**CS 330 – Programming Languages**

# HW7Prolog2

**The** **Jugs**

In the movie "Die Hard 3", Bruce Willis and Samuel L. Jackson were confronted with the following [puzzle](https://www.youtube.com/watch?v=BVtQNK_ZUJg). They were given a 3-gallon jug and a 5-gallon jug and were asked to fill the 5-gallon jug with exactly 4 gallons. This problem generalizes the puzzle. Blah blah

**Accessary Functions**

Define move rule(s) for the six basic operations of water jugs as in Die Hard, similar to those defined in FWGC. The state of a jug is defined as a list of [WaterInA, WaterInB, CapA, Cap]. The following is the specs. (40 points)

* fillA/2 – fill Jug A, (5 points)
* fillB/2 (5 points)
* emptyA/2 (5 points)
* emptyB/2 (5 points)
* aToB/2 – pour everything from A to B (5 points)
* bToA/2 (5 points)

Each of these functors takes 2 (list) parameters (hence the /2): Current, Next. For example:

?- aToB([1, 1, 5, 3], L).

L = [0, 2, 5, 3].

?- aToB([5, 2, 5, 3], Next).

Next = [4, 3, 5, 3].

You will then define a functor named *move/2* which combines all these operations using ‘;’, ex.,

move(Current, Next) :- fillA(Currect, Next); fillB(Current, Next) …

**B. Define a functor printJugs/1** which accept as argument a list describing the solution path in the form of [state1, state2, …., goalState], where each state is a list of two numbers indicating the amount of water in jugs A and B. It should print the list of moves as indicated in the solution path. (20 points)

**C. Depth-First Search** (40 points)

Write prolog rules that will allow a depth-first search by SWI Prolog to solve the jugs problem.

Be sure to study class examples such as FWGC as your solution could be very similar the example. Note that the top-level rule:

jugs(SizeA, SizeB, GoalSize, SolutionPath).

take 4 arguments*: SizeA, SizeB, GoalSize*, and *SolutionPath*, where *SizeA* and *SizeB* are the sizes of the two jugs A and B, respectively, with SizeA >= SizeB, *GoalSize* is the desired amount of water in the larger jugs, and SolutionPath is the variable which will be instantiated with the solution path when it’s found. Since the SolutionPath could be longer than what swi-porlog is willing to display with full details, your rule must call the printJugs functor to print the list of moves when a solution is found.

A diagram of a search process

Description automatically generated

A computer code with black text

Description automatically generated

A diagram of a diagram

Description automatically generated

A diagram of a diagram

Description automatically generated

A diagram of a tree

Description automatically generated

A close-up of a search engine diagram

Description automatically generated

A whiteboard with writing on it

Description automatically generated