

16.410-13: Principles of Automated Reasoning and Decision Making

Midterm

October 20th, 2010

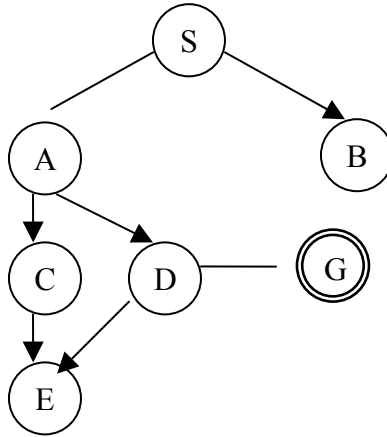
Name	
E-mail	

Note: Budget your time wisely. Some parts of this quiz could take you much longer than others. Move on if you are stuck and return to the problem later.

Problem Number	Max	Score	Grader
Problem 1	30		
Problem 2	20		
Problem 3	35		
Problem 4	35		
Total	120		

Problem 1 Uninformed Search (30 points)

In this problem we exercise your understanding of the uninformed search algorithms introduced in class – depth-first, breadth-first and iterative-deepening. We start by exploring the task of finding a path from state **S** to state **G** in the following directed graph using uninformed search:



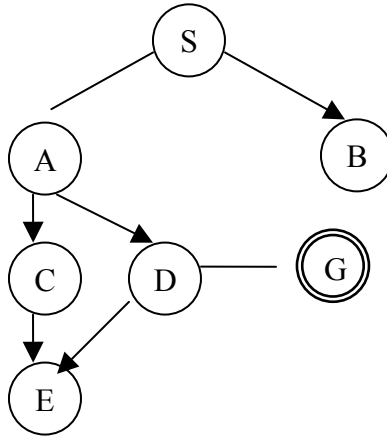
Make the following assumptions:

- Depth and breadth-first search explore a node's children in **alphabetical order**.
- The depth-limited search called by iterative deepening search also explores a node's children in **alphabetical order**.
- Each search algorithm **uses a visited list** to prune nodes, **unless** otherwise stated.
- The search stops as soon as the goal is **expanded** (not when it is first visited).

In the following, recall that a node is **expanded** when the search algorithm takes a path to that node off the queue, and attempts to create its children. A node is **visited** when a path to that node is first placed on the queue.

Part 1.A Depth-first Search with a Visited List (12 points)

The above graph is repeated here for your convenience:



For each iteration of depth-first search, fill in the path being expanded, the state in the search queue and the visited list after expansion. Search begins with the path (S) on the queue and node S on the visited list. Remember to **use a visited list**. Stop if search does not terminate after 6 steps.

	Path being expanded	Queue after expansion	Visited list after expansion
1		(S)	S
2			
3			
4			
5			
6			

Part 1.B Search Issues (6 Points)

In each of the following, circle T if the statement is True, and F, otherwise:

- a. T F Assuming an edge cost of 1, breadth-first search **with a visited list** is guaranteed to find a shortest path to the goal if one exists.
- b. T F Assuming an edge cost of 1, depth-first search **with a visited list** is guaranteed to find a shortest path to the goal if one exists.
- c. T F Breadth-first search **without a visited list** is guaranteed to find a path to the goal if one exists.
- d. T F Depth-first search **without a visited list** is guaranteed to find a path to the goal if one exists.

The following are all statements about Iterative Deepening search. For each statement, circle T if the statement is True, and F, otherwise.

Iterative Deepening Search...

- e. T F will typically require much less memory than breadth-first search.
- f. T F has a smaller worst case run-time overhead for search trees with a small branching factor, when compared to search trees with a larger branching factor.

Part 1.C Complexity Analysis (12 points)

In this problem we consider the complexity of searching a tree of **depth d** , in which the **branching factor** is not constant, but **varies** as a function of **node level i** . In particular, the branching factor b_i of each node at level i is $b_i = d - i$, with the **root** node appearing at **level 0**. For example, the root has d children, all nodes at level 2 have $d-1$ children, nodes at level 3 have $d-2$ children, and so on; at level d , the tree dies out and no node at level d has a child node).

Part 1.C.1 Number of nodes in the tree (6 points)

Given a tree of depth d , write down an expression for the number of nodes at each level of the tree, up to d . This expression must be a function of the level number m and the depth of the tree d .

Write an expression for the total number of number of nodes in the tree for a tree with depth d . This expression must be a function of the depth of the tree d . (You do **not** need to solve the recursion in this step).

Let $T(d)$ denote the total number of nodes in the tree. Write an equation for the total number of nodes in the tree in a recursive form. That is, write an equation for $T(d)$ in terms of $T(d-1)$ (the number of nodes in a tree that has depth $d-1$). (You do **not** need to solve the recursion in this step).

Part 1.C.2 Worst Case Run-time of Breadth First Search (6 points)

Next consider runtime. What is the **maximum** number of **nodes** that might be **generated (visited)** during **breadth-first search**, in order to reach a **goal** that is placed on level **d**? You can write your answer in terms of the depth of the tree (denoted as **d**) and the total number of nodes in the tree (denoted as **T(d)**).

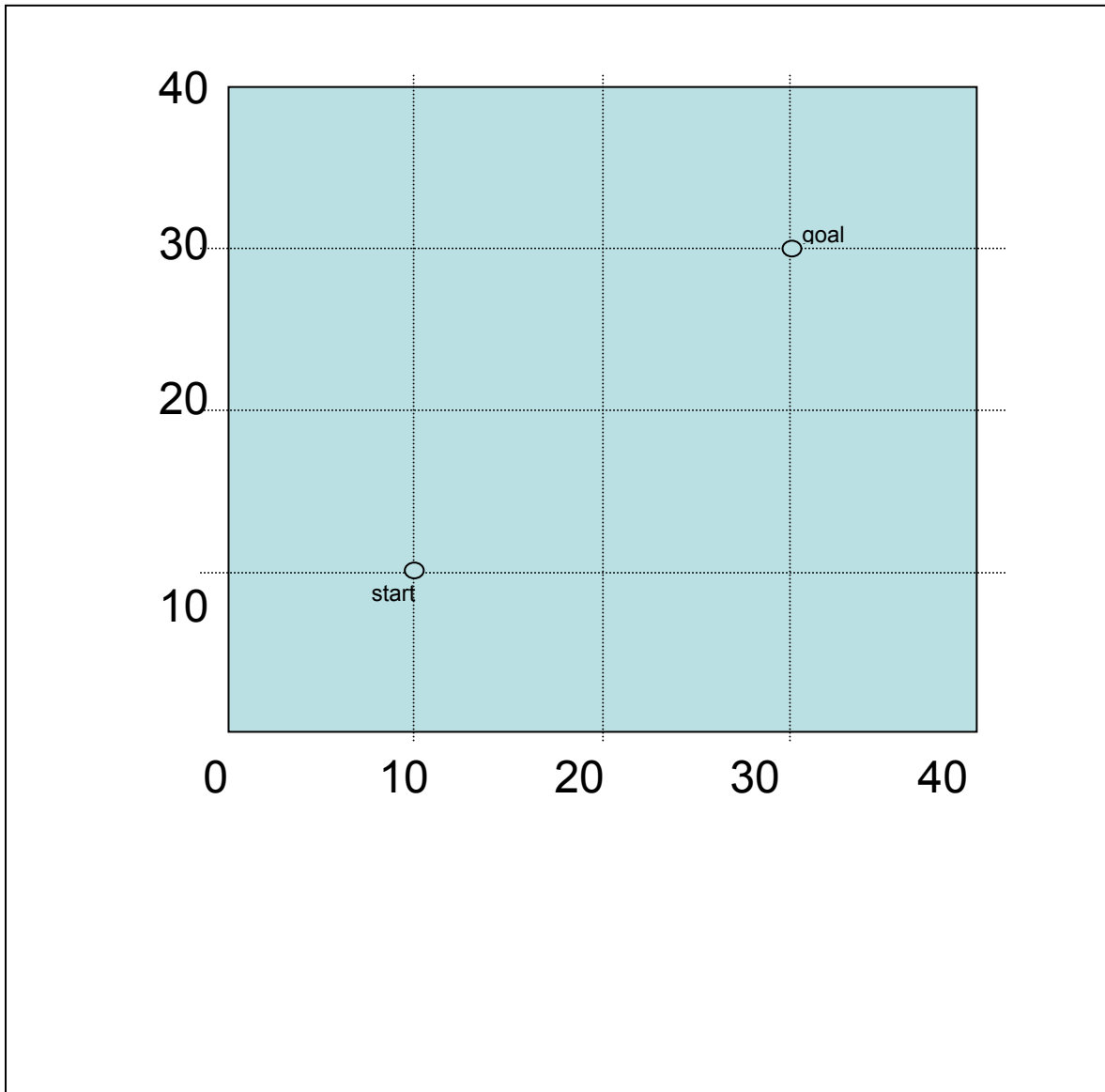
For full credit you must **explain your derivation** below:

Problem 2 Formulating Roadmaps (20 points)

In this problem we considered the mapping of a path-planning problem with obstacles to a road map, using the concepts of configuration spaces and visibility graphs. A robot has dimensions 2 feet by 4 feet (it's a rectangle) and is positioned so that its center is at location $[10, 10]$ (all dimensions in feet). It must reach a goal position at location $[30, 30]$, while avoiding all obstacles. There are two rectangular obstacles, which are five by five foot, and centered at $[20, 30]$ and $[20, 10]$. The robot can translate along any vector, as long as it doesn't collide with an obstacle, but can not rotate.

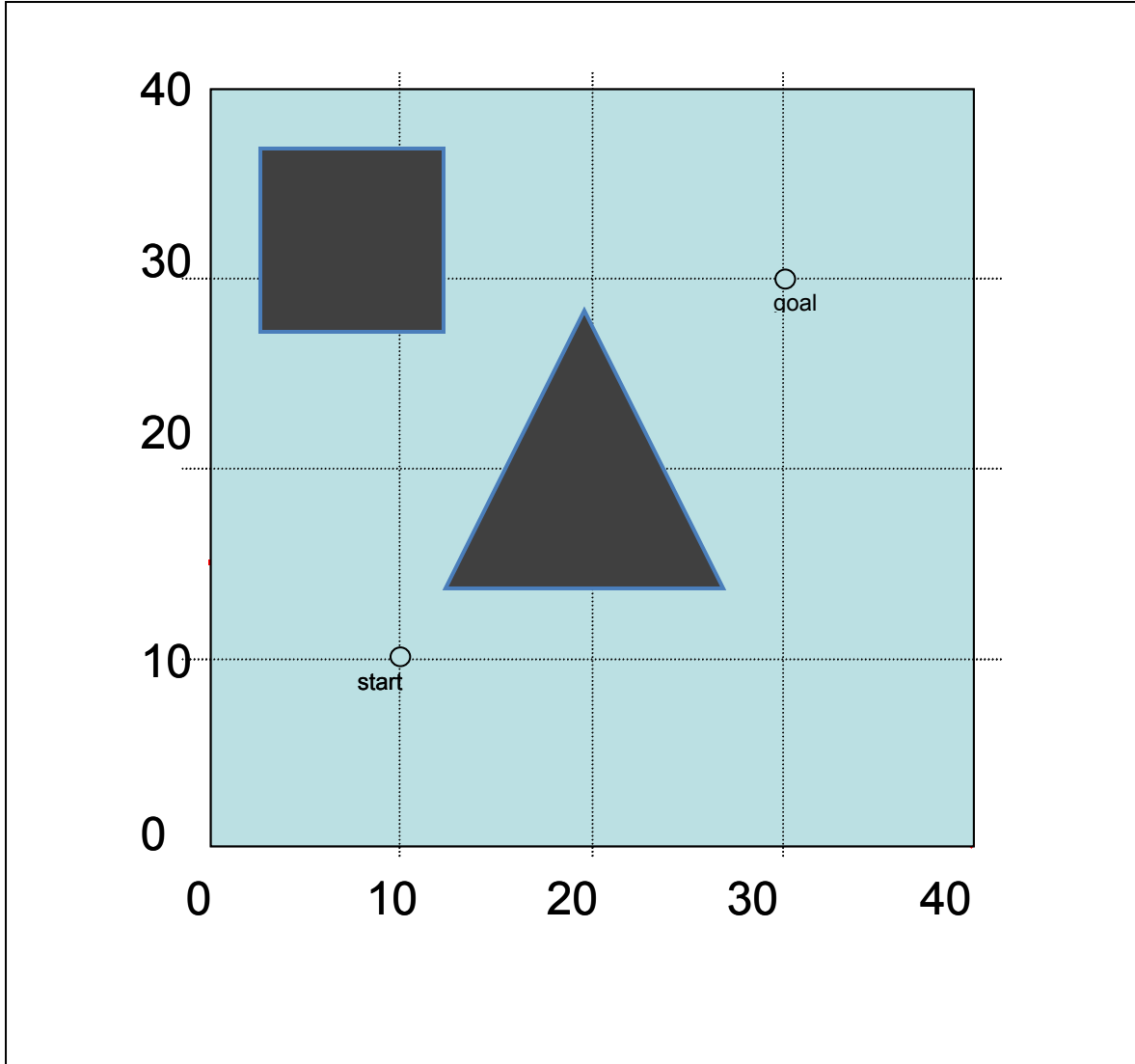
Part 2.A – Drawing the Configuration Space (10 points)

First consider mapping the problem above to an equivalent C-space. In the space below, please draw the C-space map for the problem. As stated earlier, when the robot changes direction, assume it **does not rotate**, hence the orientations of the robot's boundaries do **not change**.



Part 2.B Visibility Graph (10 points)

Consider the problem instance with two obstacles in the following figure. Draw the corresponding visibility graph on the figure.



Problem 3 Constraint Programming (35 points)

In this problem we consider the two essential elements of constraint programming, constraint modeling, and constraint satisfaction.

Part 3.A Constraint Modeling (12 points)

Betty, John and Alfredo are about to order their dinner at a fine restaurant. Your job is to design a robot Waiter who will recommend what they should order. You will accomplish this by formulating the decision problem as a constraint satisfaction problem.

You know that the menu is comprised of the following:

Main Courses

Beef Bourguignon (BB)
Chicken Kiev (CK)
Glazed Salmon (GS) (Fish)
Pasta Primavera (PP) (Vegetarian)
Mushroom Risotto (MR) (Vegetarian)

Wines

Red Cabernet (Ca)
Merlot (M)
White Chardonnay (Ch)
Pinot Grigio (PG)

In addition, Betty, John and Alfredo tell you that your recommendation must satisfy the following constraints:

- Each of the three diners will order a main course and a suitable wine.
- Betty is a vegetarian, and will not eat a course with chicken, beef, or fish.
- John wants to impress Betty, and will order the same main course.
- Red wine can be ordered with beef or fish, but not chicken or vegetarian courses.
- Alfredo is rebellious and will not select wine that either Betty or John has ordered.

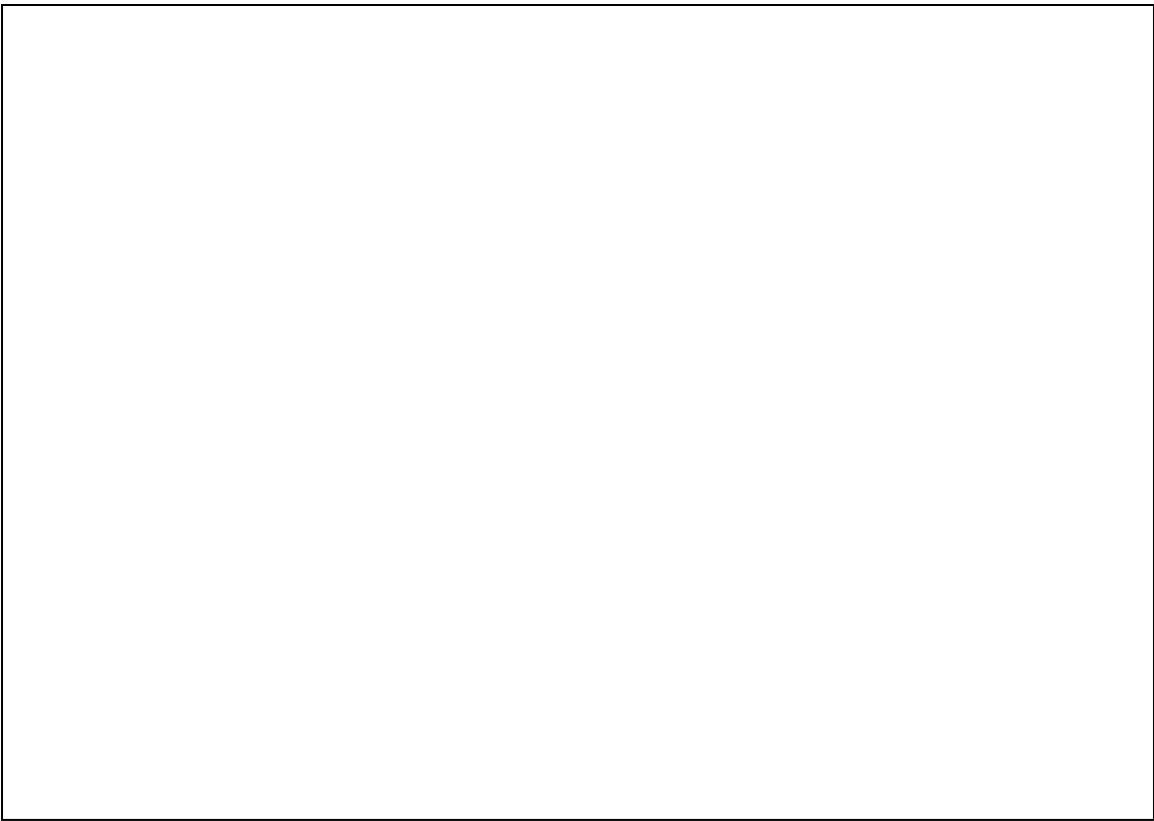
Please formulate this problem as a constraint satisfaction problem.

List all variables and their corresponding domains. Denote the menu items by using their short form (e.g., BB for Beef Bourguignon).

Draw a constraint graph for your encoding of this problem. Label the vertices and edges with a symbol for each corresponding variable and constraint.



Write a description for each constraint, consisting of its scope and relation. Describe the relation by enumerating allowed assignments to the variables in the scope.



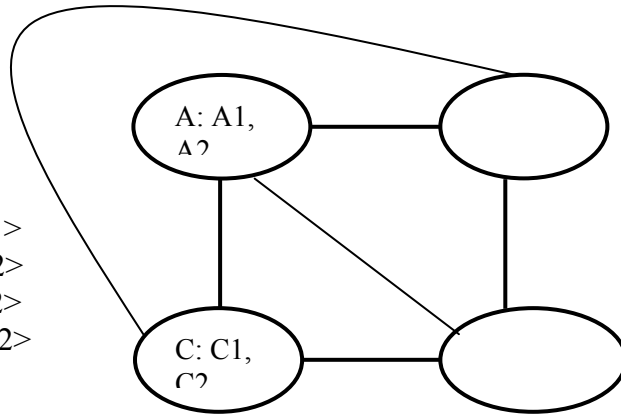
Part 3.B Constraint Satisfaction (23 points)

Consider a constraint satisfaction problem consisting of four variables:

- A: {A1, A2}
- B: {B1, B2}
- C: {C1, C2}
- D: {D1, D2}

And constraints:

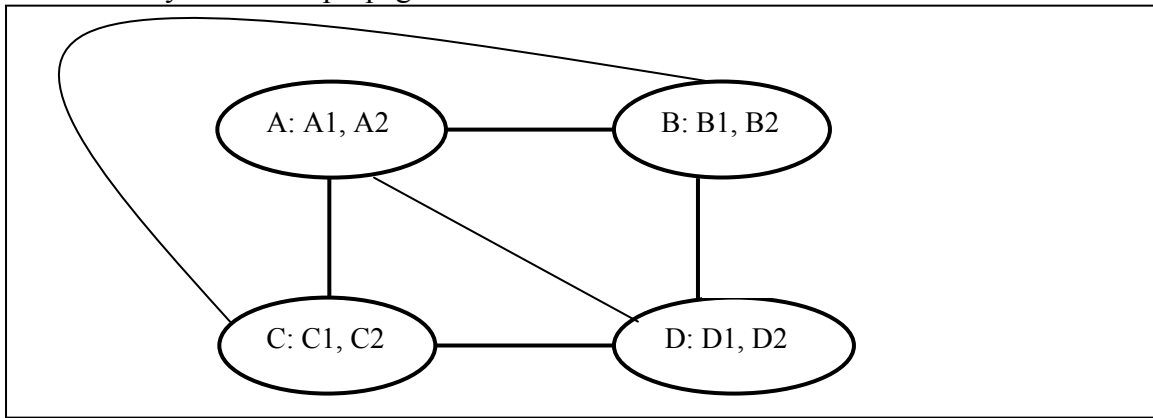
- A-B: $\langle A1, B1 \rangle$ or $\langle A2, B1 \rangle$
- A-C: $\langle A1, C1 \rangle$ or $\langle A2, C2 \rangle$
- B-D: $\langle B1, D1 \rangle$ or $\langle B1, D2 \rangle$
- C-D: $\langle C2, D1 \rangle$ or $\langle C2, D2 \rangle$
- B-C: $\langle B1, C2 \rangle$
- A-D: $\langle A2, D2 \rangle$



Part 3.B.1 Constraint Propagation (10 points)

3.B.1.a Pruned Domains (4 points)

On the following copy of the constraint graph, cross out each domain element that is eliminated by constraint propagation:



3.B.1.b Examined Arcs (6 points)

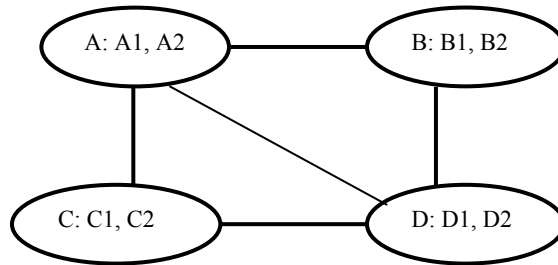
List the arcs examined by constraint propagation, in the order in which they are examined. For each arc, list its pair of variables, and use “>” to indicate the direction in which arc consistency is tested (similar to the lecture notes):

Part 3.B.2 Backtrack Search and Forward Checking (13 points)

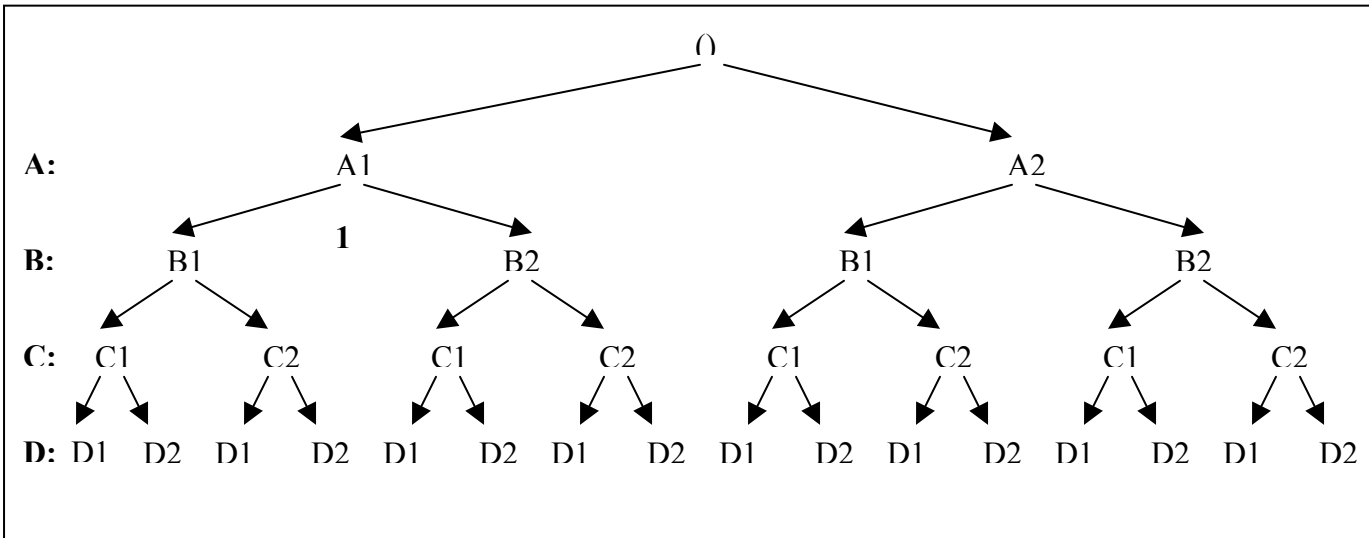
In this part we **search** for the **first** consistent solution to the constraint problem described above, stopping when found. Variables are ordered alphabetically, and values are ordered numerically.

3.B.2.a Backtrack Search (no Forward Checking) (5 points)

- A-B:** $\langle A1, B1 \rangle$ or $\langle A2, B1 \rangle$
- A-C:** $\langle A1, C1 \rangle$ or $\langle A2, C2 \rangle$
- B-D:** $\langle B1, D1 \rangle$ or $\langle B1, D2 \rangle$
- C-D:** $\langle C2, D1 \rangle$ or $\langle C2, D2 \rangle$
- B-C:** $\langle B1, C2 \rangle$
- A-D:** $\langle A2, D2 \rangle$



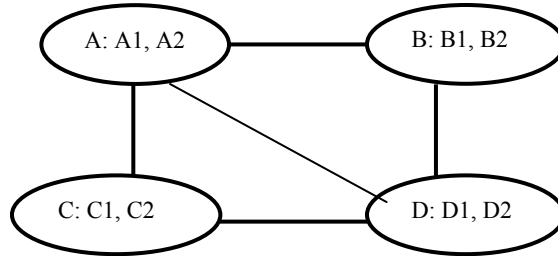
Indicate which assignments are made and the order of those assignments, by writing a number under each assignment made, on the copy below. We have written “1” under the first assignment. Circle the first consistent assignment. Cross-out assignments pruned.



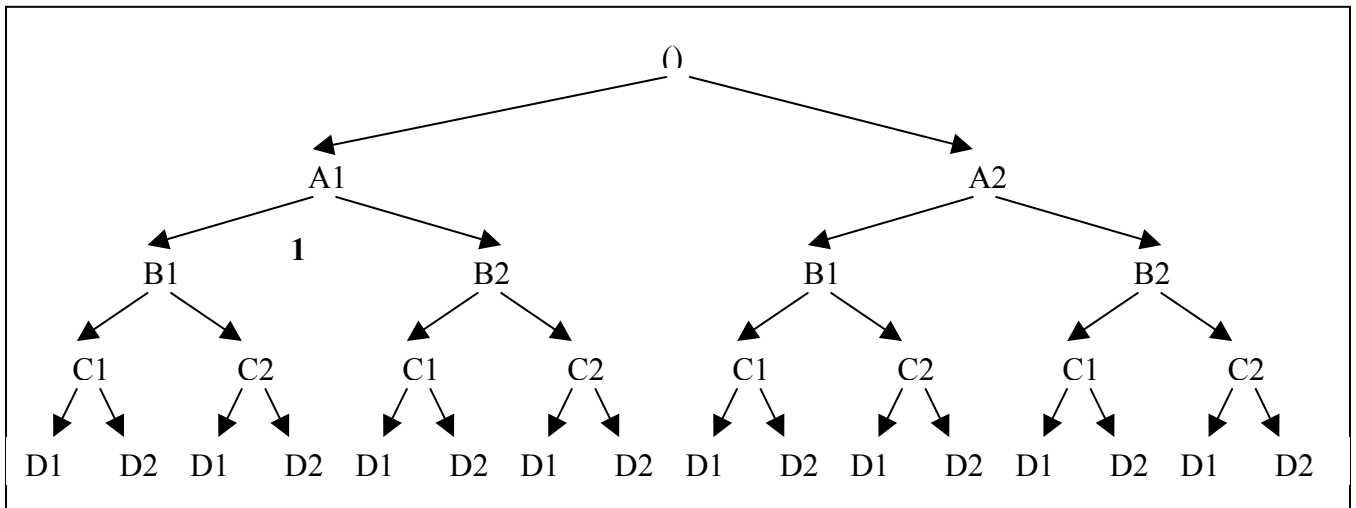
Space available to make comments or show your work:

Part 3.B.2 Backtrack Search with Forward Checking (8 points)

- **A-B:** $\langle A1, B1 \rangle$ or $\langle A2, B1 \rangle$
- **A-C:** $\langle A1, C1 \rangle$ or $\langle A2, C2 \rangle$
- **B-D:** $\langle B1, D1 \rangle$ or $\langle B1, D2 \rangle$
- **C-D:** $\langle C2, D1 \rangle$ or $\langle C2, D2 \rangle$
- **B-C:** $\langle B1, C2 \rangle$
- **A-D:** $\langle A2, D2 \rangle$



Indicate which assignments are made and the order of those assignments, by writing a number under each assignment made on, the copy of the search tree below. We have written 1 under the first assignment. Circle the first consistent assignment. Cross out the assignments that are pruned.



Space available to make comments or show your work:
 (Please indicate the pruned domain values for each step)

Problem 4 – Planning (35 points)

In this problem we consider the formulation of activity planning problems as atomic actions in PDDL, the construction of planning graphs, and properties of solution extraction from the planning graph.

Part 4.A Formulating Planning Problems (10 points)

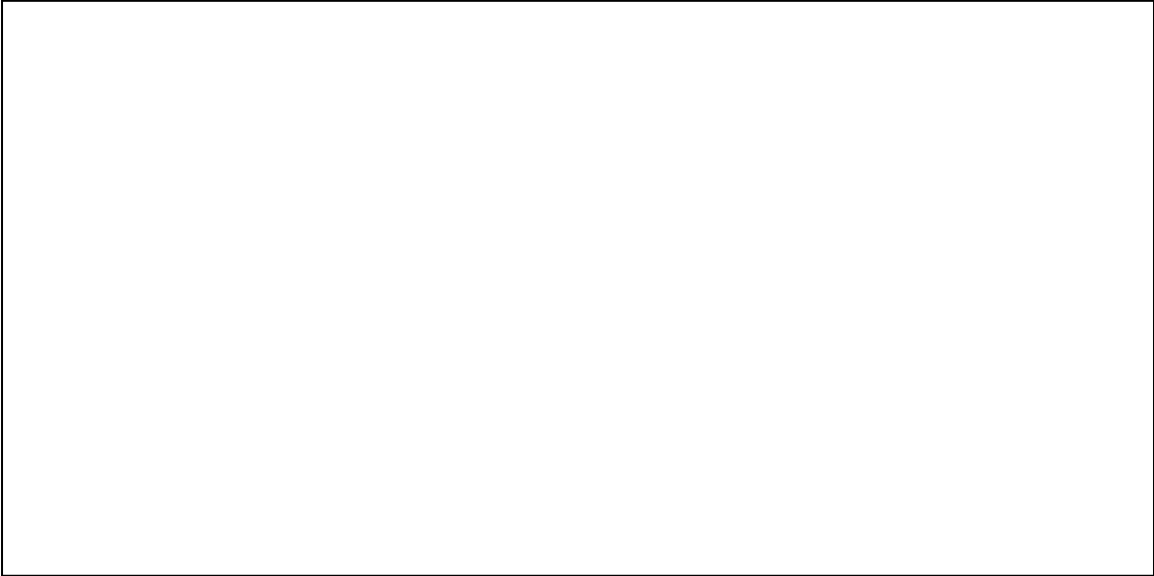
Recently the elevators in the Stata center have been running very slow. Sertac, who sits on the seventh floor, is busy delivering his paper on time, but needs to get the exam papers to Brian's office, which is on the 1st floor. In order not to waste time, Sertac has decided to use one of his robots to carry the exam papers from the 7th floor to the 1st floor. Luckily, his robot already has a built-in activity planner, which takes a problem description in the PDDL format, and generates and executes a plan that satisfies the PDDL description. Please help Sertac formulate this problem using atomic operators in PDDL.

Part 4.A.1 (3 points)

List and describe your predicates, actions, initial condition, and goal predicates.

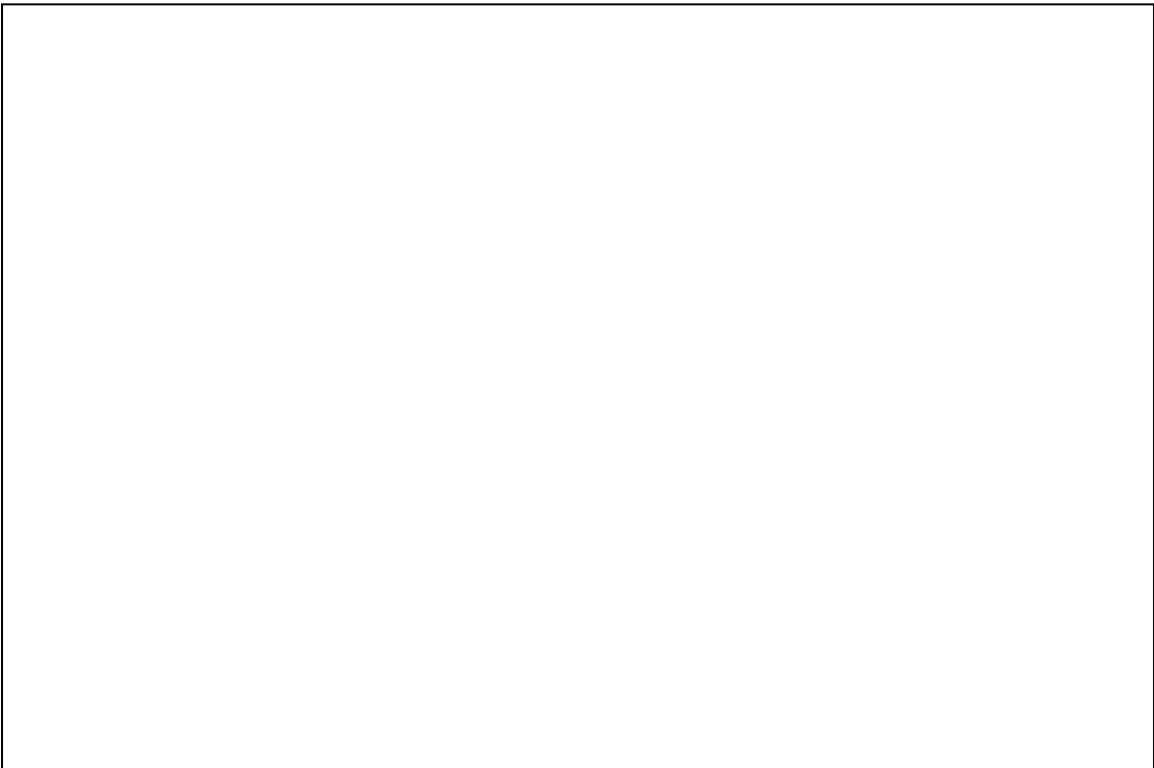
Part 4.A.2 (3 points)

State any assumptions that you make, and justify why your representation is at the right level of abstraction.



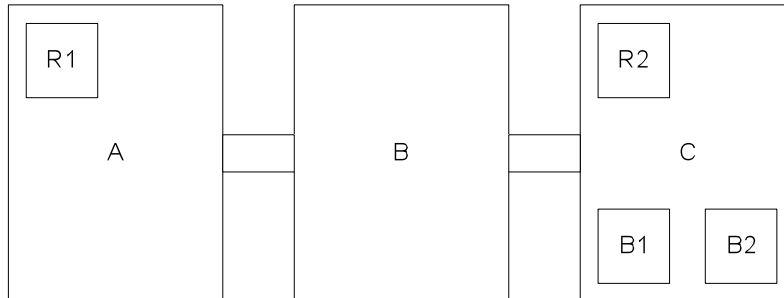
Part 4.A.3 (4 points)

Point out one limitation of atomic actions in PDDL (as presented in class) that limits the effectiveness of your solution. Augment PDDL with a new syntax that remedies this limitation. Explain the meaning of this augmentation and give one example action description.



Part 4.B Constructing Planning Graphs (15 points)

Consider the plan graph for the following problem. The Smith residence is a small house consisting of three rooms, A, B, and C, arranged as shown below:



Dr. Smith is away, but he has left two robots, R1 and R2, in charge of the house. Dr. Smith asked them to move two boxes, B1 and B2, from room C to room A. R2 is initially in C, and R1 is initially in A. Let's help the robots figure out the best way to do this.

To generate a plan for the robots, we formulate the problem in PDDL and give the description to Graph Plan, as follows (note that for the `:parameters` field of `:action`, an expression like `(Robot ?R)` specifies that there is a parameter `?R` that is of type "Robot"):

Operators:

```
(:action move-left
  :parameters ((Robot ?R) (Room ?From) (Room ?To))
  :preconditions (:and
    (left-of ?To ?From)
    (in ?R ?From))
  :effects (:and
    (in ?R ?To)
    (del in ?R ?From)))

(:action move-right
  :parameters ((Robot ?R) (Room ?From) (Room ?To))
  :preconditions (:and
    (right-of ?To ?From)
    (in ?R ?From))
  :effects (:and
    (in ?R ?To)
    (del in ?R ?From)))

(:action carry-left
  :parameters ((Robot ?R) (Box ?Box) (Room ?From) (Room ?To))
  :preconditions (:and
    (left-of ?To ?From)
    (in ?R ?From)
    (in ?Box ?From))
  :effects (:and
    (in ?R ?To)
    (in ?Box ?To)
    (del in ?R ?From)
    (del in ?Box ?From)))
```



```

(:action carry-right
  :parameters ((Robot ?R) (Box ?Box) (Room ?From) (Room ?To))
  :preconditions (:and
    (right-of ?To ?From)
    (in ?R ?From)
    (in ?Box ?From))
  :effects (:and
    (in ?R ?To)
    (in ?Box ?To)
    (del in ?R ?From)
    (del in ?Box ?From)))

```

Initial State Facts:

```

(:init
  (Robot R1)      (in R1 A)
  (Robot R2)      (in R2 C)
  (Room A)        (in B1 C)
  (Room B)        (in B2 C)
  (Room C)        (left-of A B)
  (Box B1)        (left-of B C)
  (Box B2)        (right-of B A)
                  (right-of C B) )

```

Goal State Facts:

```

(:goal
  (in B1 A)      (in B2 A) )

```

Part 4.B.1 Plan Graph Minus Mutex (10 points)

For the PDDL description above, fill in the following plan graph for the first level. Show Level 1 operators and Level 2 facts. Do not show mutex relations:

Level 1 Facts	Level 1 Actions	Level 2 Facts
(in R2 C)		
(in B1 C)		
(in B2 C)		
(in R1 A)		

Part 4.B.2 Mutex for the Plan Graph (5 points)

In the following table, list **three pairs** of mutex actions for the Level 1 actions in Part 4.B.1. For each pair, specify the type of mutex relation (“deletes precondition,” “deletes effect,” or “inconsistent preconditions”). Note that you may not need all of the entries shown in this table:

Action Mutex Pair	Mutex Type
1	
2	
3	

Part 4.C Proving completeness of Graph Plan (10 points)

Give an inductive proof for the fact that the Graph Plan algorithm finds **any** plan of length N that is **complete** and **consistent**.

What is the induction on?

Formulate and prove the base case:

Formulate and prove the induction step:

MIT OpenCourseWare
<http://ocw.mit.edu>

16.410 / 16.413 Principles of Autonomy and Decision Making
Fall 2010

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.