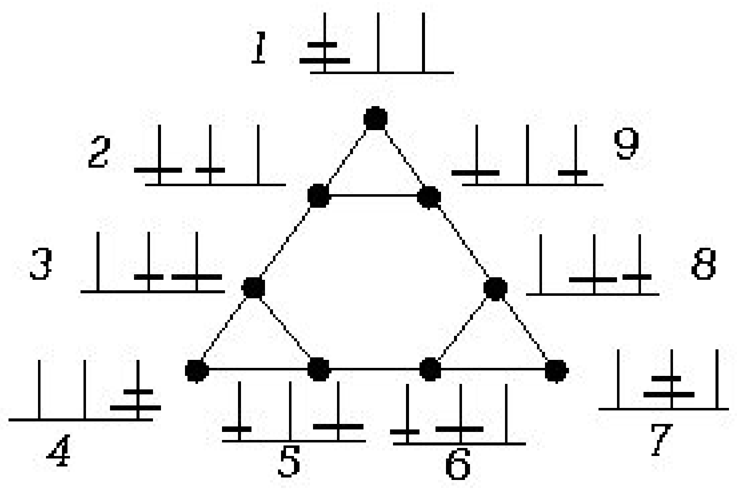
Using Soar with a Small Mobile Robot

Paul Benjamin

Pace University

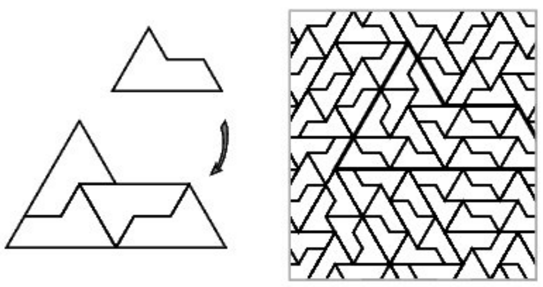
Looking for Good Representations

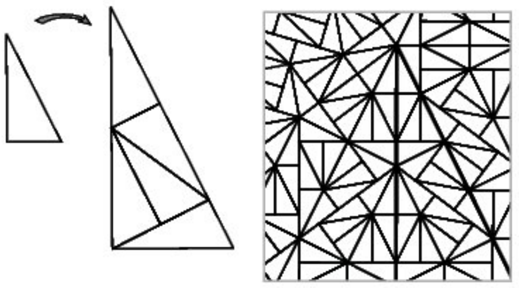


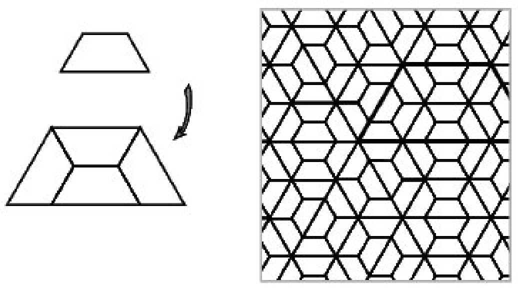
Good Formulation: Move the small disk from peg i to peg j
Move the large disk from peg i to peg j

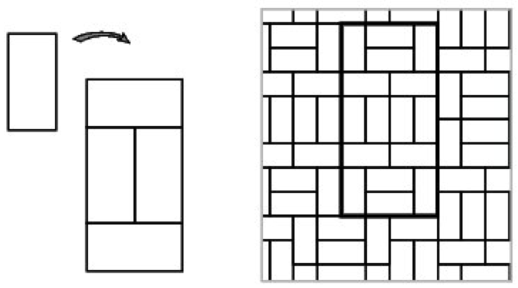
Not-so-good Formulation: Move the top disk from peg i to peg j

Substitution Tilings

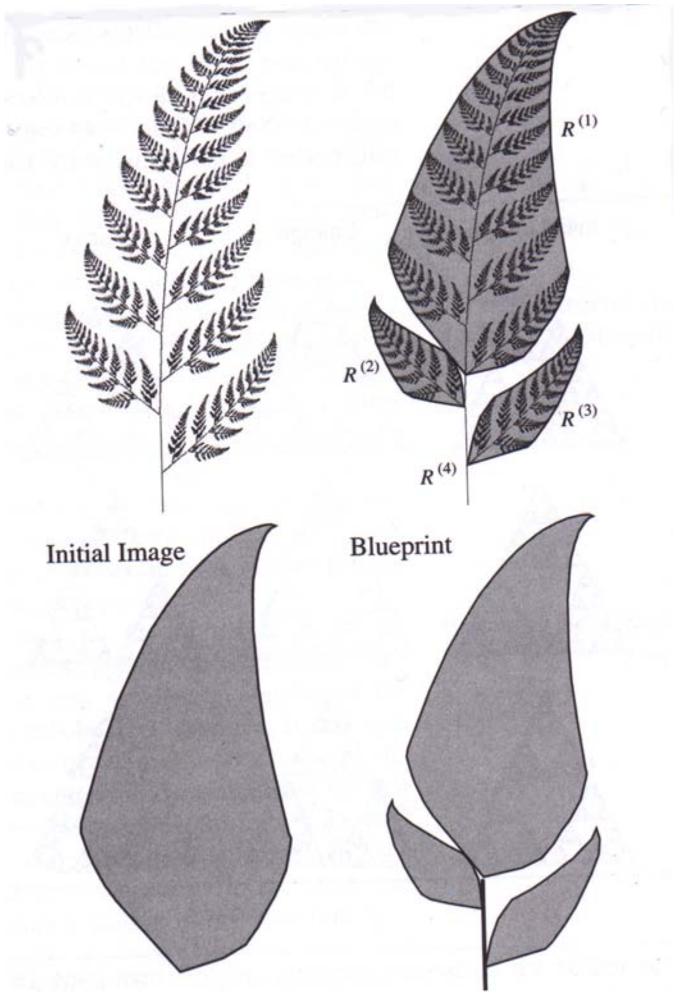




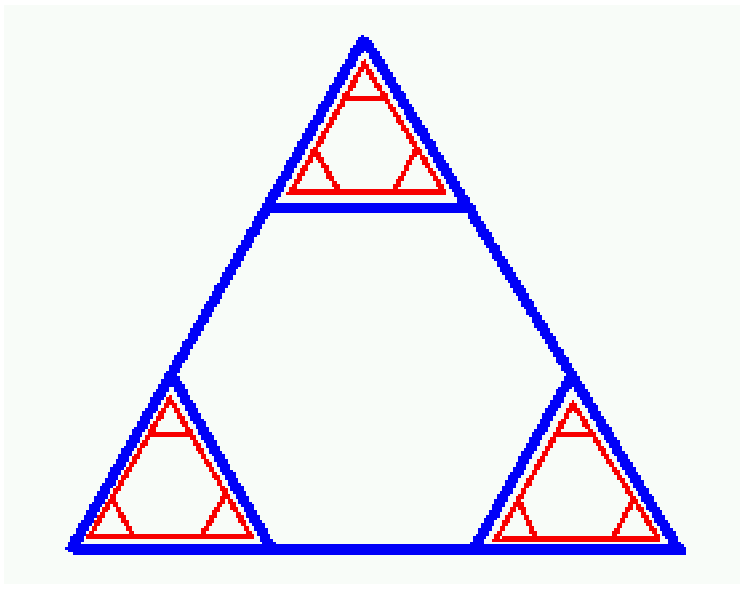




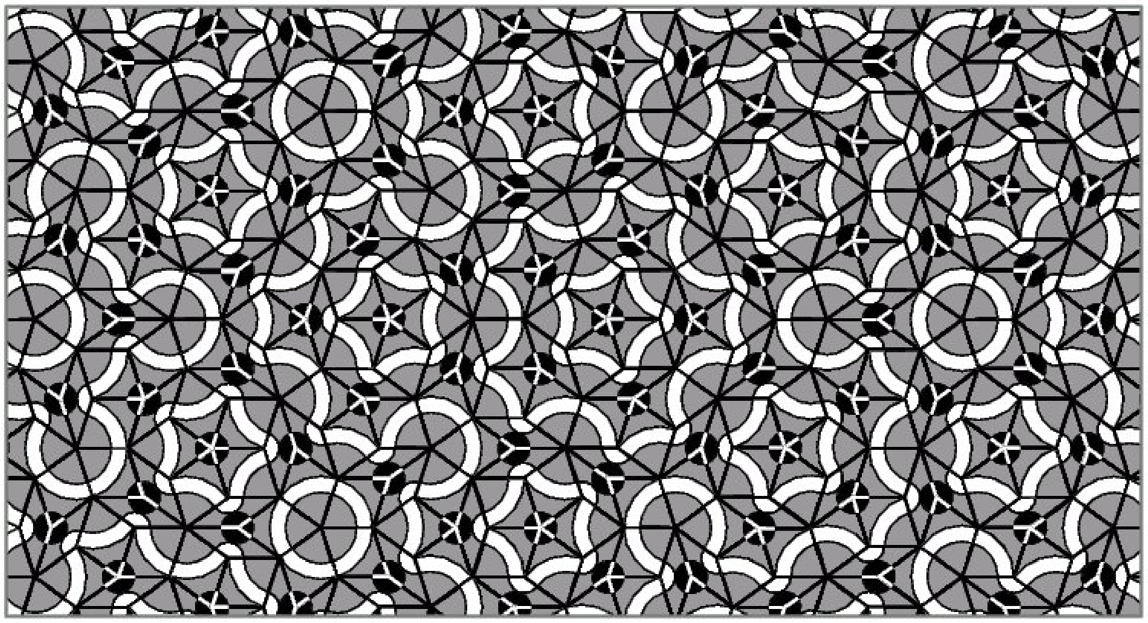
Growing a Fern



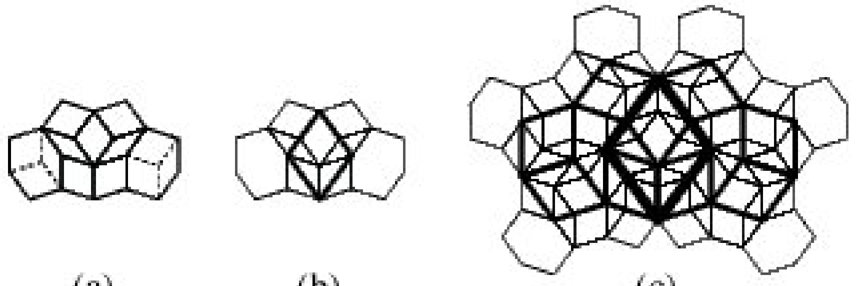
Obvious Substitution Tiling in State Space



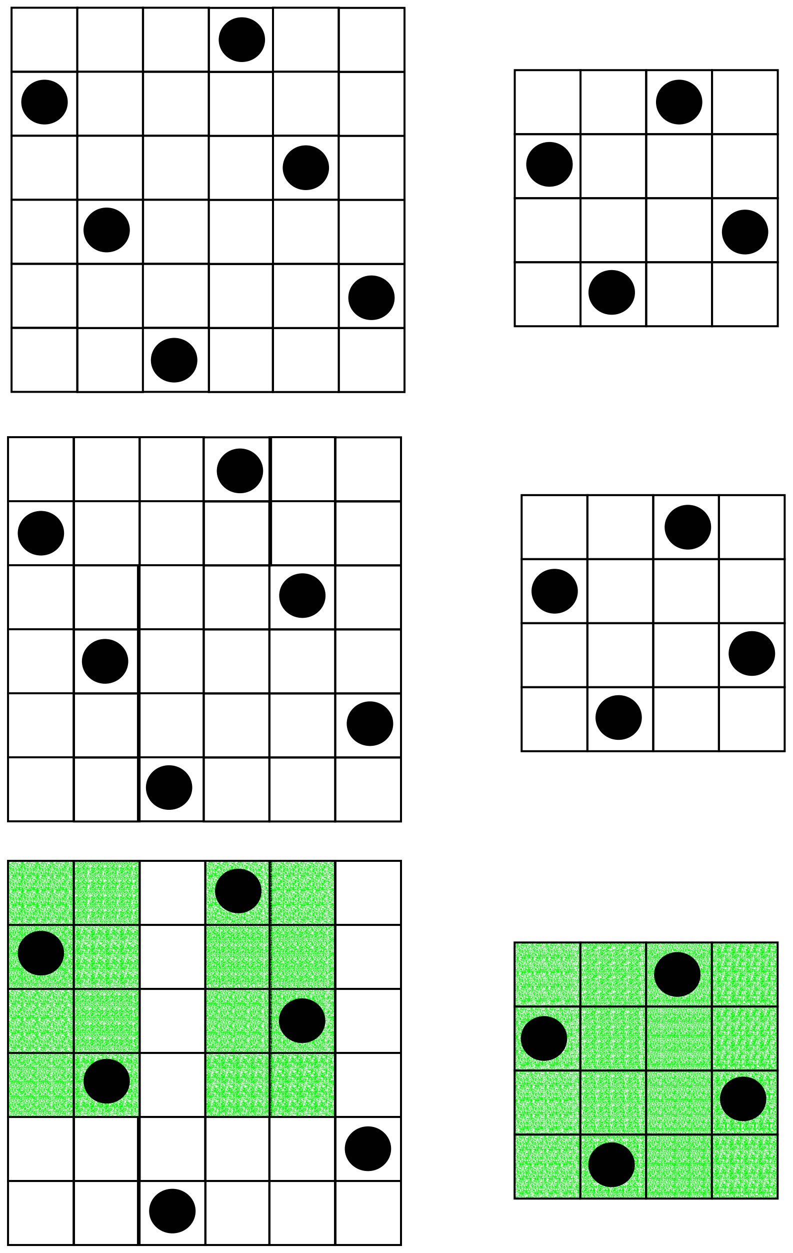
Penrose Tiling



Not-so-obvious Substitution Tiling


(a) (b) (c)

The N Queens has a Substitution Tiling Structure



Reason for using Soar

Many hard real problems have a substitution tiling structure, so they are isomorphic across many scales. For a problem solving architecture to exploit this structure to solve the problem using local search, it must have a computational self-similarity mechanism.

Universal subgoaling is the only such mechanism I know of.

The Robot

Pioneer 2 DXE
Stereo color vision/pan-tilt
16 Sonars
Microphone
Speakers
Bump sensors
Wireless Ethernet

Two Requirements of Robot Programming

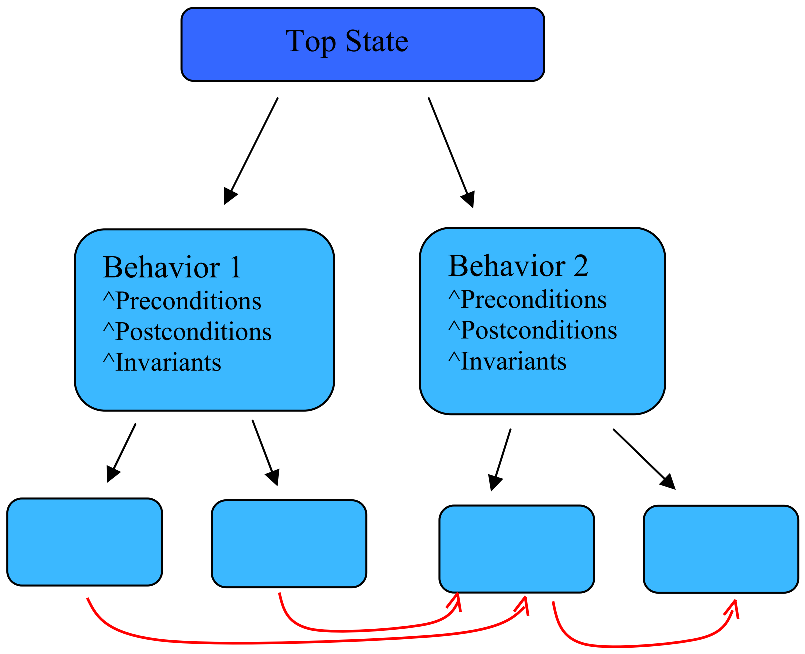
The world is continuous and always “on”.

Truly concurrent actions are required.

Approach:

Use a declarative representation of the operators.

A Declarative Copy of Soar in Soar



The behaviors are persistent operators, often abstract. They are organized in a partial order.

Invariants are productions that monitor the boundary conditions of the behaviors.

Task-independent operators organize the behaviors.

The behaviors are declarative and can be chunked into productions. This requires operators that know how to match conditions. Using selection.soar we can go the opposite direction, elaborating a state space and actions and attaching them to the top state.

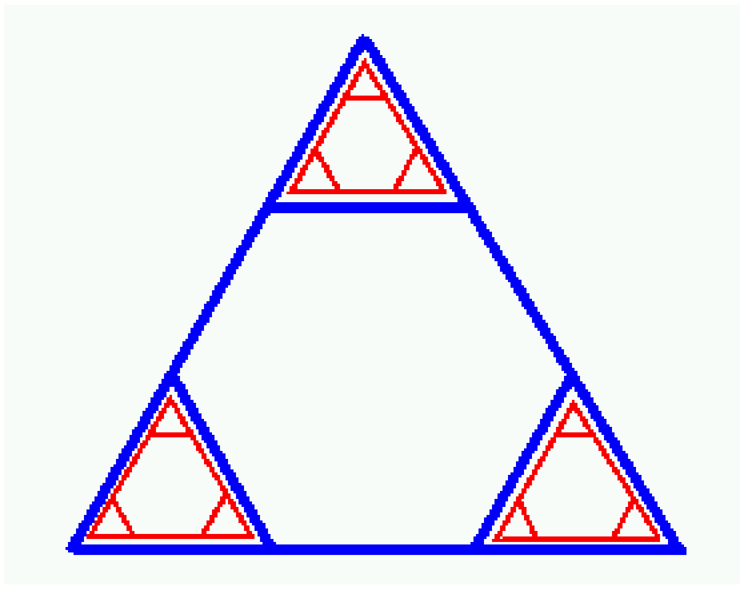
Finding Substitution Tilings in Soar

Use selection.soar to elaborate the state space and actions and attach them to the top state.

Create subgoals to find actions that hold each feature invariant while changing other features. This causes search and finds chunks that decompose the state space into subspaces that change individual features.

Create a subgoal to find isomorphisms between these subspaces, and check each one to see if the isomorphism creates a substitution tiling (it needs to cover the original state space.)

Once a tiling has been found, solve larger problems by solving only the inflated copy of the state space, then subgoaling to solve the next smaller subspace, etc.



Summary

Universal subgoaling is perfectly suited to exploit self-similar structure in problem solving.

The goal of this project is to investigate the relationship between the self-similar structures present in the perceptions and actions of a mobile robot.

This requires easy transformations between procedural and declarative representations of operators.