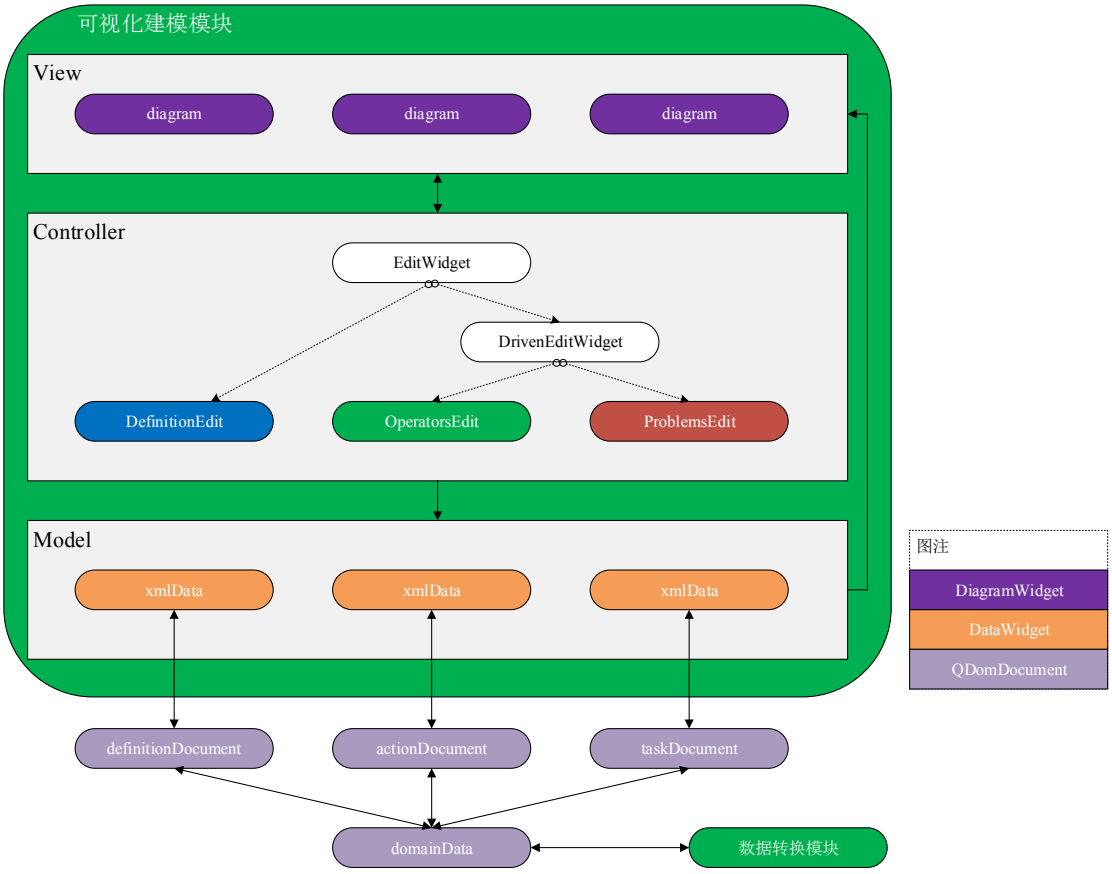

图 6-2 可视化建模模块架构图

1) EditWidget, DrivenEditWidget, DefinitionEdit, OperatorsEdit, ProblemsEdit

EditWidget 类继承于 QT 的图形视图（QGraphicsView），该类有 3 个特定实现的子类，分别是 DefinitionEdit、OperatorsEdit、ProblemsEdit，它们之间的类继承示意图如图 6-3 所示。

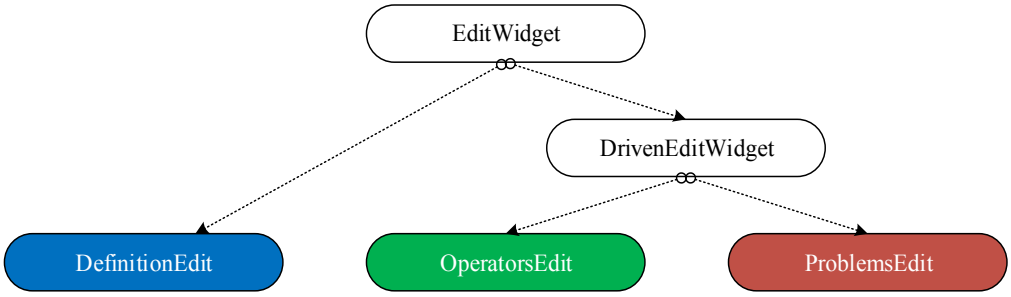

图 6-3 EditWidget 类继承示意图

DefinitionEdit、OperatorsEdit、ProblemsEdit 的作用如表 6-1 所示。

表 6-1 特定实现的 EditWidget 子类

特定实现的 EditWidget 子类	功能
DefinitionEdit	声明类（class）和谓词（predicate），负责处理声明表达语言编辑界面的特定业务逻辑
OperatorsEdit	使用变量（variable）和预先声明的谓词（predicate）定义规划的动作，负责处理定义规划动作编辑界面的特定业务逻辑
ProblemsEdit	使用对象（object）和预先声明的谓词（predicate）定义规划的问题，负责处理定义规划问题编辑界面的特定业务逻辑

EditWidget 类及其子类的类图如图 6-4 所示。

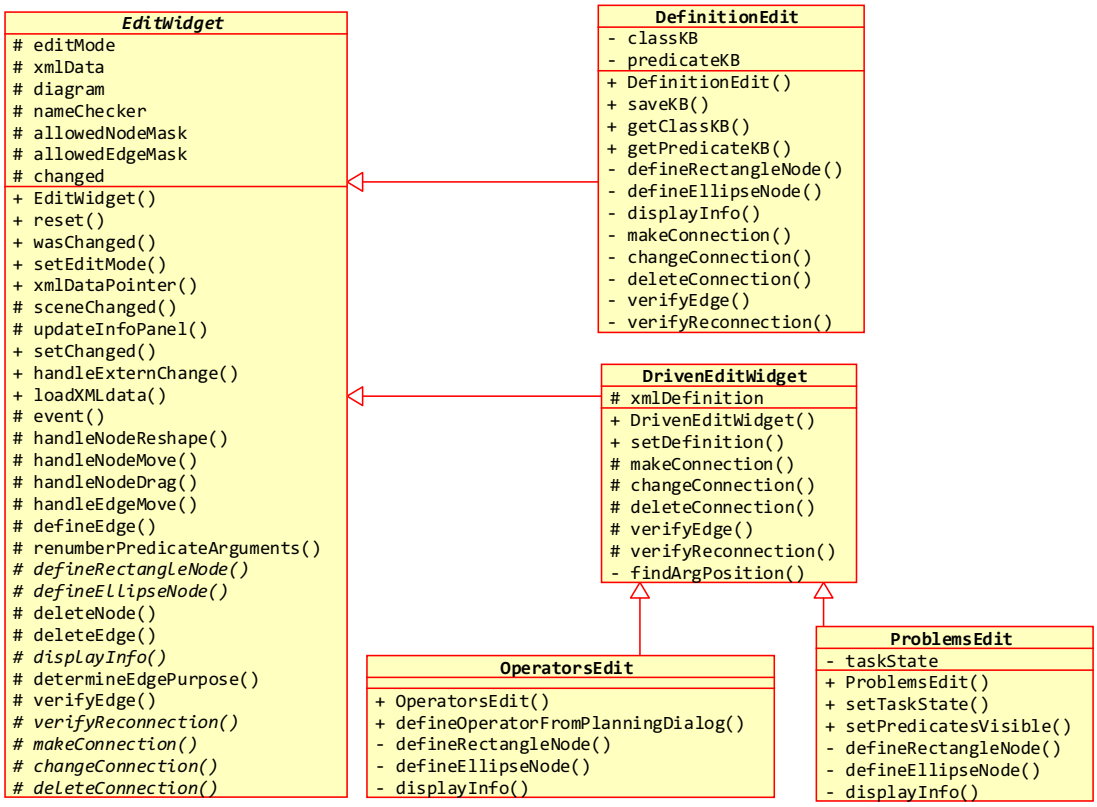

图 6-4 EditWidget 类及其子类类图

EditWidget 类是实现 MVC 模式控制器的基类，负责处理 3 个特定的编辑界面中的通用业务逻辑，在视图（DiagramWidget）和模型（DataWidget）之间充当桥梁角色，负责分发处理来自 DiagramWidget

的事件，并把来自 DiagramWidget 的图形更改信息同步到 DataWidget。EditWidget 类的主要函数功能如表 6-2 所示。

表 6-2 EditWidget 类的主要函数功能

函数	功能
setEditMode	设置 EditWidget 的编辑模式：添加谓词、添加类或变量或对象、添加关联线条、删除元素。设置 DiagramWidget 的图形模式
loadXMLdata	加载 XML 文档到 DataWidget
event	分发处理来自 DiagramWidget 的事件
handleNodeReshape	处理节点形状改变事件
handleNodeMove	处理节点移动事件
handleNodeDrag	处理节点拖拽事件
handleEdgeMove	处理线条移动事件
defineEdge	定义新线条
renumberPredicateArguments	重新计算谓词参数序号
deleteNode	删除节点以及相关联的线条
deleteEdge	删除线条
determineEdgePurpose	根据线条起点终点的节点类型：矩形（类）、椭圆（谓词），判断线条关联的属性：类的继承、谓词参数关联
verifyEdge	验证线条关联的属性，只有相应编辑界面所允许的线条关联属性才能通过验证

DefinitionEdit 类负责声明类（class）和谓词（predicate），负责处理声明表达语言编辑界面的特定业务逻辑。另外，DefinitionEdit 依赖于领域知识库 KAVIClassKB 和 KAVIPredicateKB，在声明表达语言编辑界面中声明类和谓词时，可以获取领域知识库的知识模板进行智能提示并自动补全，并自动添加新的知识模板到领域知识库。该类的主要函数功能如表 6-3 所示。

表 6-3 DefinitionEdit 类的主要函数功能

函数	功能
saveKB	更新领域知识库的缓存知识模板库，并保存领域知识库
defineRectangleNode	定义矩形节点（类）
defineEllipseNode	定义椭圆节点（谓词）
displayInfo	创建相应节点的属性信息界面
makeConnection	创建线条关联
changeConnection	改变线条关联
deleteConnection	删除线条关联
verifyEdge	验证线条，包括线条关联的属性、多重继承、循环继承、重载谓词
verifyReconnection	验证线条重新关联的属性：无意义取消重连、仅改变线条在原节点内坐标、线条两端节点改变

DrivenEditWidget 继承于 EditWidget，负责处理 OperatorsEdit 定义规划动作编辑界面和 ProblemsEdit 定义规划问题编辑界面的通用业务逻辑，OperatorsEdit 和 ProblemsEdit 依赖于 DefinitionEdit 声明的表达语言，与 DefinitionEdit 的 DataWidget 兼容。该类的主要函数功能如表 6-4 所示。

表 6-4 DrivenEditWidget 类的主要函数功能

函数	功能
setDefinition	设置 OperatorsEdit 和 ProblemsEdit 所依赖的 DefinitionEdit 的 DataWidget
makeConnection	创建线条关联
changeConnection	改变线条关联
deleteConnection	删除线条关联
verifyEdge	验证线条，包括线条关联的属性、多重继承、循环继承、重载谓词
verifyReconnection	验证线条重新关联的属性：无意义取消重连、仅改变线条在原节点内坐标、线条两端节点改变
findArgPosition	为谓词新添加的参数寻找可兼容的参数序号

OperatorsEdit 类使用变量（variable）和预先声明的谓词（predicate）定义规划的动作，负责处理定义规划动作编辑界面的特定业务逻辑。该类的主要函数功能如表 6-5 所示。

表 6-5 OperatorsEdit 类的主要函数功能

函数	功能
defineOperatorFromPlanningDialog	响应来自规划界面的规划验证的创建新动作修复选项
defineRectangleNode	基于 DefinitionEdit 中声明的类，定义矩形节点（变量）
defineEllipseNode	基于 DefinitionEdit 中声明的谓词，定义椭圆节点（谓词）
displayInfo	创建相应节点的属性信息界面

ProblemsEdit 类使用对象（object）和预先声明的谓词（predicate）定义规划的问题，负责处理定义规划问题编辑界面的特定业务逻辑。该类的主要函数功能如表 6-6 所示。

表 6-6 ProblemsEdit 类的主要函数功能

函数	功能
setPredicatesVisible	根据谓词的所在状态：初始状态、目标状态，设置谓词的可视性
defineRectangleNode	基于 DefinitionEdit 中声明的类，定义矩形节点（对象）
defineEllipseNode	基于 DefinitionEdit 中声明的谓词，定义椭圆节点（谓词）
displayInfo	创建相应节点的属性信息界面

2) DiagramWidget

如图 6-5 所示是 DiagramWidget 类的类图。DiagramWidget 类继承于 QT 的图形场景（QGraphicsScene），实现了 MVC 模式的视图（View）。负责维护、管理 3 个编辑界面的图形（diagram），图形的基本元素是节点（Node）和线条（Edge）的实例，均继承于 QT 的图形元素 QGraphicsItem 以及 QGraphicsLineItem。每一个图形元素的实例都有全

局唯一的编号，作为维护图形元素实例的容器的关键字（key），维护图形元素实例的容器如表 6-7 所示。

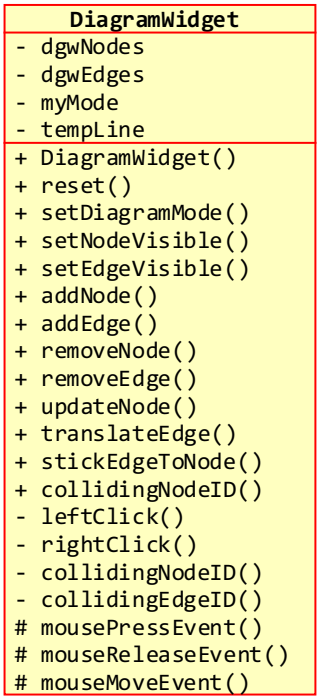

图 6-5 DiagramWidget 类图

表 6-7 DiagramWidget 维护图形元素实例的容器

容器的数据结构	功能
QHash<int, Node*> dgwNodes	维护节点图形元素实例
QHash<int, Edge*> dgwEdges	维护线条图形元素实例

DiagramWidget 类的主要函数功能如表 6-8 所示。

表 6-8 DiagramWidget 类的主要函数功能

函数	功能
reset	重置清空 DiagramWidget 的图形元素实例
setDiagramMode	设置 DiagramWidget 的图形模式：检测单击、插入线条、移动图形元素
addNode	添加节点并可视化节点
addEdge	添加线条并可视化线条
removeNode	移除节点

removeEdge	移除线条
updateNode	更新节点
translateEdge	根据给定矢量移动线条的端点
stickEdgeToNode	移动线条端点到节点内
leftClick	传播左键单击事件到 EditWidget
rightClick	传播右键单击事件到 EditWidget
mousePressEvent	根据图形模式，捕获鼠标按下事件
mouseReleaseEvent	根据图形模式，捕获鼠标释放事件，并传播定义线条事件到 EditWidget
mouseMoveEvent	根据图形模式，捕获鼠标移动事件

3) DataWidget

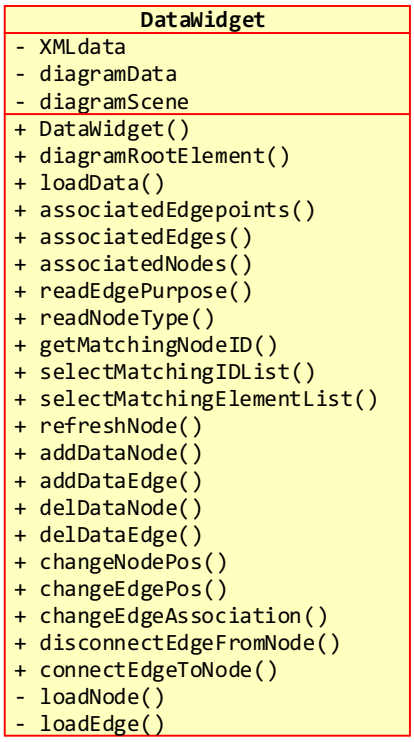

图 6-6 DataWidget 类图

图 6-6 是 DataWidget 类的类图。DataWidget 类实现了 MVC 模式的模型（Model）。负责维护、管理 3 个编辑界面的图形数据，图形数据由 QT 的 XML 文档类（QDomDocument）维护，如表 6-9 所示，所使用的 XML 标签描述详见附录 A1。

表 6-9 DataWidget 相关 XML 文档类

关联的编辑界面	QDomDocument
DefinitionEdit	definitionDocument
OperatorsEdit	actionDocument
ProblemsEdit	taskDocument

DataWidget 类的主要函数功能如表 6-10 所示。

表 6-10 DataWidget 类的主要函数功能

函数	功能
loadData	加载 XML 文档，并更新 DiagramWidget
refreshNode	刷新节点，并更新 DiagramWidget
addDataNode	添加节点，并更新 DiagramWidget
addDataEdge	添加线条，，并更新 DiagramWidget
delDataNode	删除节点，并更新 DiagramWidget
delDataEdge	删除线条，并更新 DiagramWidget
changeNodePos	改变节点坐标
changeEdgePos	改变线条坐标
changeEdgeAssociation	改变线条所关联的节点
connectEdgeToNode	关联线条到节点
disconnectEdgeFromNode	从节点移除关联线条

在可视化建模模块中，EditWidget 类、DataWidget 类和 DiagramWidget 类提供核心的业务处理功能，它们之间的交互逻辑如图 6-7 所示。

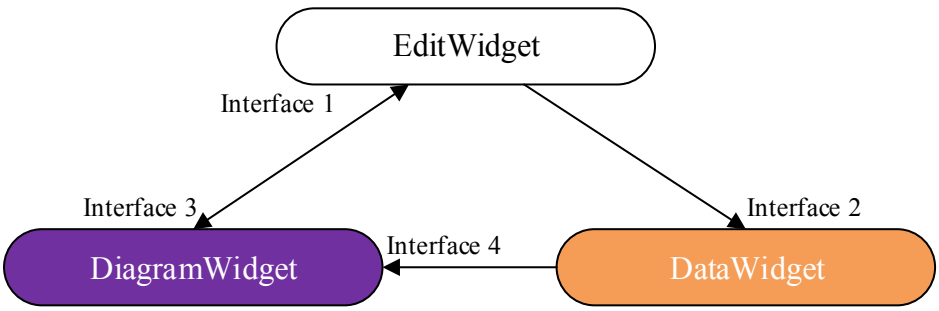

图 6-7 可视化建模模块交互逻辑图

图 6-7 中的箭头表示类之间的交互接口，下面分别介绍每一个接口。

1) 接口 1： DiagramWidget ⇨ EditWidget

这个接口负责处理将 DiagramWidget 中触发的鼠标事件进行封装并传播到 EditWidget，通过 QT 的基本事件机制实现。继承于 QT 的 QEvent 事件类，封装了自定义事件的 DiagramEvent 类，该类在 QEvent 基础上可以携带额外的信息，类图如图 6-8 所示，提供的额外信息如表 6-11 所示，主要函数功能如表 6-12 所示。

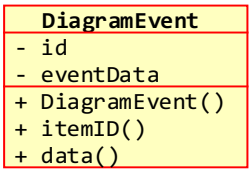

图 6-8 DiagramEvent 类图

表 6-11 DiagramEvent 事件类的额外信息

额外信息的数据结构	功能
int id	触发事件的图形元素（节点、线条）的编号
QVariant eventData	事件的特定数据

表 6-12 DiagramEvent 类的主要函数功能

函数	功能
itemID	获取触发事件的图形元素的编号
data	获取事件的特定数据

DiagramEvent 类所有支持的事件信息如表 6-13 所示。

表 6-13 DiagramEvent 类支持的事件

事件名称	事件描述	事件数据	事件触发所在类
NodeLeftClick	左键单击节点，删除节点	-	DiagramWidget
NodeRightClick	右键单击节点，请求展示节点信息	-	DiagramWidget
DiagramLeftClick	左键单击空白处，	单击的坐标	DiagramWidget

	定义节点		
DiagramRightClick	暂未使用	-	-
EdgeRightClick	暂未使用	-	-
EdgeLeftClick	左键单击线条，删除线条	-	DiagramWidget
NodeMoved	拖拽节点后释放鼠标左键，完成移动节点	移动的矢量	Node
NodeReshaped	节点形状改变（重命名节点）	-	Node
NodeDrag	移动鼠标拖拽节点，正在移动节点	移动的矢量	Node
EdgeDefined	拖拽临时线条后释放鼠标左键，定义线条	起点节点编号、终点节点编号、起点坐标、终点坐标	DiagramWidget
EdgeMoved	拖拽线条端点后释放鼠标左键，完成移动线条	端点的指向性、移动的矢量	Edge

2) 接口 2： EditWidget ⇨ DataWidget

在 EditWidget 接收并分发处理来自 DiagramWidget 的事件时，这个接口负责处理将图形元素的更改同步到 DataWidget 中，通过 EditWidget 调用 DataWidget 的函数实现，其中涉及到的函数如表 6-14 所示。另外通过接口 4，对于 DataWidget 的所有更改都被同步到相关联的 DiagramWidget 中。

表 6-14 EditWidget 同步 DataWidget 所用函数

函数	功能
addDataNode	添加节点
addDataEdge	添加线条
delDataNode	删除节点
delDataEdge	删除线条
changeNodePos	改变节点坐标
changeEdgePos	改变线条坐标

changeEdgeAssociation	改变线条所关联的节点
connectEdgeToNode	关联线条到节点
disconnectEdgeFromNode	从节点移除关联线条

3) 接口 3： EditWidget ⇨ DiagramWidget

在 EditWidget 接收并分发处理来自 DiagramWidget 的事件时，这个接口负责处理没有涉及到 DataWidget 的业务逻辑，通过 EditWidget 调用 DiagramWidget 的函数实现，其中涉及到的函数如表 6-15 所示。

表 6-15 EditWidget 操作 DiagramWidget 所用函数

函数	功能
setDiagramMode	设置图形模式：检测单击、插入线条、移动图形元素
setNodeVisible	设置节点可视性
setEdgeVisible	设置线条可视性
translateEdge	根据给定矢量移动线条的端点
stickEdgeToNode	移动线条的端点到节点内
collidingNodeID	获取与线条指定端点相接触的节点

4) 接口 4： DataWidget ⇨ DiagramWidget

这个接口负责处理将涉及到 DataWidget 的所有更改同步到相关联的 DiagramWidget 中，通过 DataWidget 调用 DiagramWidget 的函数实现，其中涉及到的函数如表 6-16 所示。

表 6-16 DataWidget 操作 DiagramWidget 所用函数

函数	功能
addNode	添加节点并可视化节点
addEdge	添加线条并可视化线条
removeNode	移除节点
removeEdge	移除线条
updateNode	更新节点

6.3 语法约束检查模块

语法约束检查模块负责对通过可视化建模生成的领域模型进行约束检查，本节基于 3.1 领域知识一致性检查中的领域约束、4.1.2 功能性需求中的约束检查，描述平台的语法约束检查模块的详细设计，该模块的类图如图 6-9 所示。

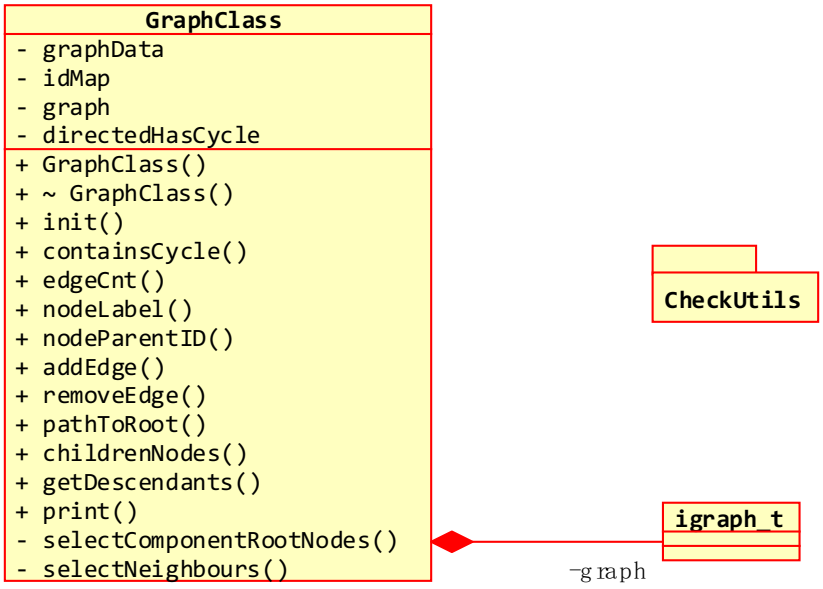

图 6-9 语法约束检查模块类图

1) igraph_t

平台的可视化建模方法采用的图形是包含节点和线条的图，igraph_t 类用于维护、管理和分析这些图结构，包括检测环路、连通量、节点的邻居等等。该类用于表示图结构的数据结构如表 6-17 所示，主要函数功能如表 6-18 所示。

表 6-17 igraph_t 类表示图结构的数据结构

图结构的数据结构	功能
bool directed	无向图或者有向图
QSet<int> vertices	图结构的节点编号的集合
QMultiMap<int, int> edges;	图结构的边的集合

表 6-18 igraph_t 类的主要函数功能

函数	功能
init	使用指定的节点数目和图的方向性初始化图
addEdge	使用给定起点、终点添加边
addEdges	添加边的集合
removeEdge	使用给定起点、终点删除边
degree	计算给定节点的度
neighborhood	获取给定节点临近区域里的所有节点
subcomponent	获取给定节点所在连通量的所有节点
neighbors	获取给定节点的邻接节点

2) GraphClass

基于 igraph_t 类，GraphClass 提供对图结构的通用操作，负责将可视化建模中的图形（diagram）的 XML 数据映射为图结构 igraph_t，并对其进行维护管理。该类的主要数据结构如表 6-19 所示，主要函数功能如表 6-20 所示。

表 6-19 GraphClass 类的主要数据结构

数据结构	功能
QDomElement graphData	GraphClass 类所依赖的图形 XML 数据
QMap<int, int> idMap	图形 XML 数据的节点编号与图结构的节点编号的映射，GraphClass 类的函数操作图形 XML 数据的节点编号
igraph_t graph	GraphClass 类的图结构实例

表 6-20 GraphClass 类的主要函数功能

函数	功能
GraphClass	使用给定的图形 XML 数据构造 GraphClass
Init	使用给定图形 XML 数据的节点类型、边类型、图的方向性初始化 GraphClass
containsCycle	判断图结构是否包含环路
edgeCnt	计算指定节点的给定类型的关联边的数目
nodeParentID	获取给定节点的父节点编号

addEdge	根据给定起点、终点添加边
removeEdge	根据给定起点、终点删除边
pathToRoot	获取给定节点到其根节点的路径的所有节点
childrenNodes	获取给定节点的所有子节点或者所有连通量的根节点
getDescendants	获取给定节点的所有后代节点
selectComponentRootNodes	获取所有连通量的根节点
selectNeighbours	获取给定节点的所有子节点

3) CheckUtils

基于 GraphClass 类，CheckUtils 命名空间定义了用于对领域模型进行约束检查的通用函数，如表 6-21 所示。

表 6-21 CheckUtils 命名空间的主要函数功能

函数	功能
predicateArguments	获取谓词的参数列表
checkPredicateDefinition	检查所有重载谓词的参数个数是否相同
checkPredicateInstance	检查给定的谓词实例是否兼容谓词的声明，包括参数类型、参数个数、参数次序
checkClassInstance	检查给定的类实例（变量、对象）是否兼容类的声明
checkDefinition	检查表达语言的类声明和谓词声明，重载谓词的参数个数是否相同
checkDependentDiagram	检查给定图形（动作或者问题）与表达语言的兼容性，包括所有类实例和谓词实例
brokenDiagrams	使用 checkDependentDiagram 检查动作集合里的每个动作或者问题集合里的每个问题的图形与表达语言的兼容性，并返回正确兼容和不正确兼容的动作或者问题集合
selectPredicateDefinition	从表达语言中筛选出与给定谓词实例匹配度最高的谓词声明，包括参数类型和参数次序
classFamily	获取给定类到祖先类的路径里的所有家族类

6.4 数据转换模块

数据转换模块负责语言和相关输入输出文件的转换，相关类图如图 6-10 所示。

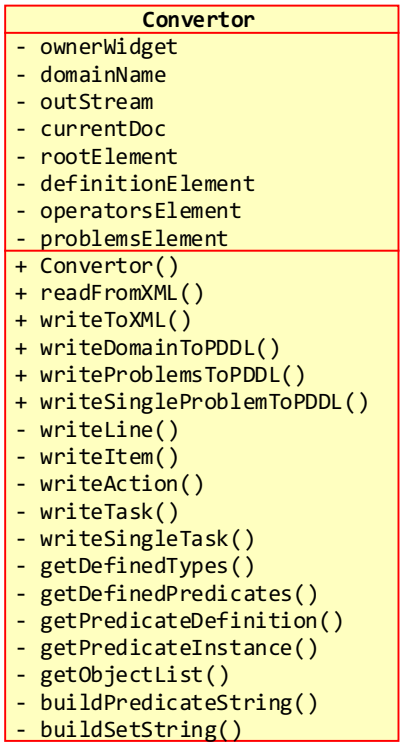

图 6-10 Convertor 类图

Convertor 类负责处理由 DataWidget 维护管理的编辑界面的 XML 图形数据的本地文件导入和导出，以及将领域模型的 XML 图形数据转换为 PDDL 语言格式文件，该类的主要函数功能如表 6-22 所示。

表 6-22 Convertor 类的主要函数功能

函数	功能
readFromXML	从本地 XML 文件导入 QDomDocument 数据
writeToXML	写 QDomDocument 数据到本地 XML 文件
writeDomainToPDDL	写表达语言以及规划动作的 QDomDocument 数据到本地 PDDL 文件
writeProblemsToPDDL	写所有规划问题的 QDomDocument 数据到本地 PDDL 文件

writeSingleProblemToPDDL	写指定的规划问题的 QDomDocument 数据到本地 PDDL 文件

6.5 规划模块

规划模块负责执行规划并展示规划结果，相关类图如图 6-11 所示。

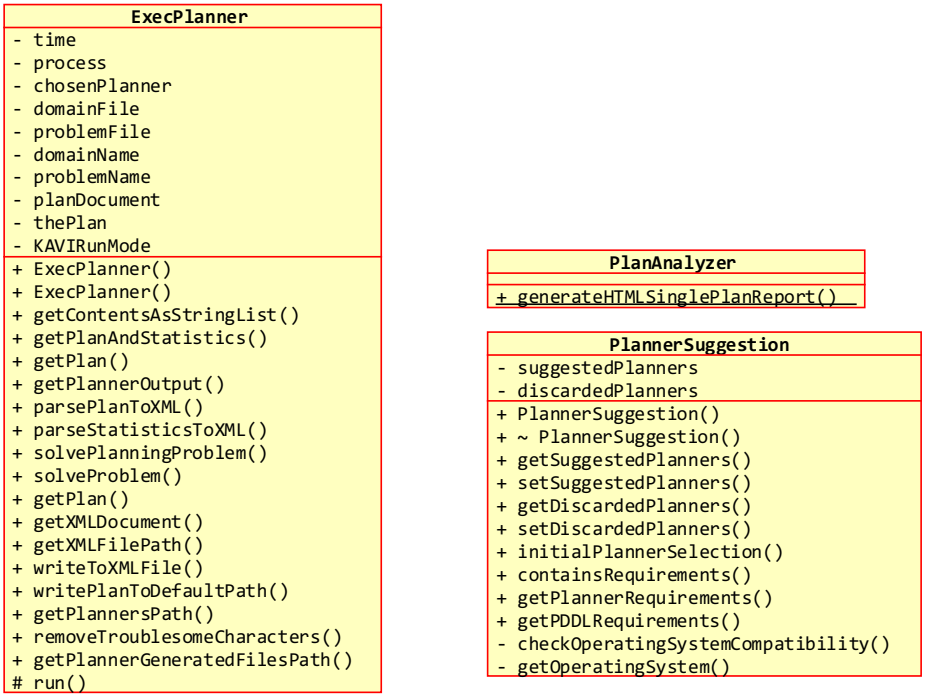

图 6-11 规划模块相关类图

平台集成了外部规划系统，包括多种规划器（Metric-FF、Blackbox、LPG-td），规划器的设置支持 XML 配置化。规划模块利用领域定义 PDDL 文件、问题定义 PDDL 文件、所选择的规划器进行规划求解，通过把 PDDL 文件作为规划器可执行程序的输入，获取规划器的控制台输出，对规划器的输出进行解析重组，得到规划的结果，规划结果支持 XML 配置化，可以把规划的 XML 数据转换为 HTML 格式进行展示，以及生成规划的文平台件。规划模块执行规划的过程示意图如图 6-12 所示。

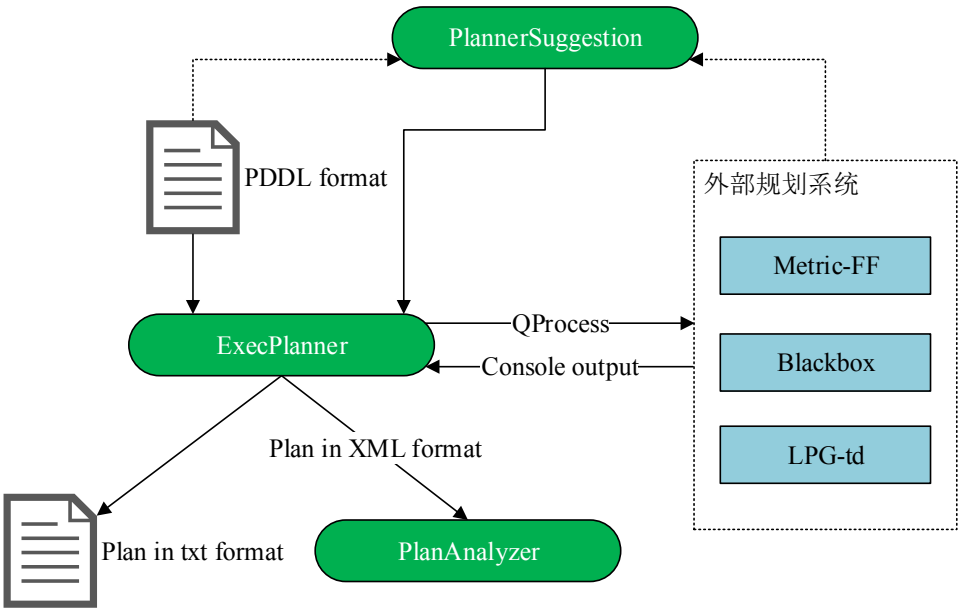

图 6-12 规划模块执行规划过程示意图

1） PlannerSuggestion

PlannerSuggestion 类负责为执行规划所需的领域定义 PDDL 文件从规划器集合中筛选出兼容的规划器，该类的主要函数功能如表 6-23 所示。

表 6-23 PlannerSuggestion 类的主要函数功能

函数	功能
initialPlannerSelection	给定领域定义 PDDL 文件和规划器集合，根据 PDDL 语言特性的兼容性、操作系统的兼容性初始化 PlannerSuggestion
getSuggestedPlanners	初始化 PlannerSuggestion 后，获取与给定领域定义 PDDL 文件兼容的规划器集合

2） ExecPlanner

因为执行规划需要调用外部可执行程序（规划器），为了让系统在执行规划过程中避免 UI 长时间处于停滞状态，执行规划需要使用多线程。ExecPlanner 类继承于 QT 的线程类（QThread），作为多线程执行规划的规划执行器实体，负责处理执行规划的各种业务逻辑。在规划执行器实体 ExecPlanner 中，PDDL 文件作为规划器可执行程序的输入，通过 QT 的进程类（QProcess）调用规划器，得到规划器的控制台输出，ExecPlanner 对规划器的输出进行解析重组，生成规划结果 XML 数据和

文平台件。该类的主要函数功能如表 6-24 所示。

表 6-24 ExecPlanner 类的主要函数功能

函数	功能
getPlanAndStatistics	将规划器的输出分组，分为规划动作序列部分和规划统计信息部分
getPlannerOutput	使用 PDDL 文件作为规划器输入，调用规划器进行规划求解，获取规划器输出
parseStatisticsToXML	将规划统计信息解析为 XML
parsePlanToXML	将规划动作序列解析为 XML
solvePlanningProblem	使用 PDDL 文件和规划器进行规划问题求解

3） PlanAnalyzer

PlanAnalyzer 类负责将规划的 XML 数据转换为 HTML 格式。

6.6 规划验证模块

规划验证模块负责执行规划验证并展示详细的验证信息，该模块的相关类图如图 6-13 所示，执行规划验证的过程示意图如图 6-14 所示。

平台集成了外部规划验证系统（VAL），规划验证器的设置支持 XML 配置化。规划验证模块利用领域定义 PDDL 文件、问题定义 PDDL 文件、规划文件、规划验证器进行规划验证，通过把 PDDL 文件、规划文件作为规划验证器的输入，获取规划验证器的控制台输出，对规划验证器的输出进行解析重组，得到规划的验证结果。其中，如果规划被验证为不合理的，可以从规划验证器的输出解析得到规划的修复建议。

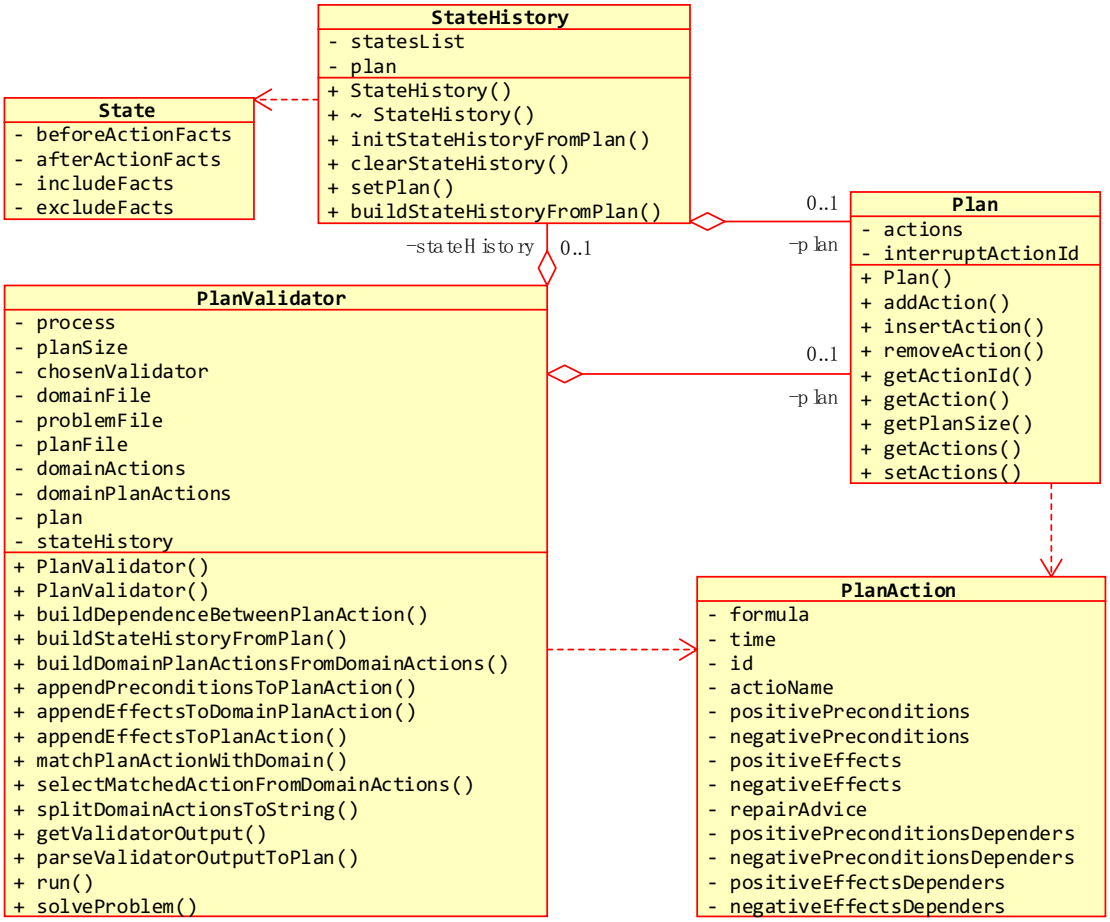

图 6-13 规划验证模块相关类图

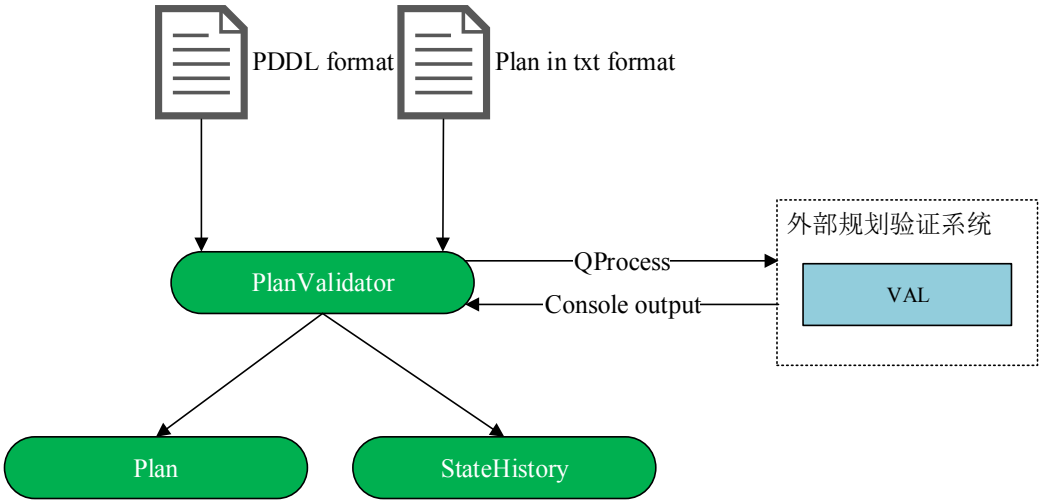

图 6-14 规划验证模块执行规划验证过程示意图

基于规划验证器的规划验证结果，为了获取到规划的动作序列的详细信息、世界状态的转移信息，规划验证模块对领域定义 PDDL 文件、问题定义 PDDL

文件进行解析，结合规划文件里的规划动作序列，封装了 4 个维护相关信息的类，分别是规划动作类（PlanAction）、规划类（Plan）、世界状态类（State）、世界状态历史类（StateHistory）。

规划动作类（PlanAction）封装了规划动作信息，其中如果规划动作是有缺陷的，则还包含了针对相应规划动作的修复建议，如表 6-25 所示。

表 6-25 PlanAction 数据结构及其描述

数据结构	描述
int id	规划动作的序号
QString formula	规划动作的原子式
double time	规划动作的时间
QSet<QString> positivePreconditions	规划动作的正前置条件
QSet<QString> negativePreconditions	规划动作的负前置条件
QSet<QString> positiveEffects	规划动作的正后置效果
QSet<QString> negativeEffects	规划动作的负后置效果
QMap<int, QSet<QString> > positivePreconditionsDependers	规划动作正前置条件的依赖动作
QMap<int, QSet<QString> > negativePreconditionsDependers	规划动作负前置条件的依赖动作
QMap<int, QSet<QString> > positiveEffectsDependers	规划动作正后置效果的被依赖动作
QMap<int, QSet<QString> > negativeEffectsDependers	规划动作负后置效果的被依赖动作
QMap<QString, bool> repairAdvice	规划动作的修复建议，QString 为世界状态的原子式，bool 为该原子式的建议值

规划类（Plan）封装了规划的动作序列，依赖于规划动作（PlanAction），如表 6-26 所示。

表 6-26 Plan 数据结构及其描述

数据结构	描述
QMap<int, PlanAction> actions	规划动作的序列

int interruptActionId	有缺陷的规划动作的序号

世界状态类（State）封装了世界的状态信息，与规划动作（PlanAction）相对应，如表 6-27 所示。

表 6-27 State 数据结构及其描述

数据结构	描述
QSet<QString> beforeActionFacts	执行相应的规划动作前的世界状态
QSet<QString> afterActionFacts	执行相应的规划动作后的世界状态
QSet<QString> includeFacts	执行相应的规划动作所增加的状态
QSet<QString> excludeFacts	执行相应的规划动作所删除的状态

世界状态历史类（StateHistory）封装了世界状态的转移信息，依赖于世界状态（State），基于规划类（Plan）里的动作序列构建而来，如表 6-28 所示。

表 6-28 StateHistory 数据结构及其描述

数据结构	描述
QList<State> statesList	世界状态的转移序列
Plan* plan	构建世界状态历史所基于的规划（Plan）

因为执行规划验证需要调用外部可执行程序（规划验证器），为了让系统在执行规划验证过程中避免 UI 长时间处于停滞状态，执行规划验证需要使用多线程。PlanValidator 类继承于 QT 的线程类（QThread），作为多线程执行规划验证的规划验证执行器实体，负责处理执行规划验证的各种业务逻辑。在规划验证执行器实体 PlanValidator 中，PDDL 文件、规划文件作为规划验证器可执行程序的输入，通过 QT 的进程类（QProcess）调用规划验证器，得到规划验证器的控制台输出，PlanValidator 对规划器的输出进行解析重组，获取规划验证结果。最后结合规划验证结果，通过对领域定义 PDDL 文件、问题定义 PDDL 文件进行解析，结合规划文件里的动作序列，封装得到 Plan 和 StateHistory。PlanValidator 类的主要函数功能如表 6-29 所示。

表 6-29 PlanValidator 类的主要函数功能

函数	功能

getValidatorOutput	使用 PDDL 文件、规划文件和规划验证器，调用规划验证器进行规划验证，并对规划验证器的验证输出结果进行解析重组，封装 Plan、StateHistory
parseValidatorOutputToPlan	解析规划验证器的验证输出，封装 Plan
buildDependenceBetweenPlanAction	构建 Plan 的动作序列之间的依赖关系
buildStateHistoryFromPlan	根据 Plan 构建 StateHistory

6.7 领域模型微调模块

基于 3.2 中 HILP 智能规划的设计思想、6.6 规划验证模块，平台的领域模型微调模块可以利用规划验证模块产生的 Plan 和 StateHistory，针对存在缺陷的规划动作，根据该规划动作的修复建议，提供对领域模型进行微调的修复选项，包括创建新动作、修改已有动作，并自动地跳转到相应的界面，交互式地执行修复选项。领域模型微调模块执行模型微调的过程示意图如图 6-15 所示。

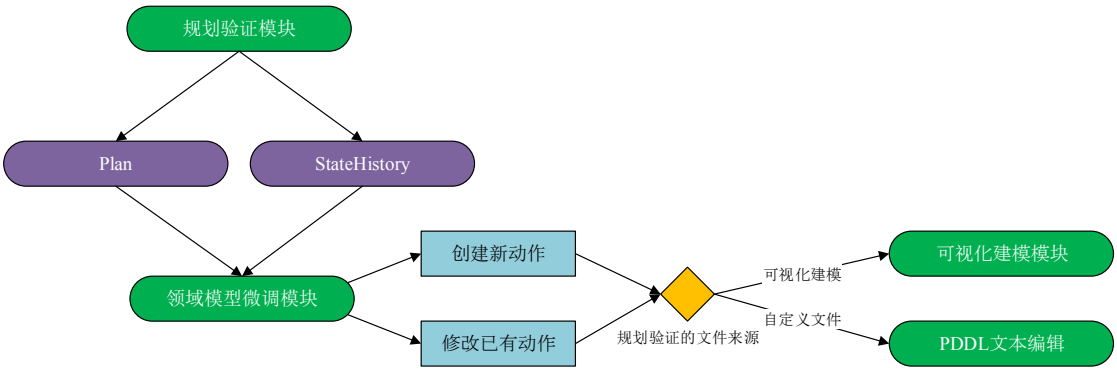

图 6-15 领域模型微调模块执行模型微调过程示意图

在规划验证过程中，如果规划被验证为不合理的，那么规划验证模块产生的 Plan 中的 interruptActionId 则标识着规划动作序列中存在缺陷的动作。领域模型微调模块根据缺陷动作中的修复建议，提供修复选项。

规划验证模块提供了 2 个可选的修复选项。一个是自动地直接在规划领域中创建一个新的动作，该动作没有任何的前提条件，只有唯一的一个后置效果，那就是修复建议中提示的有缺陷的动作所不满足的前提条件；另一个是允许用户自主的修改规划领域定义。另外，如果输入规划验证的领域定义 PDDL 文件是默认的由可视化建模产生的，那么用户的修复选项会作用于可视化建模的定

义规划动作编辑界面；如果输入规划验证的领域定义 PDDL 文件是自定义的已存在的，那么用户的修复选项会直接作用于领域定义 PDDL 文件，以文本编辑的方式。

第 7 章　系统实现

本章主要介绍人机混合智能规划平台的开发与运行环境，然后通过一个完备而具体的例子对平台的各个功能模块的运行情况进行展示，并对系统进行详细的测试。

7.1 系统开发及运行环境

平台的所有开发工作是在 Windows 系统上完成的，具体的开发环境信息如表 7-1 所示。平台运行在 Windows 系统中，具体的运行环境信息如表 7-2 所示。

表 7-1 系统开发环境

软件环境	
操作系统	Microsoft Windows [版本 10.0.10240]
预装环境	Qt 5.10.1 (MSVC 2015, 32 bit)
开发工具	Qt Creator 4.5.1
开发语言	C++
硬件环境	
处理器	Intel Core i5-3570K @ 3.40GHz 四核
系统内存	8GB

表 7-2 系统运行环境

软件环境	
操作系统	Microsoft Windows [版本不高于 10.0.10240]

7.2 系统的应用—以领域 logistics 为例

本节通过一个完备而具体的例子对平台各个功能模块的运行情况进行展示。

7.2.1 规划领域样例

本节使用的样例基于 2000 年第 2 届国际智能规划竞赛 IPC 中使用的测试数

据，包括领域和问题描述，其具体的数据信息如表 7-3 所示。

表 7-3 规划领域测试数据

domain	format	typing
logistics	STRIPS	typed
领域描述		
运输包裹，在城市内通过卡车运输，在城市之间通过飞机运输。在一个城市内的任意两个地点都是直接相连的（卡车可以在任意两个地点之间移动），城市之间的移动同理。每一个城市有一辆卡车，每一个城市有一个地点是机场。		

具体使用的数据是 logistics 领域的领域定义 PDDL 文件以及其中一个问题定义 PDDL 文件，详见附录 A2。

7.2.2 领域知识库

领域知识库主要提供 2 两个功能：

1) 为可视化建模中的声明表达语言提供智能提示和自动补全

   此功能在 7.2.3 可视化建模中展示。

2) 在开发者选项中编辑领域知识库

   点击主界面菜单栏中开发者选项的“领域知识库”按钮，打开领域知识库编辑界面，可以对领域知识模板进行添加删除修改操作，如图 7-1。

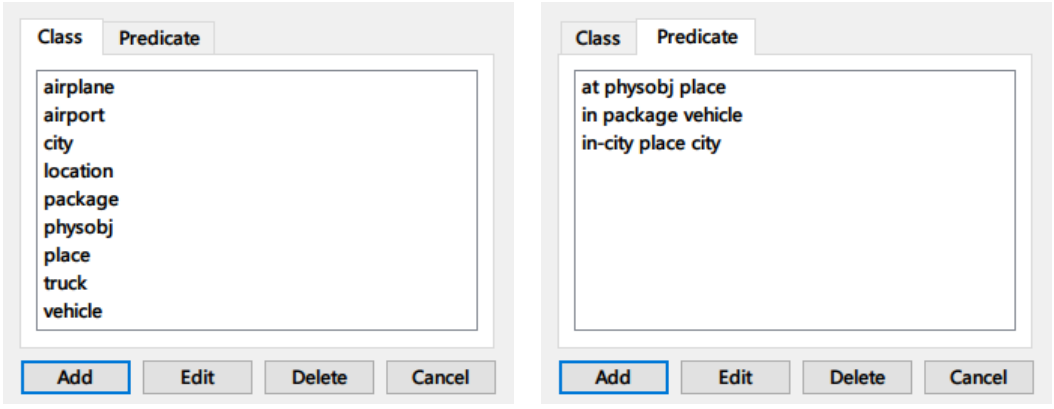

图 7-1 领域知识库编辑界面

7.2.3 可视化建模

可视化建模负责声明表达语言、定义规划动作、定义规划问题，分别对应3

个编辑界面，每个界面可提供的操作都大同小异，主要是添加谓词、添加类或变量或对象、添加关联线条、删除元素。对 logistics 领域进行可视化建模，下面以声明表达语言界面为例：

1) 添加谓词

在可视化建模编辑界面中，点击工具栏的蓝色椭圆按钮切换到添加谓词编辑模式，左键单击编辑界面空白处，打开声明谓词对话框，输入谓词（可包括谓词参数类型，以空格隔开），点击确定按钮，将在编辑界面中生成所输入谓词的对应图形元素，如图 7-2 所示。

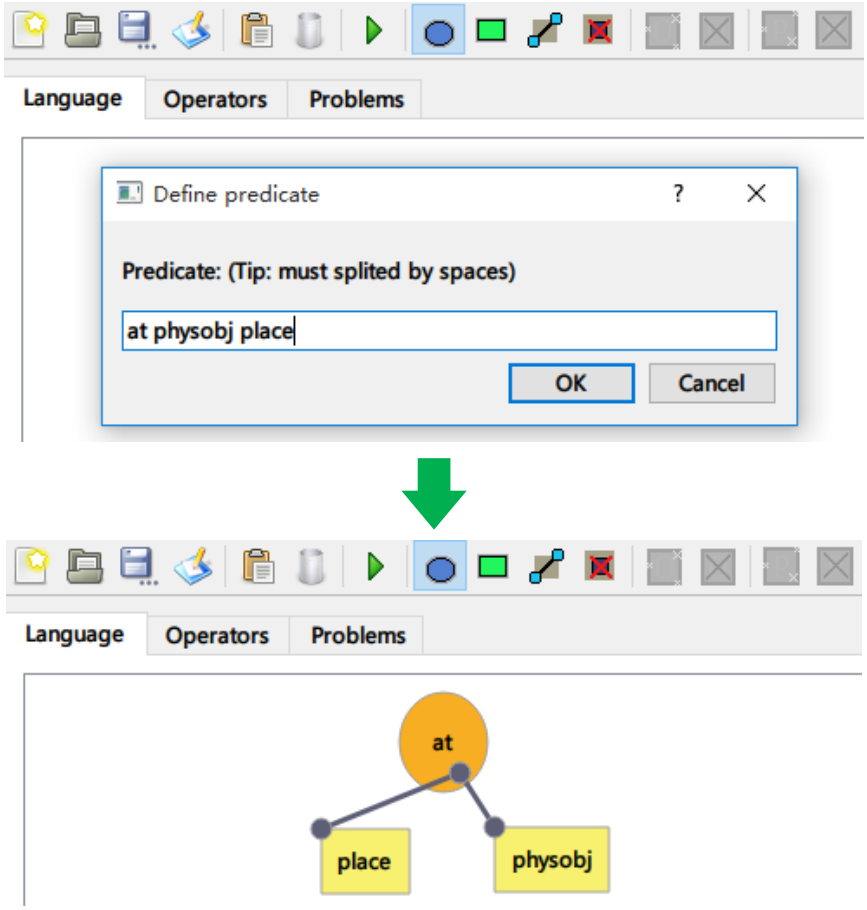

图 7-2 声明表达语言编辑界面中添加谓词

如图 7-3 所示，可利用领域知识库进行智能提示并自动补全。

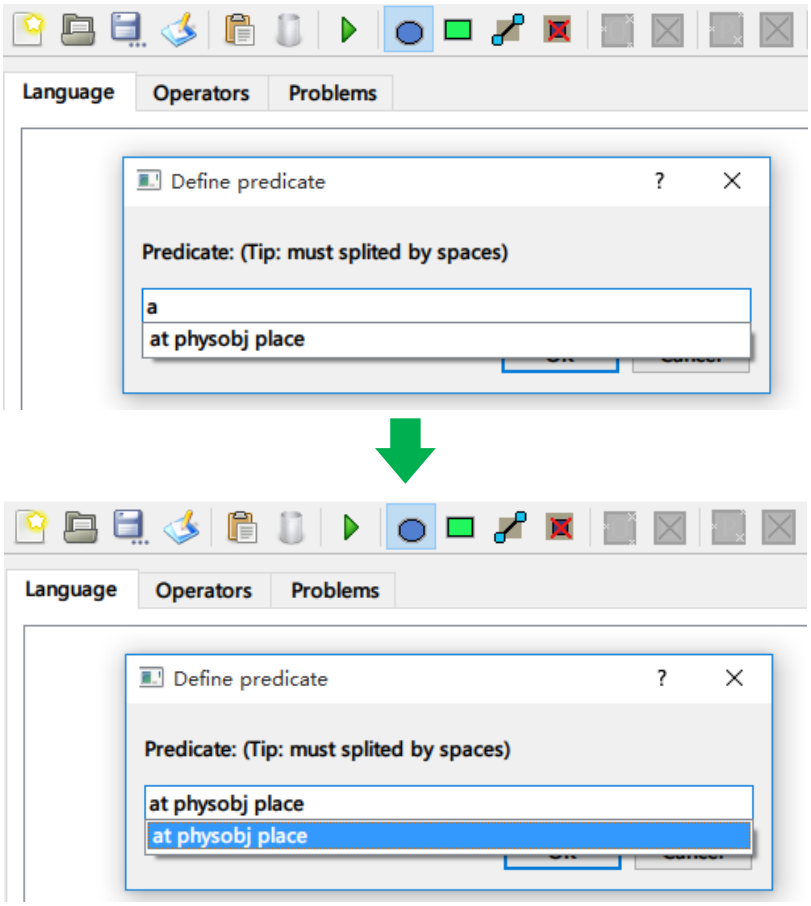

图 7-3 声明表达语言编辑界面中智能提示及自动补全

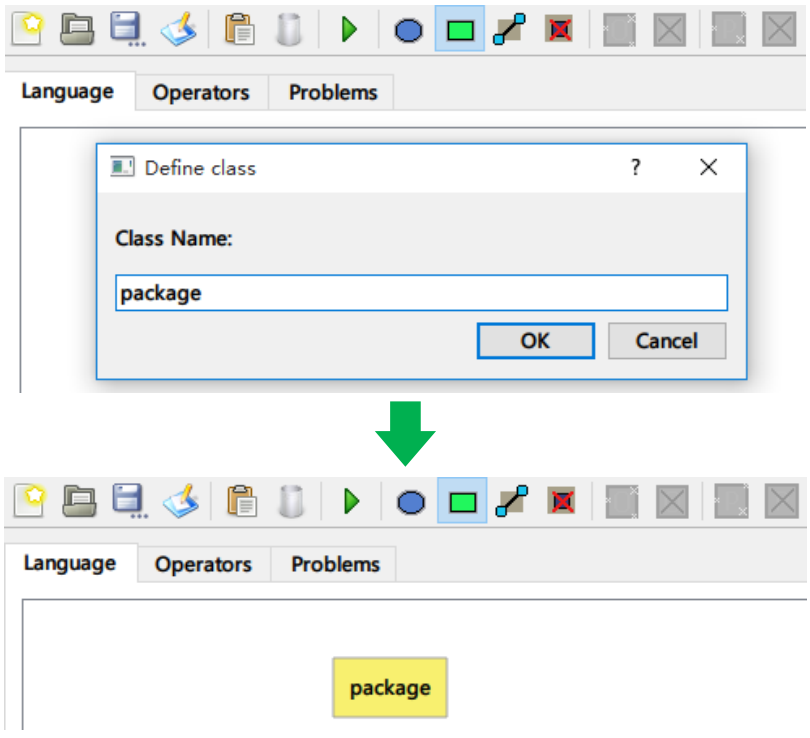

图 7-4 声明表达语言编辑界面中添加类

2) 添加类或变量或对象

在可视化建模编辑界面中，点击工具栏的绿色矩形按钮切换到添加类或变量或对象编辑模式，左键单击编辑界面空白处，打开声明类对话框，输入类，点击确定按钮，将在编辑界面中生成所输入类的对应图形元素，如图 7-4 所示。对于声明类的智能提示及自动补全与声明谓词同理。

3) 添加关联线条

在可视化建模编辑界面中，点击工具栏的黑色线条按钮切换到添加线条编辑模式，左键单击一个椭圆或者矩形并持续按住，移动鼠标至另外一个椭圆或者矩形，释放鼠标，将在编辑界面中生成关联线条，包括类继承关联、谓词参数关联，如图 7-5 所示。

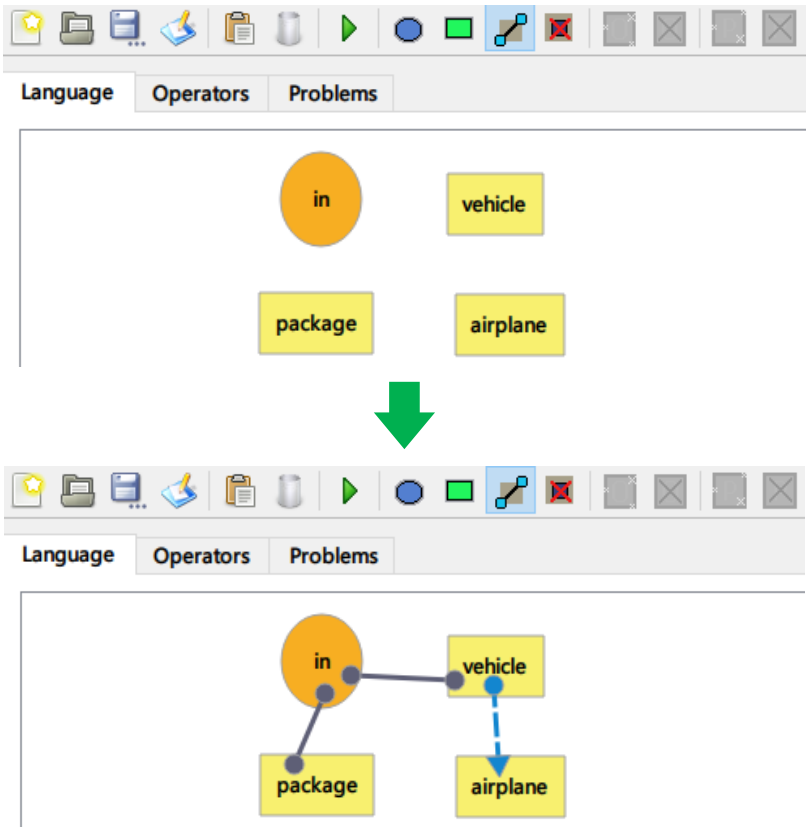

图 7-5 声明表达语言编辑界面中添加关联线条

4) 删除元素

在可视化建模编辑界面中，点击工具栏的红叉按钮切换到删除元素编

辑模式，左键单击一个椭圆或者矩形或者线条，编辑界面中将删除所点击的图形元素，如图 7-6 所示。

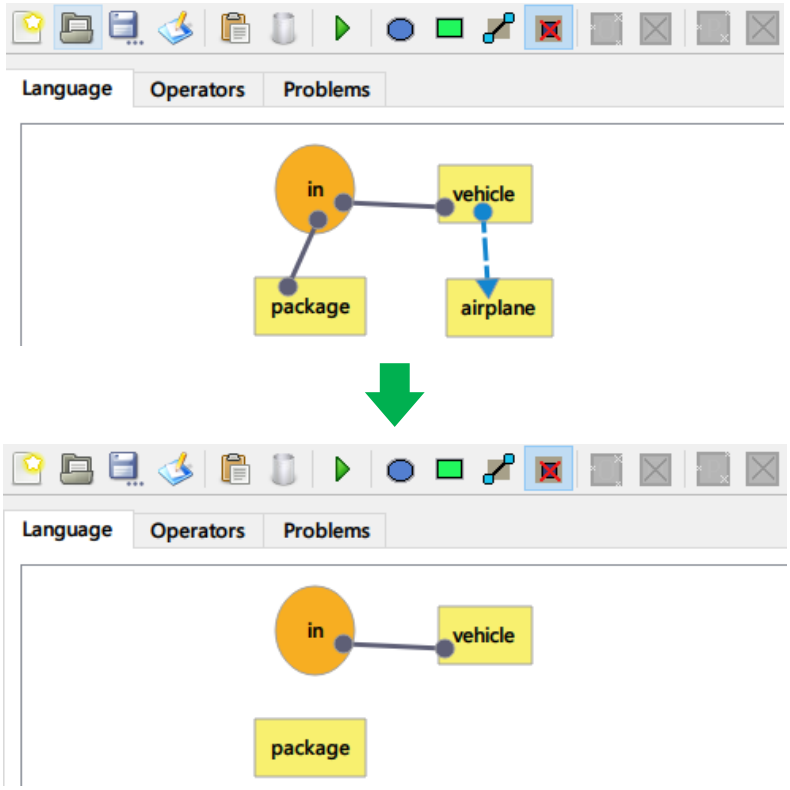

图 7-6 声明表达语言编辑界面中删除元素

需要注意的是，在定义规划动作、定义规划问题的编辑界面中，添加谓词、添加变量或对象的操作依赖于声明表达语言编辑界面中声明的谓词和类，从声明表达语言编辑界面中声明的谓词和类的集合里选择并实例化。

7.2.4 语法约束检查

在可视化建模过程中，可以进行规划领域语法约束检查。在声明表达语言编辑界面中，可以检查类和谓词的定义，对于谓词，可以检查所有重载的谓词的参数个数是否相同。在定义规划动作和定义规划问题的编辑界面中，用户可以检查动作定义、问题定义与声明表达语言界面中的领域定义的一致性，包括所有的类实例（动作定义中的变量、问题定义中的对象）和谓词实例，其中对于谓词实例可以检查谓词的参数列表。约束检查可以输出完备的检查信息、领域统计信息

以在声明表达语言编辑界面中重载谓词为例，点击工具栏的红勾按钮进

行约束检查，如图 7-7 所示。

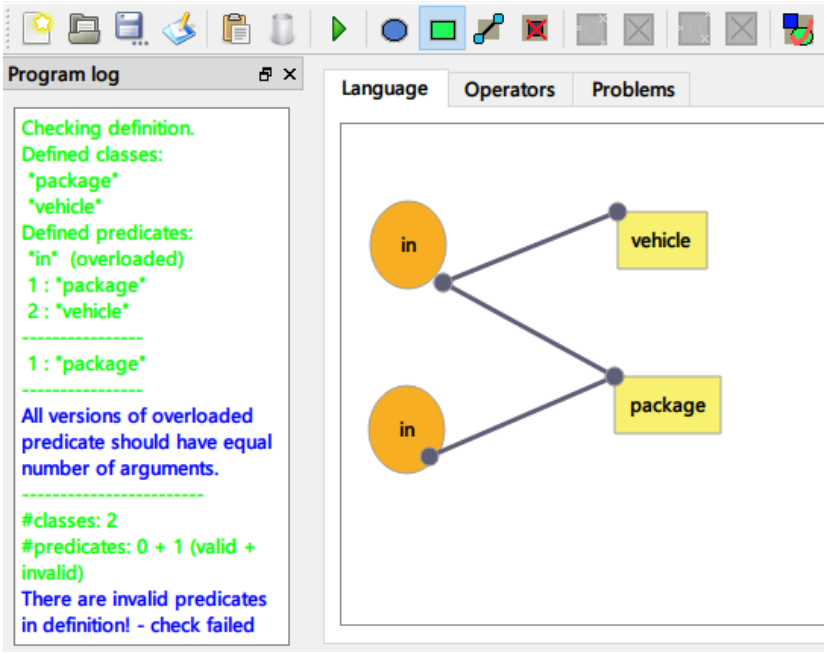

图 7-7 声明表达语言编辑界面中重载谓词约束检查

7.2.5 数据转换

在可视化建模中完成logistics领域所有的表达语言、规划动作、规划问题的建模后，数据转换模块可以将整个领域模型保存为 XML 文件，或者从 XML 文件中导入领域模型到可视化建模中。另外，可以将表达语言、规划动作导出为领域定义 PDDL 文件，将规划问题导出为问题定义 PDDL 文件。各种操作选项如图 7-8 所示。

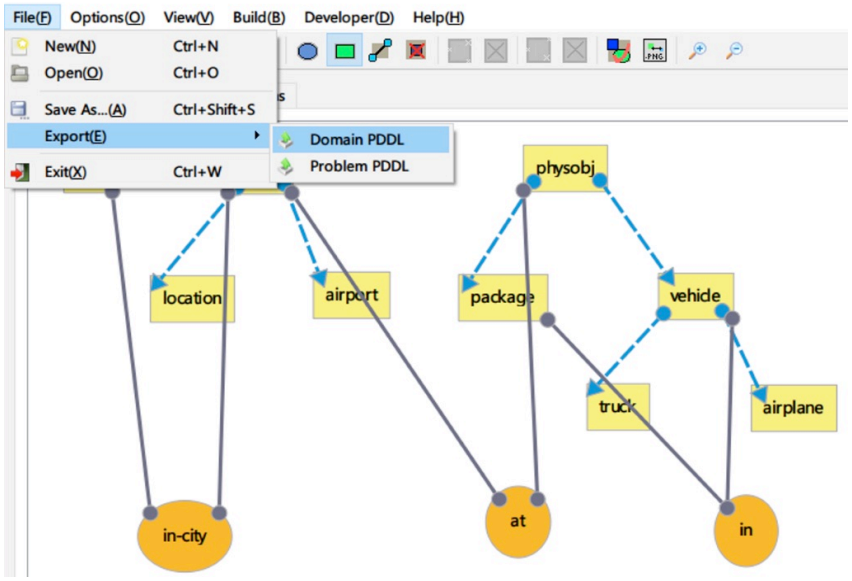

图 7-8 文件操作选项

针对 logistics 领域可视化建模导出的领域定义 PDDL 文件、问题定义 PDDL 文件，详见附录 A2，经过检查与 logistics 领域的源 PDDL 文件的模型一致。

7.2.6 规划

在可视化建模中完成logistics领域的建模后，点击工具栏的绿色三角形按钮打开规划界面，如图 7-9 所示，主要包含 4 个部分：

1) Configures

   规划、规划验证的配置选项，包括是否使用自定义文件、是否规划验证、选择自定义文件。

2) Solve

   规划、规划验证的执行选项，包括选择规划器、设置规划器、执行规划、执行规划验证、清空控制台日志输出以及规划报告。

3) Compile output

   控制台日志输出。

4) Plan Report

   规划的报告。

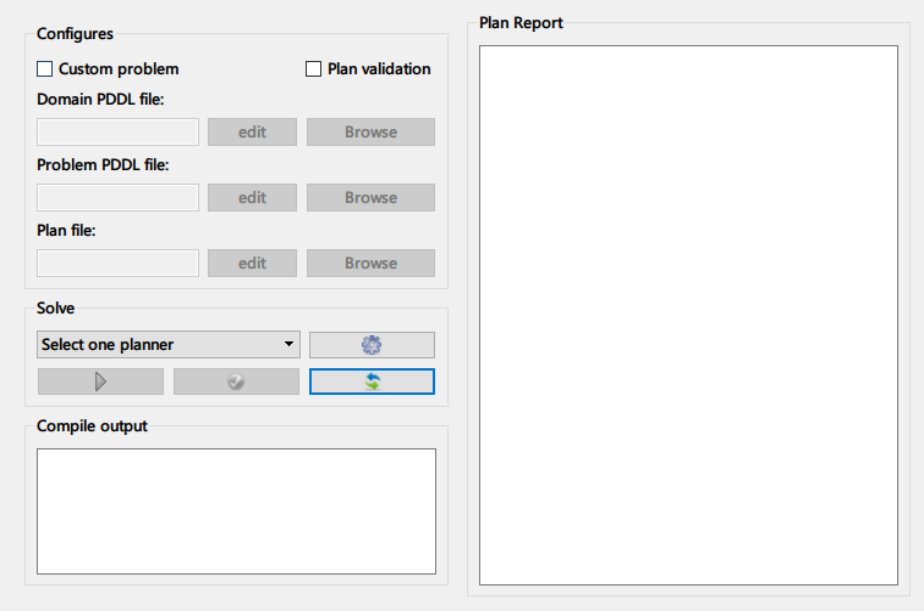

图 7-9 规划界面

这里不使用自定义文件，使用默认的由可视化建模生成的 logistics 领域的 PDDL 文件进行规划。在 Solve 部分的下拉选择框中选择规划器 Metric-FF，点击下拉选择框下方的绿色三角形按钮，执行规划。在 Plan Report 部分中展示规划结果，在 Compile output 部分中展示执行规划的日志输出，如图 7-10 所示。

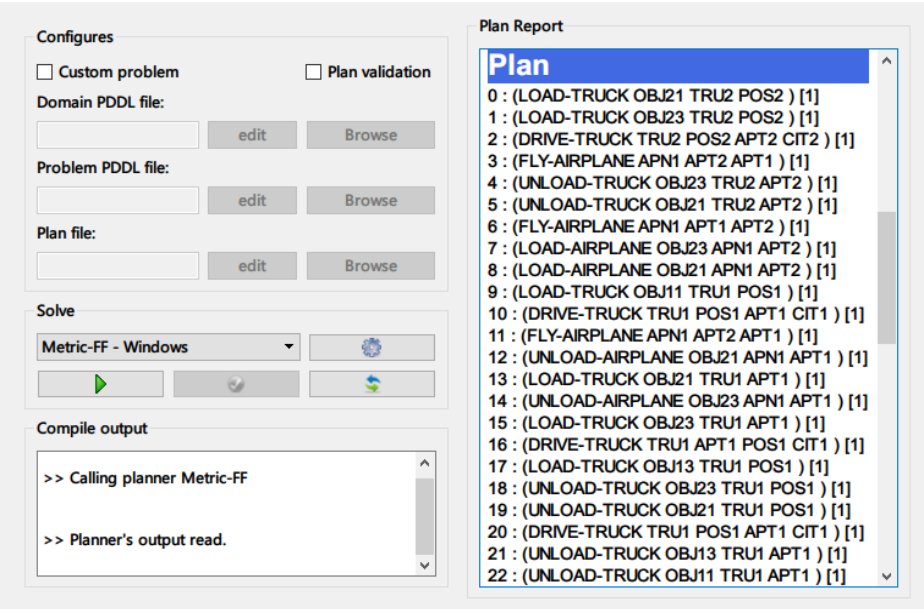

图 7-10 规划界面

由图 7-10 可见，规划成功，在 Plan Report 部分中展示的 Plan 条目内容为规划的动作序列。

7.2.7 规划验证

在 7.2.6 中对 logistics 领域的领域模型执行规划成功后，使用默认的由可视化建模生成的 logistics 领域的 PDDL 文件、默认的由规划模块生成的规划文件进行规划验证。在规划界面的 Configures 部分勾选 Plan validation，点击 Solve 部分中下拉选择框下方的绿勾按钮，执行规划验证，打开规划验证界面，如图 7-11 所示，主要包含 4 个部分：

1) Plan Process

规划的动作序列，白色标识动作为模拟规划问题初始状态、目标状态的虚拟动作，绿色标识动作为验证通过的动作，红色标识动作为验证失败的动作，灰色标识动作为未验证的动作（紧接红色标识动作）。

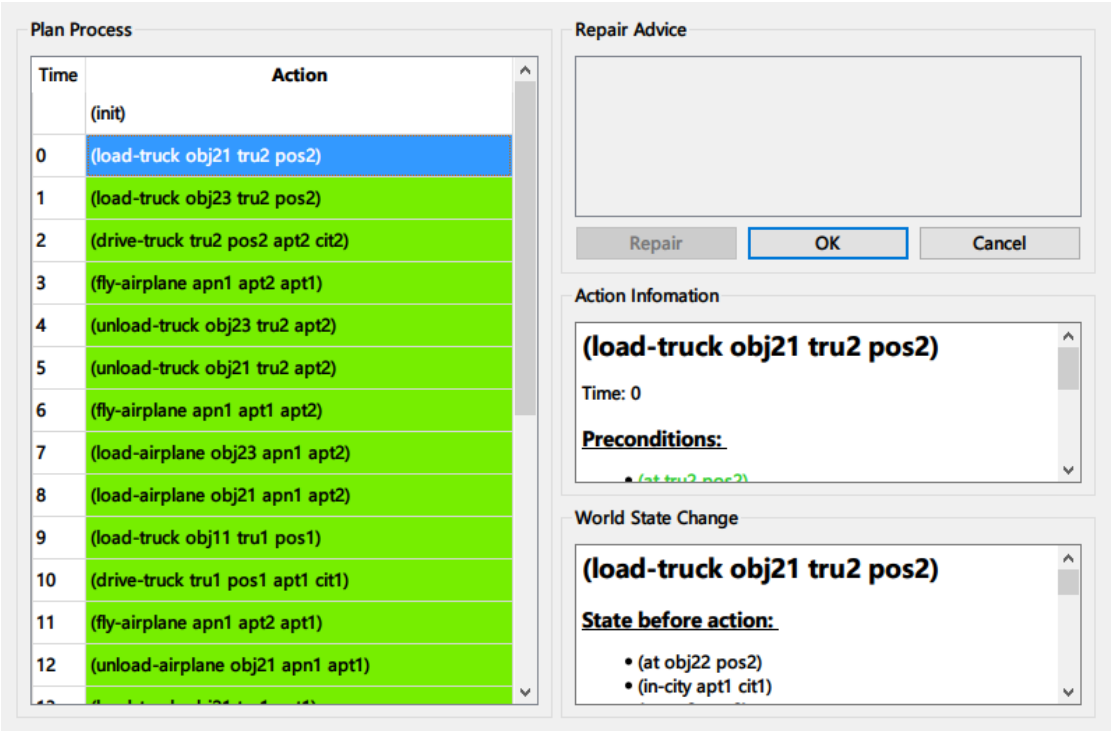

图 7-11 规划验证界面

2) Action Information

规划动作的详细信息，包括原子式、时间、前提条件、前提条件的来源依赖、后置效果、后置效果的取向依赖。

3) World State Change

伴随着规划动作执行的世界状态转移信息，包括动作执行前的世界状态、动作执行后的世界状态、执行动作所增加的状态、执行动作所删除的状态。

4) Repair Advice

针对验证失败的有缺陷的红色标识动作的修复建议，包括执行修复。

点击 Plan Process 部分中动作序列的任一动作，展示动作详细信息以及对应的世界状态转移信息，如图 7-11 所示。

7.2.8 领域模型微调

在 7.2.7 中展示了 logistics 领域规划验证成功的情况。为展示领域模型微调模块的运行情况，现构造存在缺陷的领域模型。

构造样例：针对动作序列中序号为 0 的动作（load-truck obj21 tru2 pos2），该动作的其中一个前提条件为（at tru2 pos2），来源于初始状态，把（at tru2 pos2）从初始状态中剔除。

被构造缺陷后的 logistics 模型是存在缺陷的，所以直接对 logistics 模型生成的 PDDL 文件进行规划是无解的，也就不能生成规划文件，从而进行规划验证。

对于用户来说，为定位 logistics 模型中存在的缺陷并修复，用户基于自己的经验知识，手动构造规划文件中的动作序列，利用存在缺陷的领域模型、自构造的规划文件进行规划验证，如图 7-12 所示。

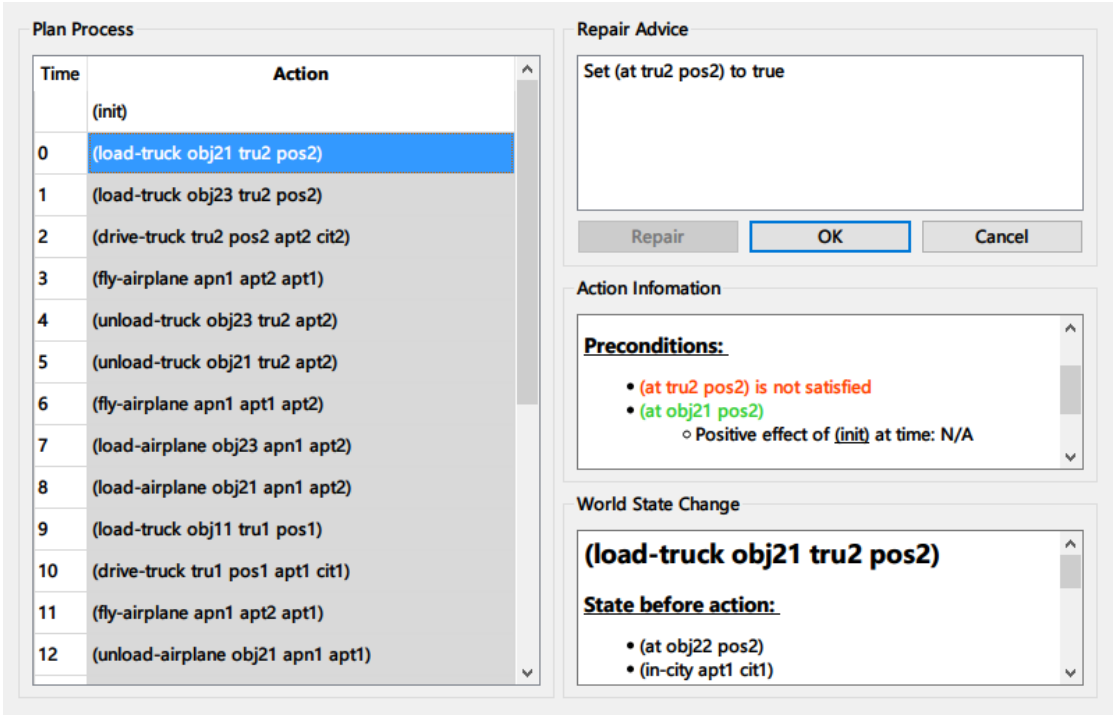

图 7-12 规划验证失败

由图 7-12 可见，在动作序列中序号为 0 的动作（load-truck obj21 tru2 pos2）验证失败，存在的缺陷为其前提条件（at tru2 pos2）不满足，因此 Repair Advice 部分中的修复建议条目为（Set (at tru2 pos2) to true）。选中修复建议条目，点击 Repair 修复按钮，打开规划修复界面，如图 7-13 所示。

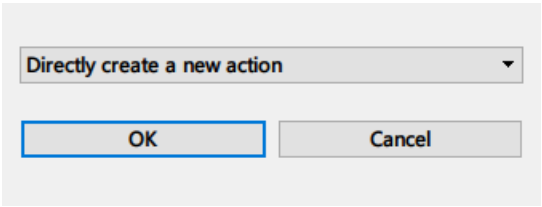

图 7-13 规划修复界面

规划修复界面中提供了 2 种修复选项，包括直接创建新动作、修改已有动作，具体见 6.7。

1) 直接创建新动作

在规划修复界面的修复选项下拉选择框中选择直接创建新动作选项，点击确定，将跳转至可视化建模中的定义规划动作编辑界面，输入新动作名称并点击确定后，将自动生成新动作的所有图形元素，如图 7-

14 所示。

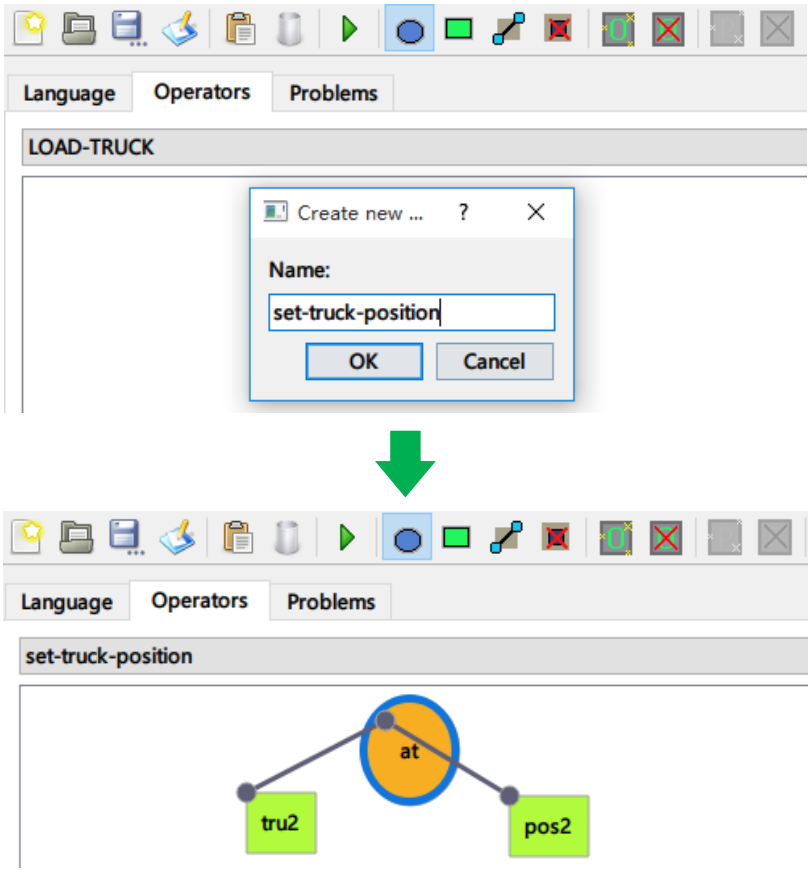

图 7-14 执行直接创建新动作修复选项

对于用户来说，根据修改后的领域模型，重新构造规划文件的动作序列，在原动作序列中序号为 0 的动作（load-truck obj21 tru2 pos2）前插入一个新创建的动作（set-truck-position tru2 pos2），该新动作能产生动作序列中序号为 0 的动作（load-truck obj21 tru2 pos2）所需的前提条件（at tru2 pos2）。利用修改后的领域模型、重新构造的规划文件进行规划验证，此时规划验证成功，如图 7-15 所示。

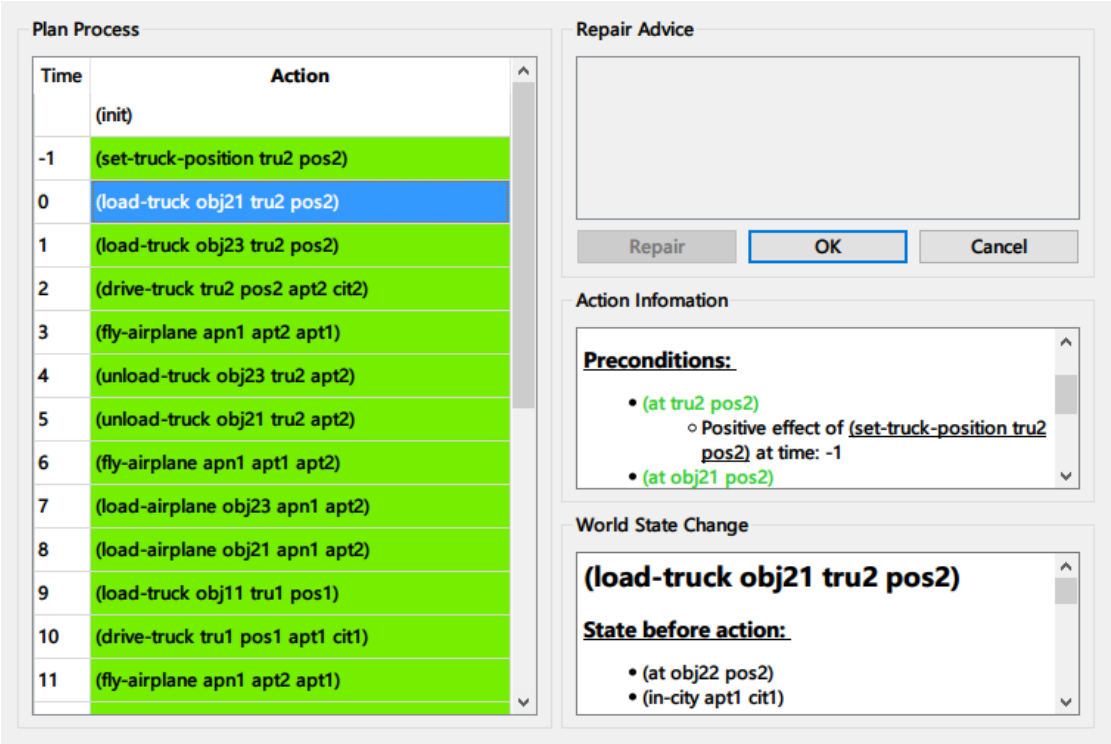

图 7-15 修改领域模型后验证成功

2) 修改已有动作

在规划修复界面的修复选项下拉选择框中选择修改已有动作选项，点击确定，将跳转至可视化建模中的定义规划动作编辑界面，由用户自主修改领域模型，如图 7-16 所示。

以上的 2 种修复选项运行情况，均是使用默认的由可视化建模生成的 PDDL 文件、默认的由规划模块生成的规划文件进行规划验证的，对于使用自定义文件进行规划验证，这 2 种修复选项将作用于领域模型的领域定义 PDDL 文件本身，通过文本编辑器实现，而不是作用于可视化建模的编辑界面，如图 7-17、图 7-18 所示。

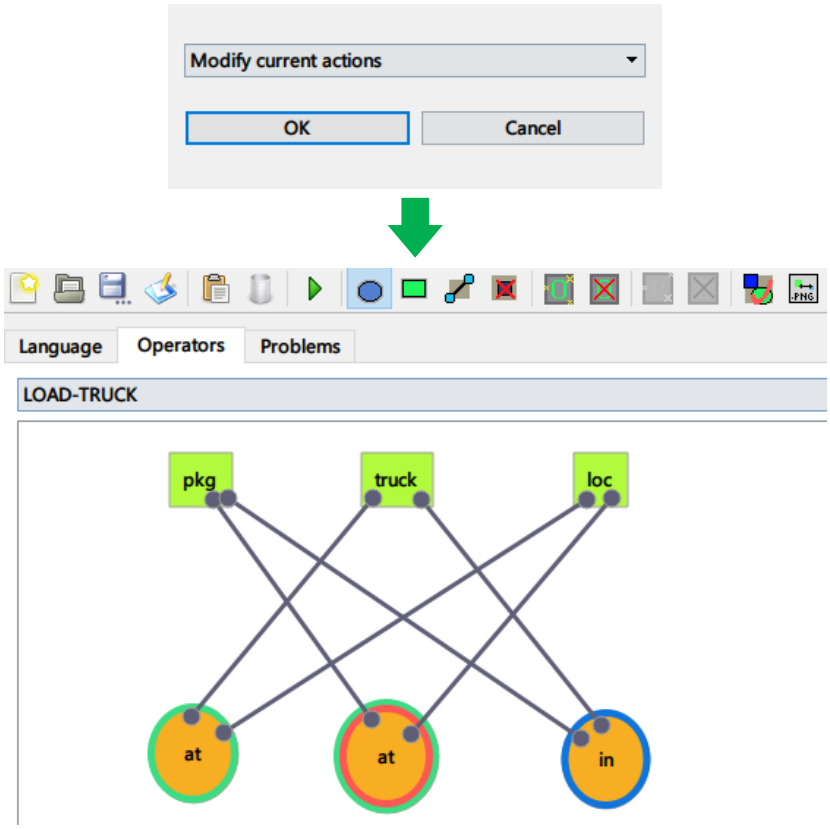

图 7-16 执行修改已有动作修复选项

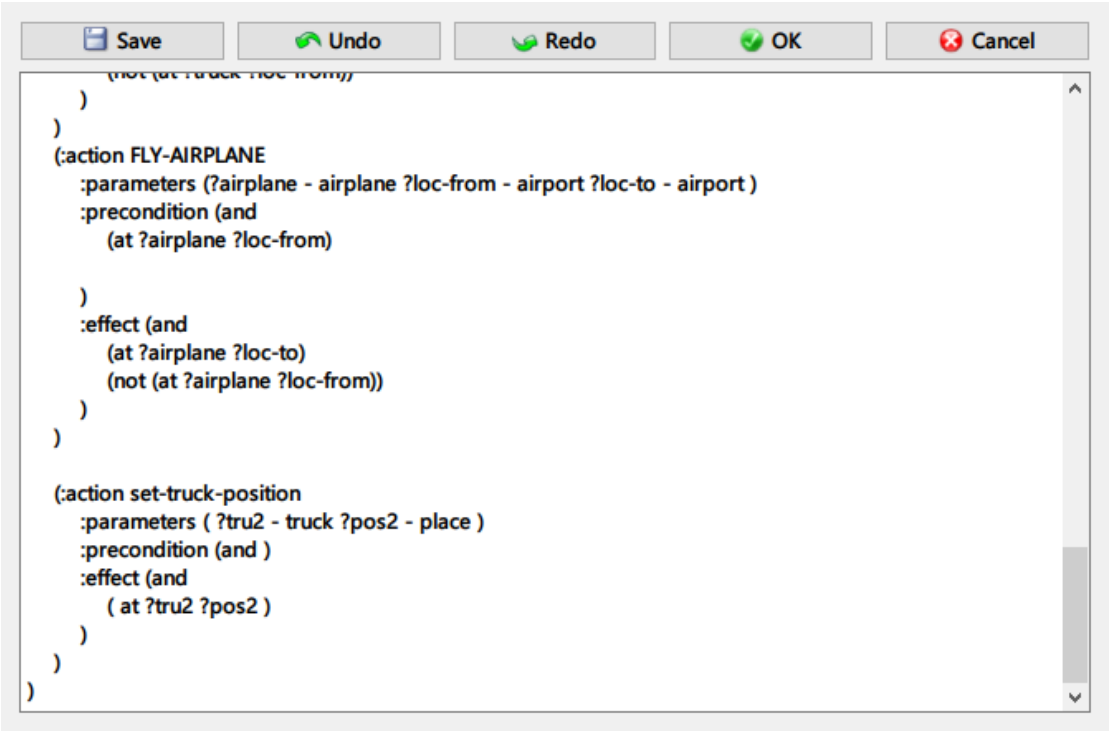

图 7-17 基于自定义文件执行直接创建新动作修复选项

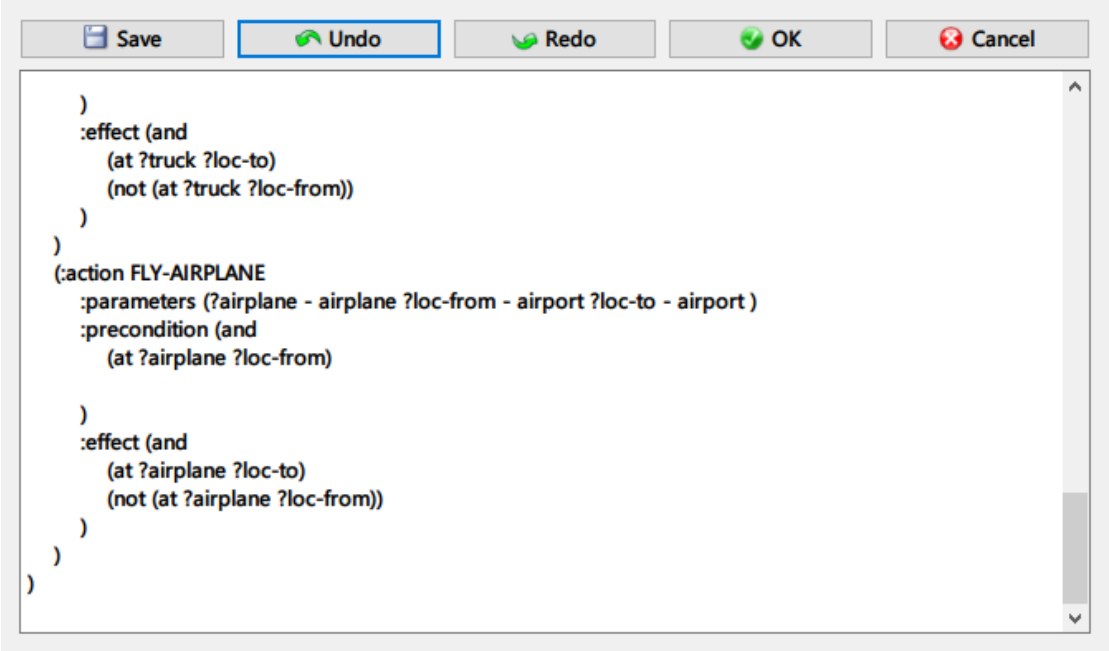

图 7-18 基于自定义文件执行修改已有动作修复选项

7.3 系统测试

本节对人机混合智能规划平台进行黑盒测试。黑盒测试在测试过程中，不考虑程序内部的运行逻辑，通过调用程序用户接口对系统进行测试，根据程序的输出信息是否符合预期结果从而判断程序功能是否可以正常运行。下面展示平台比较关键以及具有代表性的测试点、测试步骤和测试结果，如表 7-4 所示。

表 7-4 系统测试用例及测试结果

编号	所属模块	用例名称	测试步骤	预期结果	实际结果
T001	领域知识库	添加谓词模板	1. 点击菜单栏的开发者选项中的“领域知识库” 2. 点击领域知识库编辑界面的“谓词”板块 3. 点击下方“添加”按钮 4. 输入合法谓词，点击确定	1. 打开领域知识库编辑界面 2. 切换到谓词模板库编辑界面 3. 打开添加谓词对话框 4. 谓词模板库显示新增谓词模板	与预期结果一致
T002	可视化建模	添加谓词	1. 点击工具栏蓝色椭圆按钮 2. 左键单击编辑界面	1. 切换到添加谓词编辑模式 2. 打开声明谓词	与预期结果一致

			空白处 3. 输入合法谓词，点击确定	对话框 3. 编辑界面生成输入谓词的图形元素	
T003	领域知识库	添加谓词—智能提示补全	1. 点击工具栏蓝色椭圆按钮 2. 左键单击编辑界面空白处 3. 输入谓词前缀，在下拉提示框中选中谓词模板，点击确定	1. 切换到添加谓词编辑模式 2. 打开声明谓词对话框 3. 编辑界面生成选中谓词模板的图形元素	与预期结果一致
T004	可视化建模	添加关联线条	1. 点击工具栏黑色线条按钮 2. 左键单击一个矩形并持续按住 3. 移动鼠标到另外一个矩形上 4. 释放鼠标	1. 切换到添加线条编辑模式 2. 编辑界面在鼠标处生成临时的绿点 3. 编辑界面沿鼠标移动轨迹生成临时的绿色线条 4. 编辑界面生成类继承关联线条	与预期结果一致
T005	语法约束检查	检查领域语法约束	1. 点击工具栏红勾按钮	1. 系统日志窗口显示领域检查结果	与预期结果一致
T006	数据转换	导出领域定义 PDDL 文件	1. 点击菜单栏的文件的“导出”选项中的“领域 PDDL” 2. 选择导出文件夹，输入 PDDL 文件名，点击确定	1. 打开导出领域定义 PDDL 文件对话框 2. 在所选的导出文件夹中生成了命名为所输入的文件名的领域定义 PDDL 文件	与预期结果一致
T007	规划	执行规划	1. 点击规划界面的绿色三角形按钮	1. 规划界面的 Plan Report 部分窗口显示详细的规划结果报告	与预期结果一致
T008	规划验证	执行规划验证	1. 点击规划界面的绿勾按钮 2. 点击选中规划验证	1. 打开规划验证界面，显示规划的动作序列	与预期结果一致

			界面中 Plan Process 窗口的动作序列中的任意动作	2. 显示所选中动作的详细信息以及世界状态变化	
T009	规划领域微调	创建新动作—默认文件	1. 点击选中规划验证界面 Repair Advice 窗口中的任意修复建议条目，点击“修复” 2. 在修复界面的修复选项下拉选择框中选择直接创建新动作选项，点击确定 3. 在创建新动作对话框中输入新动作名称，点击确定	1. 打开修复界面 2. 跳转至可视化建模中的定义规划动作编辑界面，打开创建新动作对话框 3. 定义规划动作编辑界面中自动生成新动作的所有图形元素	与预期结果一致
T010	规划领域微调	创建新动作—自定义文件	1. 点击选中规划验证界面 Repair Advice 窗口中的任意修复建议条目，点击“修复” 2. 在修复界面的修复选项下拉选择框中选择直接创建新动作选项，点击确定	1. 打开修复界面 2. 跳转至领域定义 PDDL 文件的文本编辑界面，自动生成并添加新动作的 PDDL 语法文本	与预期结果一致

结合表 7-4 中的 T003 测试用例，用户在使用平台的可视化建模进行规划领域模型建模时，无需从构建原子性的领域知识元素开始建模，通过平台的领域知识库提供的智能提示以及自动补全，可以直接实例化领域知识库中领域知识模板，从而快速地生成领域知识的相应图形元素，达到快速建模领域模型的目的，最终提升可视化建模的效率。

结合表 7-4 中的 T009、T010 测试用例，基于人在回路智能规划 HILP 的思想，用户在规划失败后，可以根据自己的经验知识调整并构造理想的规划文件，使用领域定义 PDDL 文件、问题定义 PDDL 文件以及构造的规划文件进行规划验证，验证识别规划领域中可能存在的缺陷，平台根据规划验证的结果提供修复的建议并提供修复选项，用户可以根据需要选择执行修复选项，最终达到微调领域模型的目的，构成提升领域模型的科学合理性的反馈回路。

第 8 章　总结与展望

本章先对平台的相关内容进行总结，然后提出目前人机混合智能规划平台需要改进的方面以及对后续研究工作的展望。

8.1 总结

近年来随着规划领域建模的发展，规划领域的可视化建模已经被应用于智能规划与调度的知识工程中，但是目前专注于这方面的知识工程应用普遍存在可视化建模的效率不高、欠缺对规划领域的调整的问题。针对这些问题，平台分析目前智能规划与调度知识工程、规划领域建模以及规划分析的相关工作，设计并实现人机混合智能规划平台，构建领域知识库以提升可视化建模的效率，基于 HILP，通过集成规划、规划验证功能实现对规划领域的调整。

平台的优势在于：

1) 在规划领域设计层次中构建领域知识库抽象层。依赖于领域知识库，设计人员进行可视化建模时，系统可以提供智能提示以及自动补全，从而明显地提高可视化建模的效率。

2) 基于人机混合智能的人在回路智能规划 HILP，通过集成规划、规划验证功能实现对规划领域的调整。支持规划、规划可视化以及规划验证，验证规划可能存在的缺陷，提供修复建议并支持设计人员根据修复建议直接对规划领域进行交互式微调。

3) 提供规划领域可视化建模的完整解决方案。提供领域知识库、可视化建模、约束检查、数据转换、规划求解、规划验证、领域模型微调等功能，方便设计人员在完整的集成开发环境中建模规划领域。

综上所述，人机混合智能规划平台利用领域知识库提升可视化建模的效率，利用 HILP 的思想，通过规划、规划验证实现对规划领域进行调整，能有效地解决目前相关工作可视化建模的效率不高、欠缺对规划领域的调整的问题，大大

降低设计人员进行领域建模的难度，以及协助设计人员设计出科学合理的领域模型。

8.2 展望

作为一个人机混合智能规划平台，不仅涉及到软件工程方面的实现，还需要对智能规划与调度、规划领域定义语言、规划分析等方面有充分深入的研究，涉及面广，因此目前平台还存在着需要改进的方面，以及后续可以拓展的研究：

1) 目前平台仅支持经典的 STRIPS 型规划问题。为了能支持处理更多种类的规划问题，后续需要增加更多对 PDDL 语言特性的支持。

2) 目前平台在规划和规划验证模块中，对于本身存在缺陷的领域模型，不能进行规划然后自动地产生规划文件，从而进行规划验证。需要通过设计人员根据自己的经验知识构造规划文件，从而利用规划验证的结果对领域模型进行调整。为了更加方便用户应对这种应用场景，后续需要优化规划和规划验证的设计。

3) 目前平台在领域模型微调模块中提供的修复选项只有直接创建新动作和修改已有动作。为了提高领域模型微调的准确度，后续需要根据规划动作序列之间的复杂依赖关系优化领域模型微调的算法。

4) 为了提高平台规划领域建模的成功率，后续可以集成更多的规划器、规划验证器，以及考虑多个规划器、规划验证器之间的相互作用和交叉验证。

参考文献

[1] Glinský R. Visualization and verification of plans[J]. 2011.

[2] Shah M, Chrpa L, Jimoh F, et al. Knowledge engineering tools in planning: State-of-the-art and future challenges[J]. Knowledge Engineering for Planning and Scheduling, 2013, 53.

[3] 赵广立. 混合智能：人工智能研究的下一站[N]. 中国科学报, 2017-08-03(5).

[4] Plch T, Chomut M, Brom C, et al. Inspect, edit and debug PDDL documents: Simply and efficiently with PDDL studio[J]. System Demonstrations and Exhibits at ICAPS, 2012: 15-18.

[5] Barreiro J, Boyce M, Do M, et al. EUROPA: a platform for AI planning, scheduling, constraint programming, and optimization[J]. 4th International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS), 2012.

[6] Frank J, Jónsson A. Constraint-based attribute and interval planning[J]. Constraints, 2003, 8(4): 339-364.

[7] Bernardini S, Smith D E. Developing domain-independent search control for europa2[C]. Proceedings of the Workshop on Heuristics for Domain-independent Planning at ICAPS. 2007, 7.

[8] Smith D E, Frank J, Cushing W. The anml language[J]. Proceedings of ICAPS, 2008.

[9] Vaquero T S, Silva J R, Tonidandel F, et al. itSIMPLE: towards an integrated design system for real planning applications[J]. The Knowledge Engineering Review, 2013, 28(2): 215-230.

[10] Vaquero T S, Tonaco R, Costa G, et al. itSIMPLE4. 0: Enhancing the modeling experience of planning problems[C]. System Demonstration–Proceedings of the 22nd International Conference on Automated Planning & Scheduling (ICAPS-12). 2012: 11-14.

[11] Vaquero T S, Silva J R, Beck J C. Analyzing Plans and Planners in itSIMPLE3. 1[C]. Proceeding of the ICAPS 2010 Knowledge Engineering for Planning and Scheduling Workshop. Toronto. Canada. 2010: 45-52.

[12] Simpson R M, Kitchin D E, McCluskey T L. Planning domain definition using GIPO[J]. The Knowledge Engineering Review, 2007, 22(2): 117-134.

[13] McCluskey T L, Simpson R M. Combining constraint based and classical formulations for planning domains[C]. Proceedings of The 25th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG2006). 2006: 55-65.

[14] Vodrázka J, Chrpa L. Visual design of planning domains[C]. Proceedings of ICAPS 2010 workshop on Scheduling and Knowledge Engineering for Planning and Scheduling (KEPS). 2010: 68-69.

[15] Vaquero T S, Silva J R, Beck J C. A brief review of tools and methods for knowledge engineering for planning & scheduling[J]. KEPS 2011, 2011: 7.

[16] Glinský R, Barták R. Visplan–interactive visualisation and verification of plans[J]. KEPS 2011, 2011: 134.

[17] Gerevini A E, Saetti A. An interactive environment for plan visualization and generation: Inlpg[C]. Working notes of Eighteenth International Conference on Automated Planning & Scheduling (ICAPS-08)-System Demonstration, Sydney (Australia). 2008.

[18] Howey R, Long D, Fox M. VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL[C]. Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th

IEEE International Conference on. IEEE, 2004: 294-301.
[19] Fikes R E, Nilsson N J. STRIPS: A new approach to the application of theorem proving to problem solving[J]. Artificial intelligence, 1971, 2(3-4): 189-208.
[20] Pednault E P D. Formulating multiagent, dynamic-world problems in the classical planning framework[M]. Reasoning about actions & plans. 1987: 47-82.
[21] Gelfond M, Lifschitz V. Action languages[J]. 1998.
[22] Pednault E P D. ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus[J]. Kr, 1989, 89: 324-332.
[23] McDermott D, Ghallab M, Howe A, et al. PDDL-the planning domain definition language[J]. 1998.
[24] Fox M, Long D. PDDL2. 1: An extension to PDDL for expressing temporal planning domains[J]. Journal of artificial intelligence research, 2003.
[25] 饶东宁, 蒋志华, 姜云飞. 规划领域定义语言的演进综述[J]. 计算机工程与应用, 2010, 46(22): 23-25.
[26] Hoffmann J, Georges-Köhler-Allee G. PDDL2. 2: The Language for the Classical Part of the 4th International Planning Competition[J]. 2004.
[27] 刘日仙, 谷文祥, 欧华杰. 人工智能规划领域定义语言[C]. 全国理论计算机科学学术年会, 2004.
[28] Gerevini A, Long D. Preferences and soft constraints in PDDL3[C]. ICAPS workshop on planning with preferences and soft constraints. 2006: 46-53.
[29] Gerevini A, Long D. Plan constraints and preferences in PDDL3[R]. Technical Report 2005-08-07, Department of Electronics for Automation, University of Brescia, Brescia, Italy, 2005.
[30] Gerevini A, Long D. BNF description of PDDL3. 0[J]. Unpublished manuscript from the IPC-5 website, 2005.
[31] Helmert M. Changes in PDDL 3.1[J]. Unpublished summary from the IPC-2008 website, 2008.
[32] Kovacs D L. BNF Definition of PDDL3. 1: completely corrected, without comments[J]. Unpublished manuscript from the IPC-2011 website, 2011.
[33] Hoffmann J. The Metric-FF Planning System: Translating``Ignoring Delete Lists"to Numeric State Variables[J]. Journal of Artificial Intelligence Research, 2003, 20: 291-341.
[34] Hoffmann J, Nebel B. The FF planning system: Fast plan generation through heuristic search[J]. Journal of Artificial Intelligence Research, 2001, 14: 253-302.
[35] Hoffmann J. Extending FF to numerical state variables[C]. ECAI. 2002: 571-575.
[36] Kautz H, Selman B. BLACKBOX: A new approach to the application of theorem proving to problem solving[C]. AIPS98 Workshop on Planning as Combinatorial Search. 1998, 58260: 58-60.
[37] Gerevini A, Serina I. LPG: A Planner Based on Local Search for Planning Graphs with Action Costs[C]. AIPS. 2002, 2: 281-290.
[38] Gerevini A, Saetti A, Serina I, et al. LPG-TD: a fully automated planner for PDDL2. 2 domains[C]. In Proc. of the 14th Int. Conference on Automated Planning and Scheduling (ICAPS-04) International Planning Competition abstracts. 2004.
[39] Fox M, Long D. PDDL+: Modeling continuous time dependent effects[C]. Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space. 2002, 4: 34.
[40] Kambhampati S, Talamadupula K. Human-in-the-Loop Planning and Decision Support[R]. http://rakaposhi.eas.asu.edu/hilp-tutorial/index.htm, AAAI 2015 Tutorial Forum: Tutorial SP4.

附 录 A1

表 A1-1 可视化建模图形信息 XML 标签描述

标签	描述			
<domain>	定义规划领域			
	属性			
	名称	类型	缺省值	描述
	name	NMTOKEN	Unnamed	规划领域的名称
	子标签			
	标签		描述	
	<description>		规划领域的描述	
	<definition>		定义规划领域的表达语言	
	<operators>		定义规划领域的动作集合	
	<problems>		定义规划领域的问题集合	
<definition>	定义规划领域的表达语言			
	属性			
	名称	类型	缺省值	描述
	无			
	子标签			
	标签		描述	
	<diagram>		图形	
<operators>	定义规划领域的动作集合			
	属性			
	名称	类型	缺省值	描述
	无			
	子标签			
	标签		描述	
	<action>		定义规划领域的动作	
<problems>	定义规划领域的问题集合			
	属性			
	名称	类型	缺省值	描述
	无			
	子标签			
	标签		描述	
	<task>		定义规划领域的问题	
<action>	定义规划领域的动作			
	属性			
	名称	类型	缺省值	描述
	name	NMTOKEN	无	动作的名称
	子标签			
	标签		描述	
	<diagram>		图形	
<task>	定义规划领域的问题			
	属性			

	名称	类型	缺省值	描述
	name	NMTOKEN	无	问题的名称
	子标签			
	标签		描述	
	<diagram>		图形	
<diagram>	图形			
	属性			
	名称	类型	缺省值	描述
	无			
	子标签			
	标签		描述	
	<node>		节点图形元素	
	<edge>		线条图形元素	
<node>	节点图形元素			
	属性			
	名称	类型	缺省值	描述
	id	NMTOKEN	无	节点的唯一编号
	子标签			
	标签		描述	
	<label>		节点的文本	
	<type>		节点的类型	
	<pos>		节点的坐标	
	<connections>		节点的关联	
	<set>		谓词节点所属的动作世界状态集合	
	<class>		类实例节点所属的类	
	<state>		谓词节点所属的问题世界状态集合	
<edge>	线条图形元素			
	属性			
	名称	类型	缺省值	描述
	id	NMTOKEN	无	线条的唯一编号
	子标签			
	标签		描述	
	<purpose>		线条关联的属性	
	<start>		线条的起点	
	<end>		线条的终点	
<pos>	节点的坐标			
	属性			
	名称	类型	缺省值	描述
	无			
	子标签			
	标签		描述	
	<x>		节点的 x 轴坐标	
	<y>		节点的 y 轴坐标	
<connections>	节点的关联			
	属性			
	名称	类型	缺省值	描述
	无			
	子标签			
	标签		描述	

<table>
<tr><td rowspan="2"></td><td colspan="2"><starts></td><td>以该节点为线条起点的关联线条</td></tr>
<tr><td colspan="2"><ends></td><td>以该节点为线条终点的关联线条</td></tr>
<tr><td rowspan="8"><starts></td><td colspan="4">以该节点为线条起点的关联线条</td></tr>
<tr><td colspan="4">属性</td></tr>
<tr><td>名称</td><td>类型</td><td>缺省值</td><td>描述</td></tr>
<tr><td>argn</td><td>NMTOKEN</td><td>无</td><td>谓词参数关联的次序</td></tr>
<tr><td colspan="4">子标签</td></tr>
<tr><td colspan="2">标签</td><td colspan="2">描述</td></tr>
<tr><td colspan="4">无</td></tr>
<tr><td rowspan="8"><start></td><td colspan="4">线条的起点</td></tr>
<tr><td colspan="4">属性</td></tr>
<tr><td>名称</td><td>类型</td><td>缺省值</td><td>描述</td></tr>
<tr><td>nid</td><td>NMTOKEN</td><td>无</td><td>线条起点的节点编号</td></tr>
<tr><td colspan="4">子标签</td></tr>
<tr><td colspan="2">标签</td><td colspan="2">描述</td></tr>
<tr><td colspan="2"><pos></td><td colspan="2">起点的坐标</td></tr>
<tr><td rowspan="8"><end></td><td colspan="4">线条的终点</td></tr>
<tr><td colspan="4">属性</td></tr>
<tr><td>名称</td><td>类型</td><td>缺省值</td><td>描述</td></tr>
<tr><td>nid</td><td>NMTOKEN</td><td>无</td><td>线条终点的节点编号</td></tr>
<tr><td colspan="4">子标签</td></tr>
<tr><td colspan="2">标签</td><td colspan="2">描述</td></tr>
<tr><td colspan="2"><pos></td><td colspan="2">终点的坐标</td></tr>
</table>

附 录 A2

1. logistics 领域的源领域定义 PDDL

```
;; logistics domain Typed version.
;;

(define (domain logistics)
  (:requirements :strips :typing)
  (:types truck
          airplane - vehicle
          package
          vehicle - physobj
          airport
          location - place
          city
          place
          physobj - object)

  (:predicates  (in-city ?loc - place ?city - city)
        (at ?obj - physobj ?loc - place)
```

```
        (in ?pkg - package ?veh - vehicle))

(:action LOAD-TRUCK
   :parameters    (?pkg - package ?truck - truck ?loc - place)
   :precondition  (and (at ?truck ?loc) (at ?pkg ?loc))
   :effect        (and (not (at ?pkg ?loc)) (in ?pkg ?truck)))

(:action LOAD-AIRPLANE
  :parameters   (?pkg - package ?airplane - airplane ?loc - place)
  :precondition (and (at ?pkg ?loc) (at ?airplane ?loc))
  :effect       (and (not (at ?pkg ?loc)) (in ?pkg ?airplane)))

(:action UNLOAD-TRUCK
  :parameters   (?pkg - package ?truck - truck ?loc - place)
  :precondition (and (at ?truck ?loc) (in ?pkg ?truck))
  :effect       (and (not (in ?pkg ?truck)) (at ?pkg ?loc)))

(:action UNLOAD-AIRPLANE
  :parameters    (?pkg - package ?airplane - airplane ?loc - place)
  :precondition  (and (in ?pkg ?airplane) (at ?airplane ?loc))
  :effect        (and (not (in ?pkg ?airplane)) (at ?pkg ?loc)))

(:action DRIVE-TRUCK
  :parameters (?truck - truck ?loc-from - place ?loc-
to - place ?city - city)
  :precondition
   (and (at ?truck ?loc-from) (in-city ?loc-from ?city) (in-city ?loc-
to ?city))
  :effect
   (and (not (at ?truck ?loc-from)) (at ?truck ?loc-to)))

(:action FLY-AIRPLANE
  :parameters (?airplane - airplane ?loc-from - airport ?loc-to - airport)
  :precondition
   (at ?airplane ?loc-from)
  :effect
   (and (not (at ?airplane ?loc-from)) (at ?airplane ?loc-to)))
)
```

2. logistics 领域的源问题定义 PDDL

```
(define (problem logistics-4-0)
(:domain logistics)
(:objects
 apn1 - airplane
 apt1 apt2 - airport
 pos2 pos1 - location
 cit2 cit1 - city
 tru2 tru1 - truck
 obj23 obj22 obj21 obj13 obj12 obj11 - package)

(:init (at apn1 apt2) (at tru1 pos1) (at obj11 pos1)
 (at obj12 pos1) (at obj13 pos1) (at tru2 pos2) (at obj21 pos2) (at obj22 pos2
)
 (at obj23 pos2) (in-city pos1 cit1) (in-city apt1 cit1) (in-city pos2 cit2)
 (in-city apt2 cit2))

(:goal (and (at obj11 apt1) (at obj23 pos1) (at obj13 apt1) (at obj21 pos1)))

)
```

3. 可视化建模产生的 logistics 领域的领域定义 PDDL

```
; logistics domain Typed version.

(define (domain logistics )
    (:requirements :strips :typing :negative-preconditions)
    (:types
        physobj place city - object
        vehicle package - physobj
        location airport - place
        airplane truck - vehicle
    )
    (:predicates
        (in ?a - package ?b - vehicle)
        (in-city ?a - place ?b - city)
        (at ?a - physobj ?b - place)
    )
    (:action LOAD-TRUCK
        :parameters (?pkg - package ?truck - truck ?loc - place )
        :precondition (and
            (at ?pkg ?loc)
            (at ?truck ?loc)

        )
        :effect (and
            (in ?pkg ?truck)
            (not (at ?pkg ?loc))
        )
    )
    (:action LOAD-AIRPLANE
        :parameters (?pkg - package ?airplane - airplane ?loc - place )
        :precondition (and
            (at ?pkg ?loc)
            (at ?airplane ?loc)

        )
        :effect (and
            (in ?pkg ?airplane)
            (not (at ?pkg ?loc))
        )
    )
    (:action UNLOAD-TRUCK
        :parameters (?pkg - package ?truck - truck ?loc - place )
        :precondition (and
            (at ?truck ?loc)
            (in ?pkg ?truck)

        )
        :effect (and
            (at ?pkg ?loc)
            (not (in ?pkg ?truck))
        )
    )
    (:action UNLOAD-AIRPLANE
        :parameters (?pkg - package ?airplane - airplane ?loc - place )
        :precondition (and
            (at ?airplane ?loc)
            (in ?pkg ?airplane)

        )
        :effect (and
            (at ?pkg ?loc)
            (not (in ?pkg ?airplane))
        )
    )
    (:action DRIVE-TRUCK
        :parameters (?truck - truck ?loc-from - place ?loc-to - place
```

```
                        ?city - city )
        :precondition (and
            (in-city ?loc-from ?city)
            (in-city ?loc-to ?city)
            (at ?truck ?loc-from)

        )
        :effect (and
            (at ?truck ?loc-to)
            (not (at ?truck ?loc-from))
        )
    )
    (:action FLY-AIRPLANE
        :parameters (?airplane - airplane ?loc-from - airport
                        ?loc-to - airport )
        :precondition (and
            (at ?airplane ?loc-from)

        )
        :effect (and
            (at ?airplane ?loc-to)
            (not (at ?airplane ?loc-from))
        )
    )
)
```

4. 可视化建模产生的 logistics 领域的问题定义 PDDL

```
(define (problem logistics-4-0)
    (:domain logistics )
    (:objects
         obj23 obj22 obj21 obj13 obj12 obj11 - package
         tru2 tru1 - truck
         apn1 - airplane
         cit2 cit1 - city
         pos2 pos1 - location
         apt2 apt1 - airport
    )
    (:init
        (at obj11 pos1)
        (at obj12 pos1)
        (at obj13 pos1)
        (at obj21 pos2)
        (at obj22 pos2)
        (at tru1 pos1)
        (at tru2 pos2)
        (at obj23 pos2)
        (at apn1 apt2)
        (in-city pos1 cit1)
        (in-city pos2 cit2)
        (in-city apt2 cit2)
        (in-city apt1 cit1)
    )
    (:goal
        (and
            (at obj13 apt1)
            (at obj21 pos1)
            (at obj11 apt1)
            (at obj23 pos1)
        )
    )
)
```