Buying wine

A wine seller has two wine jugs, o small one of 4 liters capacity, and a larger one of 9 liters. There is no measuring label mentioned on either of these two jugs i.e. he cannot know the exact amount filled in the jug. Can we measure all values from 1 to 9 using these unmarked jugs? (a generalisation of classical water jugs problem)

- Let $j_1 = 4$ be the capacity of the small jug and $j_2 = 9$ the capacity of the large jug.
- We use a *state space search*, where each state is represented with $J(x, y)$: x is the current amount of wine in jug j_1 and y the amount in jug j_2.
- The initial state is $J(0, 0)$.
- We use production rules to change the states of the system ($set(production)$)

For the current state $J(x, y)$, the following eight actions are possible:

1. Fill-in the small jug: $J(x, y) \rightarrow J(j_1, y)$
2. Empty the small jug: $J(x, y) \rightarrow J(0, y)$
3. Fill-in the large jug: $J(x, y) \rightarrow J(x, j_2)$
4. Empty the large jug: $J(x, y) \rightarrow J(x, 0)$
5. Empty the small jug into the large jug, if capacity allows this: $J(x, y) \wedge x + y \leq j_2 \rightarrow J(0, y + x)$
6. If j_2 does not suffice to empty j_1 then move some amount from the small jug to the larger jug, until j_2 is full: $J(x, y) \wedge x + y > j_2 \rightarrow J(x - (j_2 - y), j_2)$
7. If capacity of j_2 allows, empty the large jug into the small jug: $J(x, y) \wedge x + y \leq j_1 \rightarrow J(x + y, 0)$
8. If capacity of j_1 does not suffice to empty j_2 then move some amount from j_2, until the j_1 is full: $J(x, y) \wedge x + y > j_1 \rightarrow J(j_1, y - (j_1 - x))$

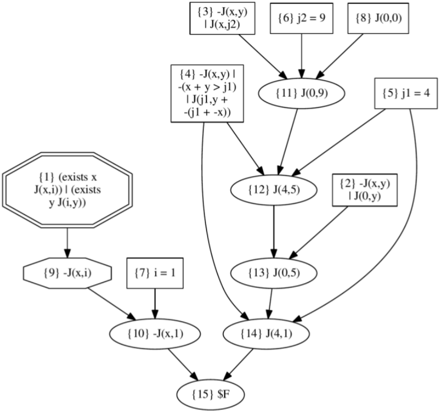
Planning with Prover9

- Rules are written in clausal form, in order to allow variables in the `#answer` directive
- The `#answer` directive is useful to print the steps for reaching the goal.

```
set(production).

formulas(demodulators).
  j1 = 4.      j2 = 9                          %small jug and large jug
  i  = 2.                                      %value to measure i=[1..9]
end_of_list.

formulas(usable).
  -J(x,y) | J(j1,y) #answer("fill the small jug")  #answer(J($j1,y)).
  -J(x,y) | J(0,y)  #answer("empty the small jug") #answer(J(0,y)).
  -J(x,y) | J(x,j2) #answer("fill the big jug")    #answer(J(x,$j2)).
  -J(x,y) | J(x,0)  #answer("empty the big jug")   #answer(J(x,0)).
  -J(x,y) | -(x+y <= j2) | J(0,y+x)
    #answer("empty the small jug into the big jug") #answer(J(0,x+y)).
   -J(x,y) | -(x+y > j2) | J(x+ -(j2+ -y),j2)
    #answer("small into big, until full") #answer(J(x+ -(j2+ -y),$j2)).
  -J(x,y) | -(x+y <= j1) | J(x+y,0)
    #answer("empty the big jug into the small jug") #answer(J(x+y, 0)).
  -J(x,y) | -(x+y >  j1) | J(j1,y + -(j1+ -x))
    #answer("big into small, until full") #answer(J($j1,y+ -($j1+ -x))).
end_of_list.

formulas(assumptions).
  J(0,0)                                     #answer("Init state: J(0,0)").
end_of_list.

formulas(goals).
  exists x J(x,i) | exists y J(i,y).
end_of_list.
```

Measuring 1 liter of wine $\exists x\ J(x,1) \vee \exists y\ J(1,y)$

$Proof_1$	#answer(Action)	#answer(State)
1	"init state"	$J(0,0)$
2	"fill the big jug"	$J(0,j_2)$
3	"big into small, until full"	$J(j_1, 9-(j_1-0))$
4	"empty the small jug"	$J(0,5)$
5	"big into small, until full"	$J(j_1, 5-(j_1-0))$



Measuring 2 liters of wine

$\exists x\ J(x, 2) \vee \exists y\ J(2, y)$

$Proof_2$	#answer(Action)	#answer(State)
1	"init state"	$J(0, 0)$
2	"fill the big jug"	$J(0, j_2)$
3	"big into small, until full"	$J(j_1, 9 - (j_1 - 0))$
4	"empty the small jug"	$J(0, 5)$
5	"big into small, until full"	$J(j_1, 5 - (j_1 - 0))$
6	"empty the small jug"	$J(0, 1)$
7	"empty the big jug into the small jug"	$J(0 + 1, 0)$
8	"fill the big jug"	$J(1, j_2)$
9	"big into small, until full"	$J(j_1, 9 - (j_1 - 1))$
10	"empty the small jug"	$J(0, 6)$
11	"big into small, until full"	$J(j_1, 6 - (j_1 - 0)))$