Autonomous Building of Structures in Unstructured Environments via AI Planning

Jamie O. Roberts, Santiago Franco, Adam A. Stokes, Sara Bernardini
{jamie.roberts, adam.stokes}@ed.ac.uk, {santiago.francoaixela, sara.bernardini}@rhul.ac.uk

Motivation

Robotic System: To facilitate the self-assembling of the Connect-R robotic system that provides the increased capability in Nuclear Environments.

Task Planning Domain:

- Capable of planning for 3D problems
- Capable of capturing the complexity of the robot
- Flexible goal language
- Human interpretable and explainable domain
- Scalable and accurate solution

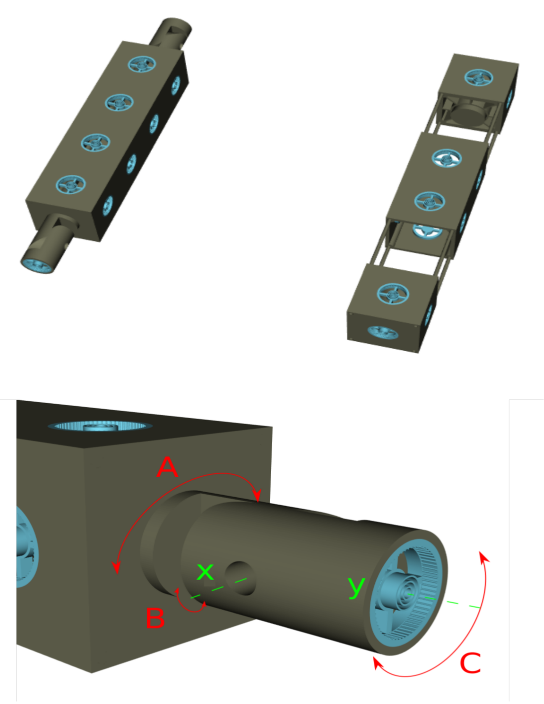
Providing Structure in Unstructured Environments

Features

- 3D: Task Planning domain reasons about an environment space of x,y and z dimensions.
- Reasoning around the Robot: The domain reasons about the most fundamental resolution of the individual robot itself.
- Flexible Goal Language: Target structures can be defined through a mixture of singular locations and entire robots of specific orientations.
- User-friendly domain for human-system interactions

Robot Design

Each robot is a symmetrical cuboid in its basic nature. Each side of the robot has 4 connection points along an equal spacing. The robot can extend it's end nodes so that it can perform a 'walking' like motion. The robot also has two manipulators in each end, which has 3 DOF's of freedom. These are used to manipulate the robot and/or other robots.



Domain Objects and Predicates

Objects:

A point (x, y, z) in space is represented by three objects of type 'coord' and this object can be seen in the domain syntax in the following examples (?x ?y ?z).

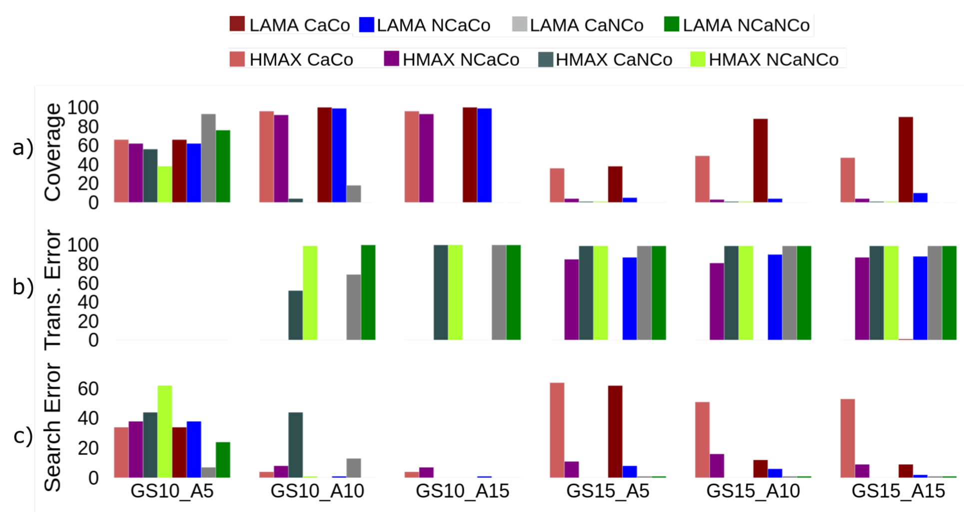
An object of type 'angle' has 3 constants: 0ang, 90ang and 450ang. This object can be seen in the domain synatx as ?ang.

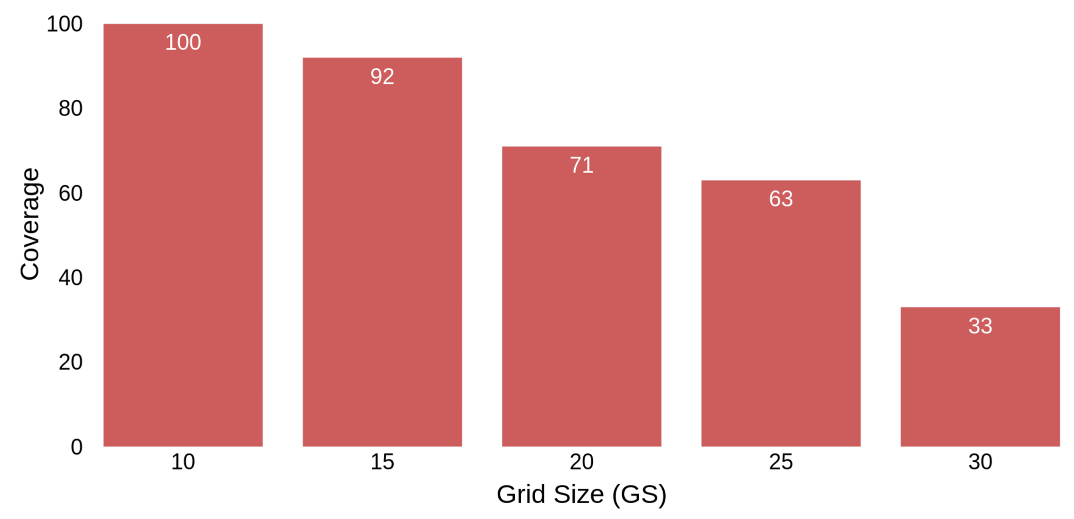
An object of type 'countable' is used to account for the number of robots in the system.

Predicates:

- *at_robot(?x ?y ?z ?ang)*
- *dyn_at(?x ?y ?z)*
- *obstacle_at(?x ?y ?z)*
- *supported(?x ?y ?z)*
- *is_outside(?c)*
- *at_aperture (?x ?y ?x ?ang)*

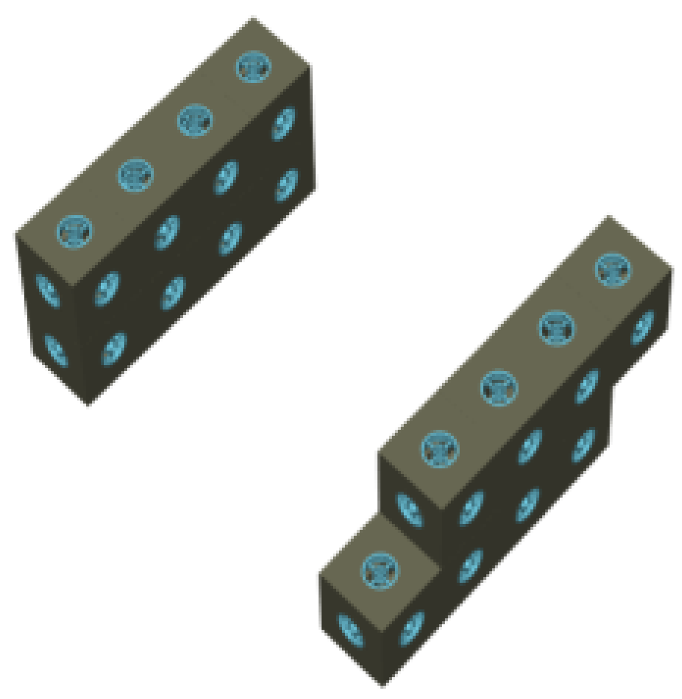
Overall Results





Example Action

```
(action traverse_plus_x)
:parameters (
  ?x1 ?y1 ?z1 ?x2 ?y2 ?z2 ?p1 ?p2
  ?p3 ?p4 ?p5 ?p6 ?un_z1 - coord)
  :precondition (and
    (plus_one ?un_z1 ?z1)
    (at_strut ?x1 ?y1 ?z1 ang0)
    (= ?y2 ?y1)
    (= ?z2 ?z1)
    (plus_one ?x1 ?x2)
    (plus_one ?x1 ?p1)
    (plus_one ?p1 ?p2)
    (plus_one ?p2 ?p3)
    (plus_one ?x2 ?p4)
    (plus_one ?p4 ?p5)
    (plus_one ?p5 ?p6)
    (dyn_at ?x2 ?y2 ?un_z1)
    (dyn_at ?p3 ?y2 ?un_z1)
    (not (dyn_at ?p6 ?y2 ?z2))
    (grounded ?p6 ?y2 ?un_z1)
    (dyn_at ?p6 ?y2 ?un_z1)
    )
  :effect (and
    (not (at_strut ?x1 ?y1 ?z1 ang0))
    (at_strut ?x2 ?y2 ?z2 ang0)
    (not (dyn_at ?x1 ?y1 ?z1))
    (not (grounded ?x1 ?y1 ?z1))
    (dyn_at ?p6 ?y2 ?z2)
    (grounded ?p6 ?y2 ?z2)
     )
)
```

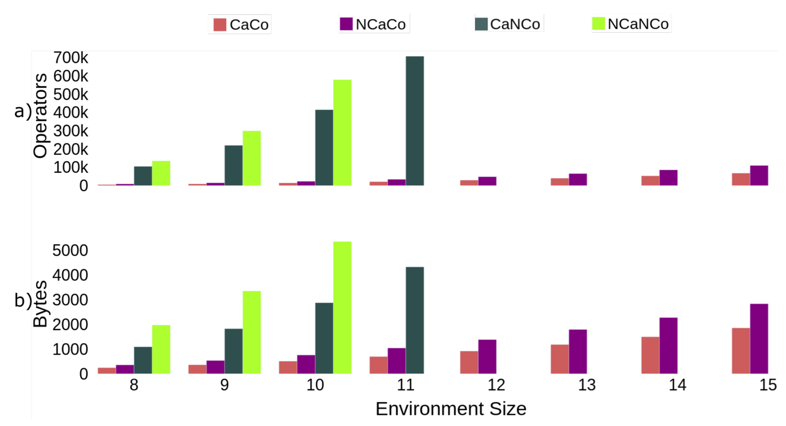


Experimental Conditions

This work evaluates the effectiveness of four different domain representations which are defined as follows:

- The Canonical domain contains the countable robot representation and the canonical orientations, referred to as **CaCo**.
- The Non-Canonical domain is the domain representation that contains only the countable robot representation, referred to as **NCaCo**.
- The Agent-Canonical domain contains only the canonical orientation, referred to as **CaNCo**.
- The Agent-Non-Canonical domain does not contain any of the contributions of the canonical representation, referred to as **NCaNCo**.

Domain Complexity



$$M_{CaCo} = 2*GS^3+3*(GS^2*(GS-3))+GS+1 \quad (1a)$$

$$M_{NCaCo} = 2*GS^3+\mathbf{6}*(GS^2*(GS-3))+GS+1 \quad (1b)$$

$$M_{CaNCo} = 2*GS^3+3*(GS^{\mathbf{3}}*(GS-3))+GS+1 \quad (1c)$$

$$M_{NCaNCo} = 2*GS^3+\mathbf{6}*(GS^{\mathbf{3}}*(GS-3))+GS+1 \quad (1d)$$

Actual Prototype

Conclusions

We showed that 3D planning for a Multi-Robot system is achievable by:

- Discretizing the Domain with high-level actions.
- Using a Canonical representation where only 3 orientations allowed: Plus-X,Plus-Y,Plus-Z.
- Using a Countable agent representation which lowers the domain complexity by an order of magnitude approximately.
- Finally we showed that the best PDDL representation (CaCo) is scalable to practical environment sizes directly writing FDR representations.