Project .....

Artificial Intelligence in the Computer Game Astro Kid

Kasper Reindahl Rasmussen (s103476)

September 20, 2015

Contents

Glossary

Astro Kid 2D flash puzzle game, where the avatar moves to the exit . 1

IPC International Planning Competition. 1

PDDL Planning Domain Definition Language, used for the IPC. 2

Astro Kid

1 Introduction/Project Description

The project concerns Artificial intelligence used for the flash game Astro Kid. This game is a puzzle game, where a Avatar has to move to a designated goal field. The Astro Kid World varies greatly in complexity depending on the size and number of objects, but generally with a slowly growing complexity as it progress trough the levels (due to more and new types of objects being add). One of the challenges with the Astro Kid world is that several actions, have consequences that have continuous effect on the world, and how to represent

these effects and interact with them. At the moment no AI exist that can interact with this world.

The main goal of this project is to analyse the Astro Kid world, and create an AI that can interact and solve problems in this world effectively. To achieve this there will be looked at different approaches for an AI to solve the problem. The first approach looked at will be how far one can get with a generic PDDL planner, such as Fast-Downward. This means that there also have to be looked at how and how much of the domain can be described in PDDL. There will also be looked at creating a more purpose build planner for the domain. The planner should be able to handle all types of actions and effects used in the Astro Kid World. To achieve this partial order planning could be used as an overall approach. The developed system should in general be generic enough, to make it easily applicable on similar domains.

The planner needs to be able to execute its plans on the domain. This will initially be done through a implementation, that simulates the Astro Kid world. A secondary goal will here be an implementation, that allow direct interaction with the actual flash game.

The Astro Kid game is designed with as slowly growing complexity or learning curve, as it progress trough the levels, it could therefore be interesting, to look at letting the AI learn, as it progress through the game as a secondary goal. To let the AI learn, would be in regards to learning how the different actions effects the world, and use that knowledge for its planing. This could be achieved by letting the AI learn with the aid of a teacher, inspired by techniques such as those used in [Walsh and Littman, 2008] and [Benson, 1996].

The success criteria for this project is to fulfil the following primary goals. The secondary goals would be good to reach, but the success of the project does not rely on it.

Primary goals

- Describe what can be described of the domain using PDDL.
- Apply a generic PDDL planner on the domain .
- Creating a more purpose build planner for the domain.
- Implement a system that simulates the Astro Kid world, and allows execution of plans.

Secondary goals

- Implement direct interaction with the Astro Kid game.
- Learning actions schemas.

2 General planning

why use a general planner

the idea of using a general planner is that its possible to solve the problem efficiently without building a planner from scratch for each problem. a general planer will however nearly always be less efficient than an planner for the specific problem. This is due to....

The main question is therefore if the general planner can solve the problem efficiently enough to be useful.

To begin solving the problem with a general planner, it needs to be described in a Planning languages. in this case PDDL is suitable choice since it is a expressive language and used for the IPC, and there therefore exists a series of planners that support the language.

single agent planing

complete pddl not fully supported by most planners

the general planner there will be used is fast downward due to it being .... open source .... and has done well in several IPC and must be considered one of the better planer out there, even though it does support a subset of the complete PDDL.

2.1 PDDL

since the chosen planer is fast downward which is limited to PDDL 2.2 level 1 + action cost, this version of pddl will be used

2.1.1 The Astro kid world

the astro kid world is a world where every thing moves in a 2d grid, each point in the world is represented in pddl as an object and its location is defined relative to its neighbours.

when looking at the astro kid world, its worth noting the different features.

worth of note is that even though its a single agent world, multiple things can happen concurrent. this i mainly due to that some actions have continuous effects (f.x. robots walking, sliding and falling).

pddl doesn't directly allow concurrent actions, unless they are changed into a single action. this could be done by having a action for each possible combination of objects moving, This approach would however leads to an explosion of possible actions since the number of actions would then be depend on the number of objects that can have a continuous effect. the scaling would therefore be terrible.

This approach might work if working under the assumption that the number of objects in the problem is fairly limited.

there are basically two types of actions in the domain, the ones where the planner has a choice (player action) and actions which are consequences/updates of earlier actions.

the main idea is therefore to separate the domain into input/choice and update, and then enforcing a ordering of the actions so a input/choice is always followed by an update

the ordering of actions can be enforced by using flags

this can be solved by ordering the actions in such a way

limit the number of actions

continuous actions are handled by locking a objects to a given course of actions by creating a predicate with it.

by checking for the predicate before anything happens to the object.

the update step is in itself split up in to parts with a strict ordering to enforce that the concurrent actions interact correctly with each others.

problem areas

Table 1: problem 4

	forall	simple
total Time	26.175s	136.157s
search Time		
instantiation Time	20.770s	1.260s
Translator axioms	1	132652
Translator peak memory	100568 KB	208356 KB
Translator task size	42751	573269

concurrent actions continuous actions effects

2.2 Fast downward

there is various ways of tweaking the performance of the planner. The two most obvious ways of doing this is changing the parameters of the planner and changing the PDDL domain to fit the planners strengths better.

2.2.1 Domain

Test have shown that use of universal quantifiers have a large impact on the running time.

when running the different versions of the domain

one thing becomes clear, the results varies greatly depending on the domain and level combination.

when looking at the results what shows is that the use of universal quantifiers greatly increases the instantiation time.

this is where the grounding of the atoms takes place.

replacing the universal quantifiers with a combination of existential quantifiers and a new action ensures a fast quicker instantiation, but have the cost that the number of axioms explodes, which in the end can greatly hamper the actual search.

the problem grows with the number of movable objects, due to all the quantifiers mainly interacts with them. unless as shown in prob4v2 where planner figures out that the objects is in fact not movable

its worth noting that in general fast downwards heuristics dosnt handle axioms well this could explain why the search fails so miserably.

the results shows that unless the problem is t

another way of tweaking the code is to optimize the Problem Definition, this can be done by removing unreachable states/ objects, more precise remove the representation of position that isnt usefull.

2.2.2 Parameters

the main parameter is the choice of heuristic. this choice is fairly limited

References

[Benson, 1996] Benson, S. S. (1996). Learning action models for reactive autonomous agents.

[Walsh and Littman, 2008] Walsh, T. J. and Littman, M. L. (2008). Efficient learning of action schemas and web-service descriptions. *AAAI Conference on Artificial Intelligence*, 23.