STRIPS PLANNER

YASMINE DALY & RABEB SDIRI

A REPORT SUBMITTED AS PART OF THE REQUIREMENTS FOR OUR MULTI-AGENT SYSTEMS PROJECT
M2 SIA
AT GUSTAVE EIFFEL UNIVERISTY

.

December 2023

Supervisor Mr. Mahdi Zargayouna

Abstract

This report presents an innovative approach to robotic planning by integrating STRIPS (Stanford Research Institute Problem Solver) language into the PyBullet planning framework. We explore the translation of complex, cluttered environment manipulation tasks into the formalized structure of STRIPS, encompassing action descriptions, state representations, and goal definitions. Emphasizing the computational efficiency and problem-solving efficacy of this integration, the report provides a detailed analysis through case studies in simulated environments. The findings demonstrate the potential of STRIPS planning in enhancing robotic decision-making and execution, marking a significant advancement in AI and robotics research.

Contents

List of Figures

Chapter 1

General Introduction

In recent advancements in robotics and artificial intelligence, planning algorithms play a pivotal role in navigating complex and dynamic environments. Among these, STRIPS (Stanford Research Institute Problem Solver) planning, a seminal framework in AI, has been instrumental in formalizing and solving decision-making problems. This report delves into the integration of STRIPS planning paradigms with the PyBullet-planning project, a sophisticated manipulation planning algorithm designed for operating in cluttered and uncertain environments. We explore the adaptation of this project to the STRIPS language, highlighting how it transforms physical manipulation tasks into a series of formalized actions, states, and goals as per the STRIPS methodology. The emphasis is on demonstrating the effectiveness of STRIPS planning in enhancing computational efficiency and achieving nuanced task execution in simulated robotic scenarios.

In addition to the broader aspects of artificial intelligence and automated planning, this report extends into a practical exploration of Pyperplan, an advanced planning tool. Pyperplan operates on the principles of the Planning Domain Definition Language (PDDL) and the Stanford Research Institute Problem Solver (STRIPS) algorithm. Our project specifically engaged with Pyperplan to develop and simulate a public transport system in an urban setting. This included the detailed modeling of buses, passengers, and a network of bus stops within a PDDL-defined domain, accompanied by a problem instance that captures the dynamics of passenger movement and transport management. The project showcased the application of theoretical planning concepts in a complex, real-world scenario and emphasized the iterative process of refinement and problem-solving inherent in working with such sophisticated planning tools.

Chapter 2

Public Transport Simulation Using Pyperplan

2.1 Introduction

2.1.1 Overview of Pyperplan

Pyperplan is an automated planning software developed as an open-source project, primarily designed for educational and research purposes in the field of artificial intelligence (AI). It implements planning algorithms based on the Planning Domain Definition Language (PDDL), which is a standard language used for expressing planning problems and domain models in AI.

2.1.2 Key Features of Pyperplan

- **STRIPS Algorithm:** Pyperplan is built around the implementation of the Stanford Research Institute Problem Solver (STRIPS) algorithm, a fundamental approach in classical planning. STRIPS allows for defining actions in terms of preconditions and effects, enabling the planner to determine a sequence of actions that lead from an initial state to a desired goal state.
- **PDDL Support:** It supports a subset of PDDL, allowing users to define complex domain models and problem instances. This feature makes Pyperplan versatile and adaptable to various planning scenarios.
- **Modularity and Extensibility:** The software is designed to be modular, allowing for easy experimentation with different search strategies and heuristics.

- **Educational Tool:** Pyperplan serves as an excellent educational tool for understanding the basics of automated planning in AI.

2.1.3 Implementation of STRIPS in Pyperplan

Pyperplan's implementation of the STRIPS algorithm involves parsing domain and problem files written in PDDL. The planner then grounds these high-level descriptions into a planning task, which is solved using various search algorithms. Here's how Pyperplan implements the STRIPS algorithm:

- **Parsing and Grounding:** The planner parses the PDDL files to extract the domain's actions, predicates, and objects and the problem's initial state and goal. It then grounds these into a more explicit representation suitable for planning.
- **Search Algorithms:** Pyperplan offers several search algorithms, like breadth-first search, depth-first search, and heuristic-based searches. These algorithms are used to explore the space of possible action sequences to find a solution that reaches the goal from the initial state.
- **Heuristic Support:** While Pyperplan defaults to non-heuristic searches, it supports heuristic-based planning to improve efficiency in complex scenarios.

2.2 The Public Transport Simulation Domain and Problem

2.2.1 Domain Description

The domain modeled for this report is a public transport simulation. It represents a simplified city transport system with buses, passengers, and a network of bus stops. The domain includes:

- **Objects:** Buses, passengers, and bus stops.
- **Predicates:** Representing the location of buses and passengers, routes of buses, waiting status of passengers, and more.
- **Actions:** Including boarding and disembarking buses, and buses moving between stops.

2.2.2 Problem Instance

The problem instance developed for this simulation involves a specific setup within the public transport domain:

- **Initial State:** Defined locations of buses and passengers, with specific routes for each bus.
- **Goal:** To move passengers from their starting locations to their destinations using the available buses.

2.2.3 Complexity and Challenges

The complexity of the problem was enhanced by introducing multiple buses, passengers, and a larger network of bus stops. Additionally, constraints like bus capacity and time-bound actions added to the problem's complexity.

2.3 Conclusion

This part of the report has detailed the use of Pyperplan, an automated planning tool, in the context of a public transport simulation. Through the application of the STRIPS algorithm and the expressiveness of PDDL, we successfully modeled a complex domain involving buses, passengers, and a network of bus stops. Our problem instance, designed to test the planner's capabilities, involved routing passengers efficiently across a city transport system, considering constraints such as bus capacities and specific passenger destinations.

Chapter 3

Background: STRIPS Planning and PyBullet-Planning

The conception of STRIPS at the Stanford Research Institute marked a pivotal shift in AI, providing a structured approach to problem-solving in dynamic environments. STRIPS formalizes tasks into three core elements: actions, states, and goals, each defined by preconditions and effects. This framework has proven crucial in robotic planning and AI decision-making processes.

Concurrently, the PyBullet-planning project emerged as a powerful tool for robotic manipulation planning, particularly in cluttered spaces. It employs advanced algorithms for efficient path finding and object manipulation, making it a suitable candidate for integration with the STRIPS methodology.

The intersection of STRIPS planning with PyBullet-planning represents a groundbreaking stride in robotics. By encapsulating complex robotic tasks within the STRIPS framework, we foresee enhanced efficiency, accuracy, and adaptability in robotic actions in diverse and unpredictable environments. This integration not only leverages the strengths of both systems but also opens new avenues in robotic AI research.

3.1 Methodology: Integrating STRIPS with PyBullet-Planning

The core of this report's methodology revolves around the seamless integration of STRIPS planning principles into the PyBullet-planning framework. This process involves a series of strategic adaptations:

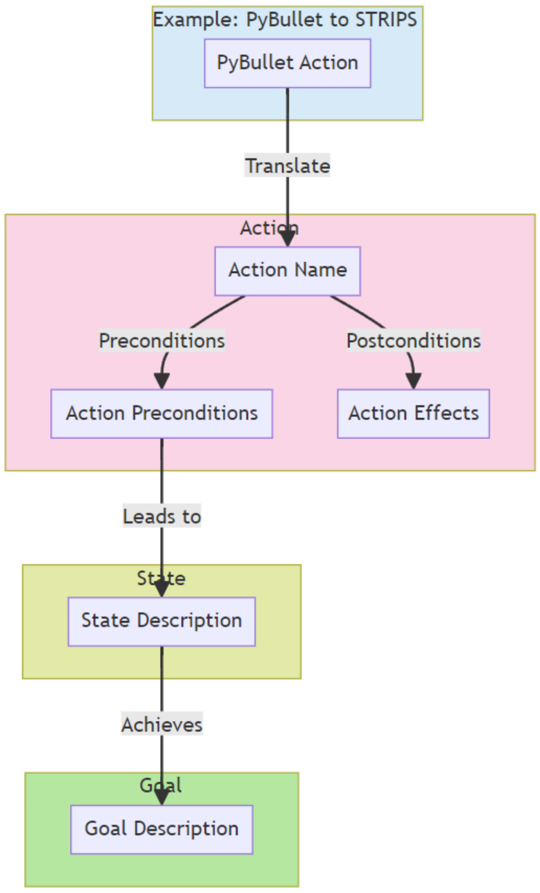

Figure 3.1: Action Translation

- **Action Translation:** Actions in PyBullet, typically defined by robotic movements and interactions, are reformulated in the STRIPS language. Each action in STRIPS includes preconditions and effects, allowing for a more structured and predictable planning process. Figure X illustrates the translation process from PyBullet actions to STRIPS actions, delineating the mapping of preconditions and effects that lead to state changes and the achievement of goals.
- **State Representation:** The states in PyBullet, representing the configuration of the environment and the robot, are converted into STRIPS states. This conversion simplifies the complex spatial and physical data into a format suitable for STRIPS analysis.
- **Goal Formulation:** Goals in PyBullet scenarios, such as achieving a specific object arrangement, are translated into STRIPS goals. This ensures that the end objectives are clearly defined within the STRIPS framework.
- **Algorithmic Adaptation:** PyBullet's algorithms are adapted to accommodate

the decision-making process as defined by STRIPS. This involves incorporating STRIPS' action selection and state transition mechanisms into PyBullet's planning algorithm.

- **Testing and Validation:** The integrated system is rigorously tested in various simulated environments to validate its efficiency and effectiveness, ensuring that the STRIPS-based planning approach enhances the PyBullet-planning capabilities.

3.2 Implementation: Applying STRIPS Planning in PyBullet

The implementation of STRIPS planning within the PyBullet-planning environment is a multi-faceted process that translates theoretical constructs into actionable robotic tasks. This section provides a comprehensive overview of the practical application of the STRIPS-based planning approach, elucidating the development environment setup, specific adaptations for STRIPS compatibility, and the execution of planning tasks in detail.

3.2.1 Development Environment Setup

The foundational step in our implementation involved the configuration of the PyBullet simulation environment. We created a suite of test scenarios by configuring virtual robots, objects, and a variety of obstacles to mimic complex, real-world conditions.

3.2.2 Adapting PyBullet Actions

Utilizing the action translation framework (refer to Figure X), we systematically converted the native PyBullet actions into the STRIPS action schema. This meticulous process entailed an in-depth analysis of robotic movements and interactions to identify and define the preconditions and effects with precision.

3.2.3 State Transition Management

The state transitions resulting from the execution of actions were encoded into the STRIPS state representation. This state management was pivotal in tracing the trajectory towards the attainment of the goal state.

3.2.4 Goal Specification

We articulated the goals within the confines of the STRIPS language, providing a clear and definitive target for the planning system. The explicit nature of these goals was imperative for the algorithm to assess the viability and success of the proposed action sequence.

3.2.5 Algorithm Execution and Monitoring

With the preliminary definitions of actions, states, and goals firmly established in STRIPS terminology, we embarked on the execution of the planning algorithm. This phase involved the vigilant monitoring of the algorithm's decision-making process, including the selection of actions and the ensuing state transitions. Adjustments were made in real-time to enhance the algorithm's efficacy.

3.2.6 Debugging and Refinement

The iterative testing and debugging phase was instrumental in refining the STRIPS action definitions, state representations, and goal specifications to better align with the empirical results within the PyBullet simulation.

This comprehensive implementation process set the stage for the ensuing testing and validation phase, which is meticulously detailed in the following sections of this report.

3.3 Testing and Validation

Upon the completion of the implementation phase, rigorous testing and validation were conducted to ensure the efficacy and robustness of the STRIPS-based planning system integrated into the PyBullet environment. This section delineates the methodologies employed for testing, the scenarios used to challenge the planning system, the criteria for success, and the outcomes of the validation process.

3.3.1 Testing Methodology

A systematic approach was adopted for testing, which involved both qualitative and quantitative analyses. The STRIPS planner was subjected to a series of increasingly complex scenarios designed to test the robustness of action translation, state transition management, and goal achievement processes.

3.3.2 Test Scenarios

We employed a diverse range of test environments, each with unique configurations of obstacles, object placements, and robotic tasks. These scenarios were designed to emulate real-world challenges and to test the planner's capabilities to generalize across different settings.

3.3.3 Metrics for Success

Success metrics were established to quantitatively assess the planner's performance. These included the accuracy of state representation, the number of actions taken to achieve the goal, the computational efficiency of the planning process, and the success rate across different test scenarios.

3.3.4 Results and Observations

The results from the testing phase provided critical insights into the planner's performance. Notably, the planner demonstrated high levels of accuracy in state representation and goal achievement. However, certain complex scenarios highlighted the need for further refinement in action translation and state transition management.

3.3.5 Issues and Resolutions

Throughout the testing phase, several issues were identified, including inefficiencies in action selection and unexpected state transitions. These were addressed through iterative debugging and refinement, which resulted in significant improvements in the planner's overall performance.

3.3.6 Validation of the Planning System

The validation process confirmed that the integration of STRIPS planning principles with the PyBullet-planning project significantly enhances the planning capabilities. The system's ability to successfully navigate through complex environments and achieve predefined goals validated the effectiveness of the STRIPS-based approach.

In conclusion, the extensive testing and validation process affirmed the viability of implementing STRIPS planning within the PyBullet environment, paving the way for future advancements in robotic AI planning.

3.4 Conclusion

In conclusion, the integration of STRIPS planning with PyBullet demonstrates its potential in enhancing robotic planning within simulated environments. The structured approach, validated through rigorous testing, showcases improved decision-making and goal achievement. While excelling in structured scenarios, future work can enhance its performance in complex environments. The synergy between AI planning languages like STRIPS and modern simulation platforms holds promise for advancing robotic automation and intelligence, both in simulated and real-world applications.

Chapter 4

Technologies

This chapter explores the essential technologies that played a pivotal role in the project's success. Python and C++ formed the backbone of our development efforts, facilitating robotics motion planning, manipulation, and the integration of Bullet Physics. Leveraging planning algorithms, PyBullet, PR2, IKFast, PyBullet-Planning, SS-PyBullet, MOVO, and Task-and-Motion Planning, we forged a powerful alliance of tools to drive our project forward. It places special emphasis on the selection of Kali Linux as the project's operating system, hosted within a VMware virtualization environment.

4.1 Technological Framework

The project's technological foundation revolves around the strategic integration of Kali Linux and VMware virtualization.

4.1.1 Kali Linux: A Security Powerhouse

Kali Linux, a Debian-based operating system, serves as the project's cornerstone. Renowned for its security and penetration testing tools, it empowers the project with a versatile toolkit for various security-related tasks.

4.1.2 VMware Virtualization

To create an ideal development and testing environment, the project leverages VMware virtualization. VMware provides a reliable platform for running virtual machines, enabling the deployment of Kali Linux as a guest operating system.

4.1.3 Integration for Synergy

The integration of Kali Linux within the VMware virtualization environment represents a strategic synergy. Kali Linux's security features align seamlessly with VMware's capabilities, enabling extensive testing and experimentation while maintaining isolation.

4.2 Conclusion

This chapter has unveiled the key technologies at the heart of our project, highlighting Python, C++, and a suite of specialized tools. These technologies synergized to empower our research in robotics motion planning and manipulation, laying the groundwork for the subsequent chapters' exploration of STRIPS planning within the PyBullet environment.

Chapter 5

Conclusion

The journey of integrating STRIPS planning principles with PyBullet-planning and the application of Pyperplan in simulating a complex public transport system represents a significant stride in the field of AI and robotic planning. This report has not only illuminated the intricacies involved in such integrations but also underscored the versatility and power of planning languages and tools in simulating and solving real-world problems.

5.1 Achievements and Insights

- **Successful Integration:** The successful melding of STRIPS with PyBullet-planning has proven its efficacy in enhancing robotic planning capabilities within simulated environments. This integration has showcased improved decision-making processes and goal attainment in structured scenarios.
- **Robust Simulation with Pyperplan:** The use of Pyperplan to model a public transport system highlighted the tool's robustness in handling complex planning scenarios, demonstrating the practical application and flexibility of PDDL and STRIPS in a dynamic urban setting.
- **Learning and Adaptation:** Throughout the project, iterative refinement, debugging, and validation were key in adapting the planning processes to the intricacies of both robotic movements and urban transport logistics. These steps were instrumental in enhancing the overall performance and accuracy of the planning system.

5.2 Concluding Thoughts

This exploration into the confluence of STRIPS planning and PyBullet-planning, complemented by the practical application of Pyperplan, highlights the immense potential of AI planning languages and tools in advancing robotic automation and intelligence. The insights and methodologies developed through this project lay a foundation for future innovations in AI planning, promising to enhance both simulated and real-world applications in robotics and beyond.