Quiz 1

- Monday Sept 21. Open for 24 hours, from 12 midnight (am) to 11:59 pm (Toronto time).
- You can only try it once.
- All questions can be viewed at once.
- Correct answers will be shown after everyone has had a chance to view the quiz.
- Quiz questions will be randomized.

- You will have 4 Hours to do the quiz. Should take about 40 mins!! So you will have time to think about the answers, go back to the lecture material, etc.
- To do the quiz effectively make sure you look at all of the required videos, use piazza and my office hours to get your questions answered.

Office Hours

- Wednesdays 1-2pm. Via Zoom. I will post a Zoom link on the course web site.
- By special arrangement if needed (email me).
- If you wish to ask questions relevant to this weeks quiz I will take questions after the tutorial.

Logistics World

- Set of Cities
- For each City a set of locations in that city.
- Some locations are Airports.
- Set of Trucks, each truck is in some city.
- Set of Airplanes
- Trucks can move between any location in the same city.
- Airplanes can move between any two airport.
- Set of packages each in some city at some location.
- Packages can be loaded into a truck or airplane if that vehicle is at the same location at the package.
- If a package is in a vehicle it is moved when the vehicle is moved.

Logistics World

- Aim is to pickup a bunch of packages and deliver them to some goal locations.

Logistics World

- Such search problems are commonly represent by
  - a **static** objects—the objects have types and are unchanged by the actions. There might also be some static facts.
  - A set of **predicates** that we use to assert facts about the objects
  - Using the objects and the predicates, the **states** are represented as a set of facts.
  - Actions have preconditions. They are only applicable to states whose set of facts satisfy the preconditions.
    - Actions transition between states by adding and deleting facts from the state to generate a new state
- This kind of representation was originally called a STRIPS representation, now with various extensions it is called PDDL—planning domain description language.

Logistics World

Lets specify the Logistics World search space using a STRIPS/PDDL.

1. Typed objects:
   t1, t2 – truck
   p1, p2 – package
   a1 – airplane
   l1, l2, l3, l4 – location

   Static facts:
   sameCity(l1,l2), sameCity(l3,l4)
   airport(l2), airport(l4)

Logistics World

States:

a set of facts involving the predicates
at(x? – truck, airplane, package, y? – location)
in(x? – package, y? – truck, airplane)

x? and y? are variables. These can be substituted by any object of that type.

Example:

Initial state I = {at(t1, l1), at(t2, l3), at(a1, l2), at(p1, l1), in(p2, t1)}

Goal state G = {at(p1, l4), at(p2, l4)}

the goal is satisfied by any state containing these facts.

Logistics World

Actions

1. move_truck(t? – truck, l1? – loc, l2? – loc)
   #move truck t? from location l1? to location l2?
   precondition: at(t?, l1?), sameCity(l1?, l2?), l1? != l2?
   add: at(t?, l2?)
   del: at(t?, l1?)
2. move_plane(a? – airplane, l1? – loc, l2? – loc)

   #move airplane a? from location l1? to location l2?
   precondition: at(?a, l1?), airport(l2?)
   add: at(a?, l2?)
   del: at(a?, l1?)

Logistics World

Actions

3. load_vehicle (p? – package, v? – truck, plane, l? – location)
   #load a package into a vehicle at location
   precondition: at(p?, l?), at(v?, l?)
   add: in(p?, v?)
   del: at(p?, l?)
4. unload_vehicle(p? – package, v? – truck, plane, l? – location)
   #unload a package from a vehicle at location
   precondition: at(v?, l?), in(p?, v?)
   add: at(p?, l?)
   del: in(p?, v?)

Example Transitions:

I = {at(t1, l1), at(t2, l3), at(a1, l2), at(p1, l1), in(p2, t1)}

➔ load_vehicle(p1, t1, l1)
{at(t1, l1), at(t2, l3), at(a1, l2), in(p1, t1), in(p2, t1)}

➔move_truck(t1, l1, l2)
{at(t1, l2), at(t2, l3), at(a1, l2), in(p1, t1), in(p2, t1)}

➔unload_vechicle(p1,t1,l2) ➔ unload_vechicle(p2,t1,l2) ➔
load_vechile(p1, a1, l2) ➔ load_vechile(p2, a1, l2) ➔
move_plane(a1, l2, l4) ➔ unload_vechicle(p1,a1,l4) ➔
unload_vechicle(p2, a1, l4) ➔
{at(t1, l2), at(t2,l3), at(a1,l4), at(p1, l4), at(p2, l4)}

This state satisfies the goal {at(p1, l3), at(p2, l3)

PDDL

- Open source high performance search engines (planners) that solve search problems expressed in PDDL exist. http://www.fast-downward.org/

- These planners also use powerful heuristic that only need to examine the PDDL description—they don't have to be constructed for each different search space.

- The search problems we will use in class are simpler and usually can be represented in custom ways (rather than using general PDDL). E.g., we can represent 8-puzzle with a list of numbers representing the tile at each location.