A Simple Problem

The Goal is to put 3 gallons in a jug
(simple enough)

- What are the possible states available within the water jug problem. What is the essential information that must be kept.
- What is the initial state – The jug is empty.
- What is the final state – The jug has 3 gallons of water in it
- What kind of rules or manipulations on the states to go from the initial state to the final state.
  - Add One Gallon
  - Add Two Gallons

The State Status

```
;-----------------------------------------------------------------------
; The status of the water in the water-jug
;
(deftemplate status
        (slot water-jug
                (type INTEGER)
                (default 0)))
```

The Initial State Setup

```
;---------------------------------------------------------
;  The initial Status
;;
(defrule Initial-State
        ?initial <- (initial-fact)
=>
        (printout t "The Initial State is an Empty Jug" crlf)
        (assert (status (water-jug 0))))
```

The Rules

```
;--------------------------------------------------------
; Rule for Adding One Gallon to Water Jug
;
(defrule Add-One-Gallon
        ?status <- (status
                        (water-jug ?amount))
=>
        (printout t "Adding 1 Gallon to "
                 ?amount
                 " to make "
                 (+ 1 ?amount)
                 " gallons" crlf)
        (modify ?status (water-jug (+ ?amount 1))))

;---------------------------------------------------------
; Rule for Adding Two Gallons to Water Jug
;
(defrule Add-Two-Gallons
        ?status <- (status
                        (water-jug ?amount))
=>
        (printout t "Adding 2 Gallons to "
                    ?amount
                    " to make "
                    (+ 2 ?amount)
                    " gallons" crlf)
        (modify ?status (water-jug (+ ?amount 2))))
```

The Final State

```
;-------------------------------------------------------
; Rule for Detecting Done with Task (i.e. the jug has 3 gallons in it)
;
(defrule Done
        ?status <- (status
                        (water-jug 3))
=>
        (printout "Done" crlf))
```

First Attempt at a Run

```
CLIPS> (reset)
CLIPS> (run 1)
The Initial State is an Empty Jug
CLIPS> (run 1)
Adding 1 Gallon to 0 to make 1 gallons
CLIPS> (run 1)
Adding 1 Gallon to 1 to make 2 gallons
CLIPS> (run 1)
Adding 1 Gallon to 2 to make 3 gallons
CLIPS> (run 1)
Adding 1 Gallon to 3 to make 4 gallons
CLIPS> (run 1)
Adding 1 Gallon to 4 to make 5 gallons
CLIPS> (run 1)
Adding 1 Gallon to 5 to make 6 gallons
```

What Happened?

```
CLIPS> (run 1)
The Initial State is an Empty Jug
CLIPS> (agenda)
0      Add-One-Gallon: f-1
0      Add-Two-Gallons: f-1
For a total of 2 activations.
CLIPS> (run 1)
Adding 1 Gallon to 0 to make 1 gallons
CLIPS> (agenda)
0      Add-One-Gallon: f-2
0      Add-Two-Gallons: f-2
For a total of 2 activations.
CLIPS> (run 1)
Adding 1 Gallon to 1 to make 2 gallons
CLIPS> (agenda)
0      Add-One-Gallon: f-3
0      Add-Two-Gallons: f-3
For a total of 2 activations.
CLIPS> (run 1)
Adding 1 Gallon to 2 to make 3 gallons
CLIPS> (agenda)
0      Add-One-Gallon: f-4
0      Add-Two-Gallons: f-4
0      Done: f-4
For a total of 3 activations.
CLIPS> (run 1)
Adding 1 Gallon to 3 to make 4 gallons
CLIPS> (agenda)
0      Add-One-Gallon: f-5
0      Add-Two-Gallons: f-5
For a total of 2 activations.
```

One Solution ... Salience

```
;-------------------------------------------------------
; Rule for Detecting Done with Task (i.e. the jug has 3 gallons in it)
; Put Salience High so when matched, it is the first rule
;
(defrule Done
        (declare (salience 1000))
        ?status <- (status
                        (water-jug 3))
=>
        (printout t "Done - Jug has Three Gallons" crlf))

CLIPS> (reset)
CLIPS> (run 1)
The Initial State is an Empty Jug
CLIPS> (run 1)
Adding 1 Gallon to 0 to make 1 gallons
CLIPS> (run 1)
Adding 1 Gallon to 1 to make 2 gallons
CLIPS> (run 1)
Adding 1 Gallon to 2 to make 3 gallons
CLIPS> (agenda)
1000   Done: f-4
0      Add-One-Gallon: f-4
0      Add-Two-Gallons: f-4
For a total of 3 activations.
CLIPS> (run 1)
Done - Jug has Three Gallons
```

Are We Really Done?

```
CLIPS> (run 1)
Done - Jug has Three Gallons
CLIPS> (run 1)
Adding 1 Gallon to 3 to make 4 gallons
CLIPS> (run 1)
Adding 1 Gallon to 4 to make 5 gallons
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-6     (status (water-jug 5)) CF 1.00
For a total of 2 facts.
CLIPS> (run 1)
Adding 1 Gallon to 5 to make 6 gallons
CLIPS> (run 1)
Adding 1 Gallon to 6 to make 7 gallons
```

Retract The Status when done

```
;-------------------------------------------------------
; Rule for Detecting Done with Task (i.e. the jug has 3 gallons in it)
; Put Salience High so when matched, it is the first rule
;
(defrule Done
        (declare (salience 1000))
        ?status <- (status
                        (water-jug 3))
=>
        (printout t "Done - Jug has Three Gallons" crlf)
        (retract ?status))
```

```
CLIPS> (reset)
CLIPS> (run 1)
The Initial State is an Empty Jug
CLIPS> (run 1)
Adding 1 Gallon to 0 to make 1 gallons
CLIPS> (run 1)
Adding 1 Gallon to 1 to make 2 gallons
CLIPS> (run 1)
Adding 1 Gallon to 2 to make 3 gallons
CLIPS> (run 1)
Done - Jug has Three Gallons
CLIPS> (run 1)
CLIPS> (run 1)
CLIPS> (run 1)
CLIPS> (agenda)
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
For a total of 1 fact.
```

The other Rule?

Rule Two was not used

Why?

What Happens if Rule two is put first?

```
CLIPS> (reset)
CLIPS> (run 1)
The Initial State is an Empty Jug
CLIPS> (run 1)
Adding 2 Gallons to 0 to make 2 gallons
CLIPS> (run 1)
Adding 2 Gallons to 2 to make 4 gallons
CLIPS> (run 1)
Adding 2 Gallons to 4 to make 6 gallons
CLIPS> (run 1)
Adding 2 Gallons to 6 to make 8 gallons
CLIPS> (run 1)
Adding 2 Gallons to 8 to make 10 gallons
```

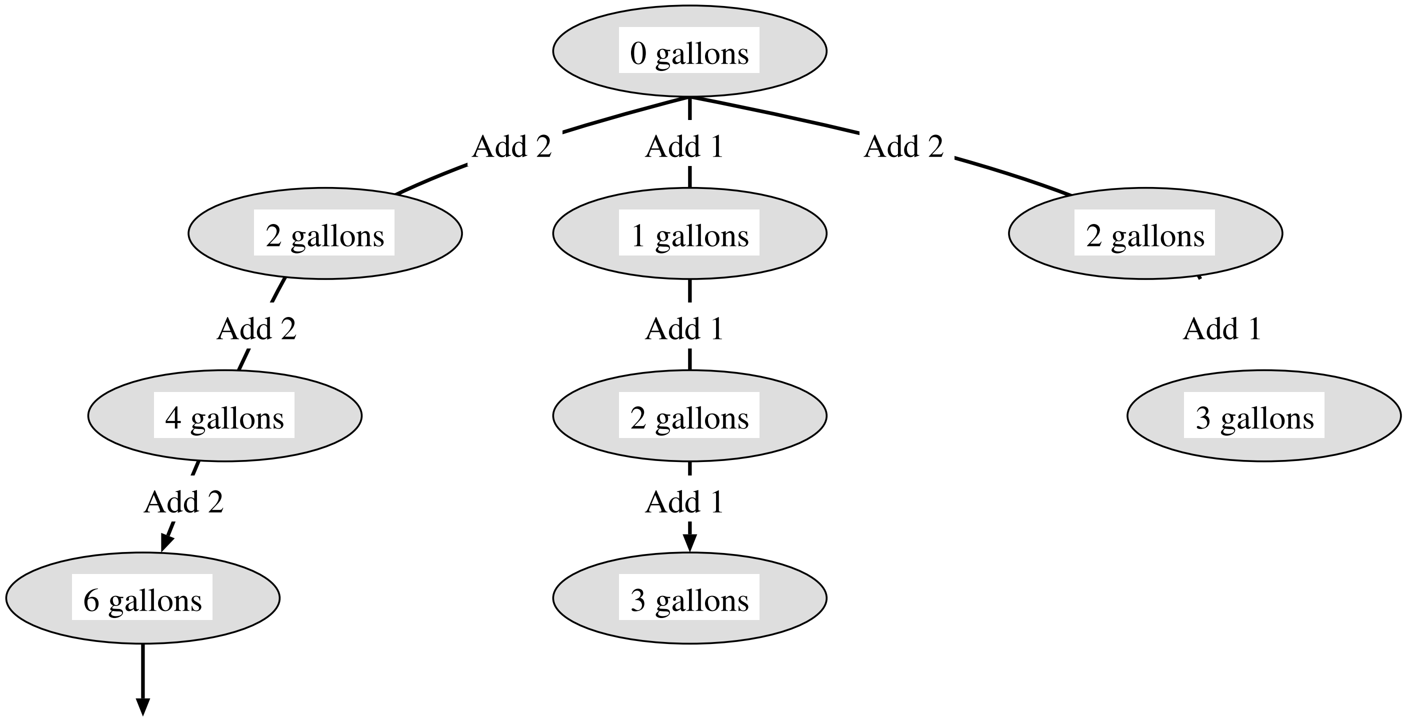
Problems

It either Finds a Solution given the setup

or it doesn't (and goes forever)

There is no **Search**

Partial Search Space of Solutions



How Can we Set up the Search?

Only One State is Current

Can only go down One Path

There is not History kept.

The Template

```
;-----------------------------------------
; The status of the water in the water-jug
;
(deftemplate status
        (slot water-jug
                (type INTEGER)
                (default 0))
        (slot depth
                (type INTEGER)
                (default 0))
        )
```

The Initialization

```
;------------------------------------------------------
;  The initial Status
;;
(defrule Initial-State
        ?initial <- (initial-fact)
=>
        (printout t "The Initial State is an Empty Jug" crlf)
        (assert (depth 0))
        (assert (status (water-jug 0) (depth 0)))
        )
```

The Intialization

```
CLIPS> (reset)
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
For a total of 1 fact.
CLIPS> (agenda)
0      Initial-State: f-0
For a total of 1 activation.
CLIPS> (run 1)
The Initial State is an Empty Jug
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-1     (depth 0) CF 1.00
f-2     (status (water-jug 0) (depth 0)) CF 1.00
For a total of 3 facts.
CLIPS> (agenda)
500    Add-One-Gallon: f-2,f-1
500    Add-Two-Gallons: f-2,f-1
100    update-depth: f-1
For a total of 3 activations.
```

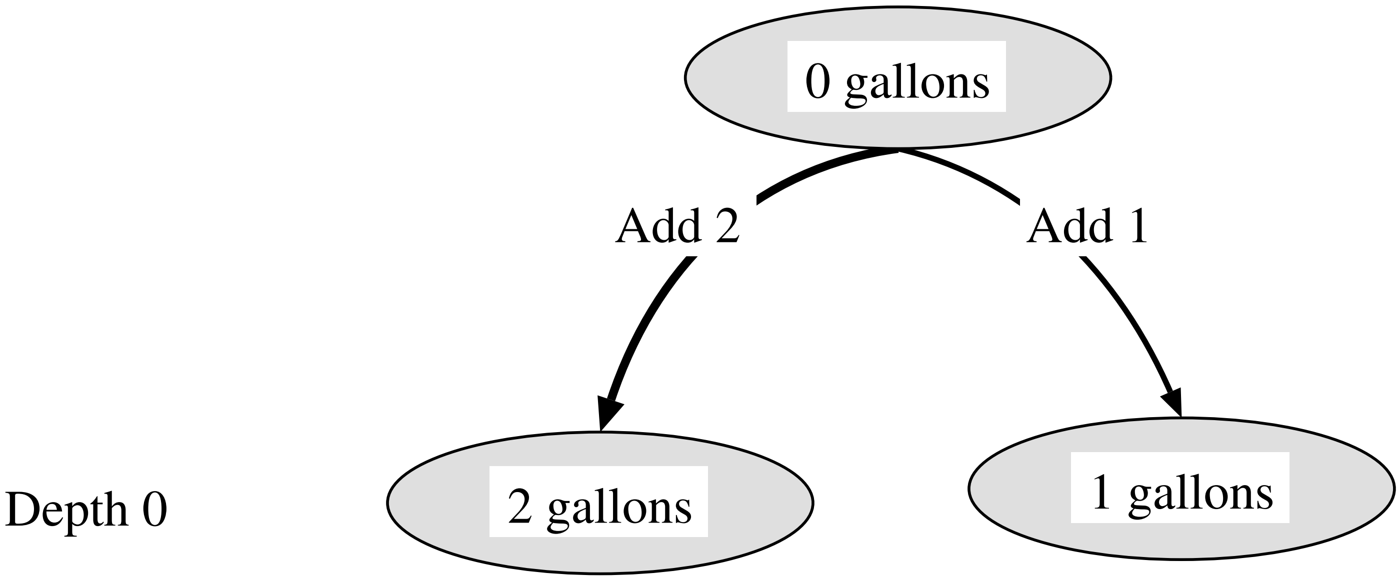
The Rule

```
;--------------------------------------------------
; Rule for Adding One Gallon to Water Jug
;
(defrule Add-One-Gallon
        (declare (salience 500))
        ?status <- (status
                        (water-jug ?amount)
                        (depth ?depth))
        (depth ?current)
        (test (= ?depth ?current))
=>
        (printout t "Adding 1 Gallon to "
                  ?amount " to make "
                  (+ 1 ?amount) " gallons" crlf)
        (assert (status (water-jug (+ ?amount 1))
                        (depth (+ ?depth 1))))
        )
```

Creating the First Level

```
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-1     (depth 0) CF 1.00
f-2     (status (water-jug 0) (depth 0)) CF 1.00
For a total of 3 facts.
CLIPS> (agenda)
500    Add-One-Gallon: f-2,f-1
500    Add-Two-Gallons: f-2,f-1
100    update-depth: f-1
For a total of 3 activations.
CLIPS> (run 1)
Adding 1 Gallon to 0 to make 1 gallons
CLIPS> (run 1)
Adding 2 Gallons to 0 to make 2 gallons
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-1     (depth 0) CF 1.00
f-2     (status (water-jug 0) (depth 0)) CF 1.00
f-3     (status (water-jug 1) (depth 1)) CF 1.00
f-4     (status (water-jug 2) (depth 1)) CF 1.00
For a total of 5 facts.
CLIPS> (agenda)
100    update-depth: f-1
For a total of 1 activation.
```

The Search Tree



Breadth First

```
;--------------------------------------------------------------
;  Rule to update current depth
;    Called only if depth is done for all rules for this level
;    (hence the lower salience)
;
(defrule update-depth
        (declare (salience 100))
        ?depth <- (depth ?current)
=>
        (retract ?depth)
        (assert (depth (+ 1 ?current)))
        )
```

Initialization of Next Level

```
CLIPS> (agenda)
100    update-depth: f-1
For a total of 1 activation.
CLIPS> (run 1)
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-2     (status (water-jug 0) (depth 0)) CF 1.00
f-3     (status (water-jug 1) (depth 1)) CF 1.00
f-4     (status (water-jug 2) (depth 1)) CF 1.00
f-5     (depth 1) CF 1.00
For a total of 5 facts.
CLIPS> (agenda)
500    Add-One-Gallon: f-3,f-5
500    Add-One-Gallon: f-4,f-5
500    Add-Two-Gallons: f-3,f-5
500    Add-Two-Gallons: f-4,f-5
100    update-depth: f-5
For a total of 5 activations.
```

The Next Level

```
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-2     (status (water-jug 0) (depth 0)) CF 1.00
f-3     (status (water-jug 1) (depth 1)) CF 1.00
f-4     (status (water-jug 2) (depth 1)) CF 1.00
f-5     (depth 1) CF 1.00
For a total of 5 facts.
CLIPS> (run 1)
Adding 1 Gallon to 1 to make 2 gallons
CLIPS> (run 1)
Adding 1 Gallon to 2 to make 3 gallons
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-2     (status (water-jug 0) (depth 0)) CF 1.00
f-3     (status (water-jug 1) (depth 1)) CF 1.00
f-4     (status (water-jug 2) (depth 1)) CF 1.00
f-5     (depth 1) CF 1.00
f-6     (status (water-jug 2) (depth 2)) CF 1.00
f-7     (status (water-jug 3) (depth 2)) CF 1.00
For a total of 7 facts.
CLIPS> (agenda)
1000   Done: f-7
500    Add-Two-Gallons: f-3,f-5
500    Add-Two-Gallons: f-4,f-5
100    update-depth: f-5
For a total of 4 activations.
```

A Solution Was Found!!!!

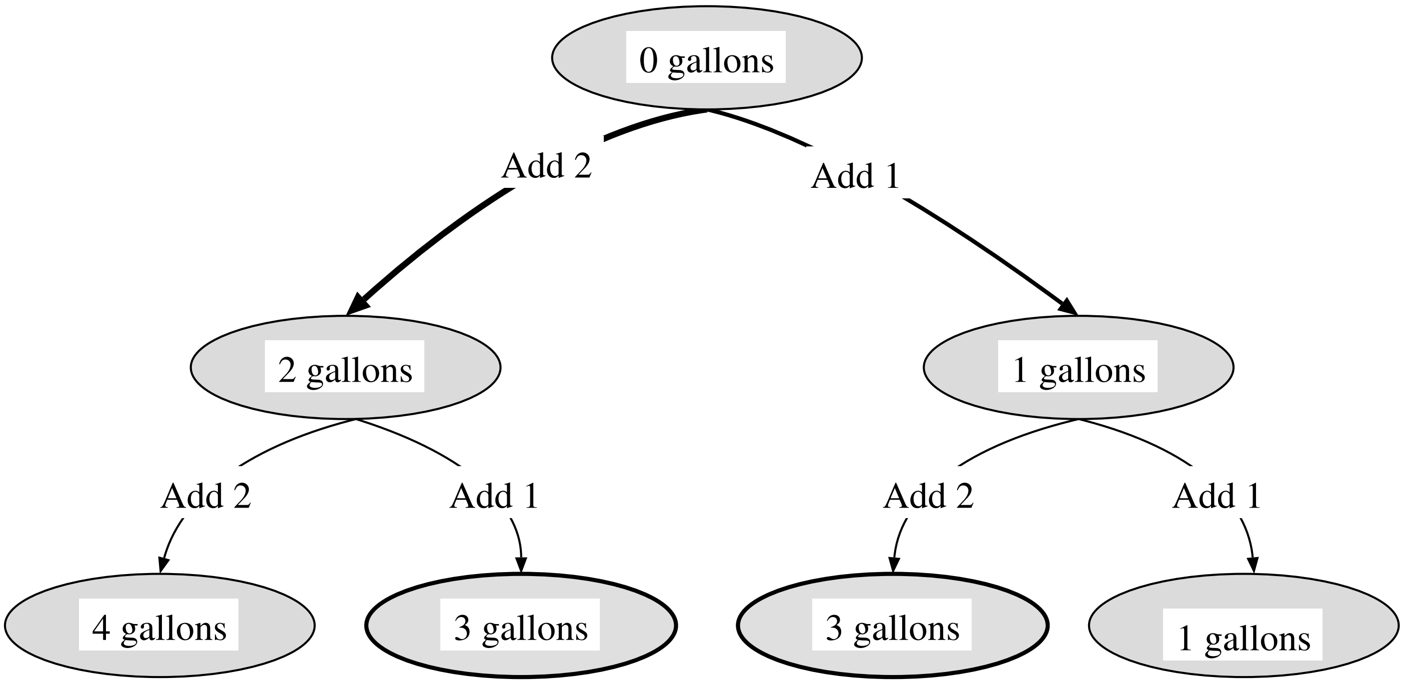
```
;-----------------------------------------------------------------------------------------
; Rule for Detecting Done with Task (i.e. the jug has 3 gallons in it)
; Put Salience High so when matched, it is the first rule
;
(defrule Done
        (declare (salience 1000))
        ?status <- (status
                        (water-jug 3))
=>
        (printout t "Done - Jug has Three Gallons" crlf)
        (retract ?status))

CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-2     (status (water-jug 0) (depth 0)) CF 1.00
f-3     (status (water-jug 1) (depth 1)) CF 1.00
f-4     (status (water-jug 2) (depth 1)) CF 1.00
f-5     (depth 1) CF 1.00
f-6     (status (water-jug 2) (depth 2)) CF 1.00
f-7     (status (water-jug 3) (depth 2)) CF 1.00
For a total of 7 facts.
CLIPS> (run 1)
Done - Jug has Three Gallons
```

Completion of Level

```
CLIPS> (run 1)
Adding 2 Gallons to 1 to make 3 gallons
CLIPS> (run 1)
Done - Jug has Three Gallons
CLIPS> (agenda)
500    Add-Two-Gallons: f-4,f-5
100    update-depth: f-5
For a total of 2 activations.
CLIPS> (run 1)
Adding 2 Gallons to 2 to make 4 gallons
CLIPS> (agenda)
100    update-depth: f-5
For a total of 1 activation.
```

Depth 1: Two Solutions



Depth 1

A Full Run

```
CLIPS> (reset)
CLIPS> (run 1)
The Initial State is an Empty Jug
Adding 1 Gallon to 0 to make 1 gallons
Adding 2 Gallons to 0 to make 2 gallons
Adding 1 Gallon to 1 to make 2 gallons
Adding 1 Gallon to 2 to make 3 gallons
Done - Jug has Three Gallons
Adding 2 Gallons to 1 to make 3 gallons
Done - Jug has Three Gallons
Adding 2 Gallons to 2 to make 4 gallons
Adding 1 Gallon to 2 to make 3 gallons
Done - Jug has Three Gallons
Adding 1 Gallon to 4 to make 5 gallons
Adding 2 Gallons to 2 to make 4 gallons
Adding 2 Gallons to 4 to make 6 gallons
Adding 1 Gallon to 5 to make 6 gallons
Adding 1 Gallon to 4 to make 5 gallons
Adding 1 Gallon to 6 to make 7 gallons
```

We Are not done

The Structure of the Search Has Been Setup

Breadth First

All the Solutions Can Be Found

What is Needed?

Once a solution is found, want to print out the path of events

The New Template

```
;------------------------------------------------------------------------------------------
; The status of the water in the water-jug
;
(deftemplate status
        (slot water-jug
                (type INTEGER)
                (default 0)
                (range 0 5))
        (slot depth
                (type INTEGER)
                (default 0))
        (slot node
                (type INTEGER)
                (default 0))
        (slot parent
                (type FACT-ADDRESS SYMBOL)
                (default no-parent)
                (allowed-symbols no-parent))
        (slot operation
                (type STRING))
        )
```

Initialization

```
;-----------------------------------------------------------------------------------------
;  The initial Status
;;
(defrule Initial-State
        ?initial <- (initial-fact)
=>
        (printout t "The Initial State is an Empty Jug" crlf)
        (assert (depth 0 0))
        (assert (status (water-jug 0) (depth 0)
                        (parent no-parent)))
        (assert (nodes 1))
        )
```

Initialize: Create the root node

```
CLIPS> (reset)
CLIPS> (run 1)
The Initial State is an Empty Jug
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-1     (depth 0 0) CF 1.00
f-2     (status (water-jug 0) (depth 0) (node 0) (parent no-parent) (operation "")) CF 1.00
f-3     (nodes 1) CF 1.00
For a total of 4 facts.
CLIPS> (agenda)
600    Increment-Node-Count: f-2,f-3
500    Add-One-Gallon: f-2,f-1
500    Add-Two-Gallons: f-2,f-1
100    finished: f-1,f-3
For a total of 4 activations.
```

Node Count

```
;-----------------------------------------------------------------------------------------
(defrule Increment-Node-Count
       (declare (salience 600))
       ?status <- (status (node ?statusnode))
       ?nodes <- (nodes ?nodecount)
       (test (= ?statusnode 0))
=>
  (assert (nodes (+ 1 ?nodecount)))
  (modify ?status (node ?nodecount))
  (retract ?nodes)
  )
```

Increment Node Count

Note Salience (Puts rule ahead of normal fill rules)

This rule fires whenever a new rule is created

This will alternate with creating nodes in the search tree

```
CLIPS> (run 1)
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-1     (depth 0 0) CF 1.00
f-4     (nodes 2) CF 1.00
f-5     (status (water-jug 0) (depth 0) (node 1) (parent no-parent) (operation "")) CF 1.00
For a total of 4 facts.
CLIPS> (agenda)
500    Add-One-Gallon: f-5,f-1
500    Add-Two-Gallons: f-5,f-1
100    update-depth: f-1,f-4
For a total of 4 activations.
```

The Main Rule

```
;--------------------------------------------
; Rule for Adding One Gallon to Water Jug
;
(defrule Add-One-Gallon
        (declare (salience 500))
        ?status <- (status
                        (water-jug ?amount)
                        (depth ?depth))
        (depth ?current ?)
        (test (= ?depth ?current))
        (test (< ?amount 3))
=>
        (assert (status (water-jug (+ ?amount 1))
                        (depth (+ ?depth 1))
                        (parent ?status)
                        (operation "Add One Gallon")
                        ))
        )
```

```
CLIPS> (run 1)
CLIPS> (agenda)
500    Add-One-Gallon: f-5,f-1
500    Add-Two-Gallons: f-5,f-1
100    update-depth: f-1,f-4
For a total of 3 activations.
CLIPS> (run 1)
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-1     (depth 0 0) CF 1.00
f-4     (nodes 2) CF 1.00
f-5     (status (water-jug 0) (depth 0) (node 1)
                (parent no-parent) (operation "")) CF 1.00
f-6     (status (water-jug 1) (depth 1) (node 0)
                (parent <Fact-5>) (operation "Add One Gallon")) CF 1.00
For a total of 5 facts.
CLIPS> (agenda)
600    Increment-Node-Count: f-6,f-4
500    Add-Two-Gallons: f-5,f-1
100    update-depth: f-1,f-4
For a total of 3 activations.
```

```
CLIPS> (run 1)
CLIPS> (run 1)
CLIPS> (run 1)
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-1     (depth 0 0) CF 1.00
f-5     (status (water-jug 0) (depth 0) (node 1)
                (parent no-parent) (operation "")) CF 1.00
f-8     (status (water-jug 1) (depth 1) (node 2)
                (parent <Fact-5>) (operation "Add One Gallon")) CF 1.00
f-10    (nodes 4) CF 1.00
f-11    (status (water-jug 2) (depth 1) (node 3)
                (parent <Fact-5>) (operation "Add Two Gallons")) CF 1.00
For a total of 6 facts.
CLIPS> agenda)
agenda
CLIPS> (agenda)
100    update-depth: f-1,f-10
For a total of 1 activation.
```

Once Again Update Depth

```
;------------------------------------------------------------------------------------------
;  Rule to update current depth
;    Called only if depth is done for all rules for this level
;    (hence the lower salience)
;
(defrule update-depth
        (declare (salience 100))
        ?depth <- (depth ?current ?last)
        ?node <- (nodes ?nodes)
        (test (> ?nodes ?last))
=>
        (retract ?depth)
        (assert (depth (+ 1 ?current) ?nodes))
        )
```

A Solution Was Encountered

```
CLIPS> (run 1)
CLIPS> (run 1)
CLIPS> (run 1)
CLIPS> (run 1)
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-5     (status (water-jug 0) (depth 0) (node 1)
                (parent no-parent) (operation "")) CF 1.00
f-8     (status (water-jug 1) (depth 1) (node 2)
                (parent <Fact-5>) (operation "Add One Gallon")) CF 1.00
f-11    (status (water-jug 2) (depth 1) (node 3)
                (parent <Fact-5>) (operation "Add Two Gallons")) CF 1.00
f-12    (depth 1 4) CF 1.00
f-14    (nodes 5) CF 1.00
f-15    (status (water-jug 2) (depth 2) (node 4)
                (parent <Fact-8>) (operation "Add One Gallon")) CF 1.00
f-16    (status (water-jug 3) (depth 2) (node 0)
                (parent <Fact-11>) (operation "Add One Gallon")) CF 1.00
For a total of 8 facts.
CLIPS> (agenda)
1000   Done: f-16
600    Increment-Node-Count: f-16,f-14
500    Add-Two-Gallons: f-8,f-12
500    Add-Two-Gallons: f-11,f-12
100    update-depth: f-12,f-14
For a total of 5 activations.
```

Detect A Solution

```
(defrule Done
        (declare (salience 1000))
        ?status <- (status
                        (water-jug 3)
                        (operation ?move))
=>
        (assert (complete ?status))
        (assert (list "Done")))

CLIPS> (run 1)
CLIPS> (agenda)
2000   Complete: f-16,f-17,f-18
600    Increment-Node-Count: f-16,f-14
500    Add-Two-Gallons: f-8,f-12
500    Add-Two-Gallons: f-11,f-12
100    update-depth: f-12,f-14
For a total of 5 activations.
```

Build Up Solution

```
(defrule Complete
        (declare (salience 2000))
        ?status <- (status (parent ?parent) (operation ?move))
        ?complete <- (complete ?status)
        ?old-list <- (list $?rest)
=>
        (assert (list ?move ?rest))
        (assert (complete ?parent))
        (retract ?complete)
        (retract ?old-list)
        )
```

Solution Built Up

```
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-5     (status (water-jug 0) (depth 0) (node 1)
                (parent no-parent) (operation "")) CF 1.00
f-8     (status (water-jug 1) (depth 1) (node 2)
                (parent <Fact-5>) (operation "Add One Gallon")) CF 1.00
f-11    (status (water-jug 2) (depth 1) (node 3)
                (parent <Fact-5>) (operation "Add Two Gallons")) CF 1.00
f-12    (depth 1 4) CF 1.00
f-14    (nodes 5) CF 1.00
f-15    (status (water-jug 2) (depth 2) (node 4)
                (parent <Fact-8>) (operation "Add One Gallon")) CF 1.00
f-16    (status (water-jug 3) (depth 2) (node 0)
                (parent <Fact-11>) (operation "Add One Gallon")) CF 1.00
f-23    (list "" "Add Two Gallons" "Add One Gallon" "Done") CF 1.00
f-24    (complete no-parent) CF 1.00
For a total of 10 facts.
CLIPS> (agenda)
2000   Done-Complete: f-24,f-23
600    Increment-Node-Count: f-16,f-14
500    Add-Two-Gallons: f-8,f-12
500    Add-Two-Gallons: f-11,f-12
100    update-depth: f-12,f-14
For a total of 5 activations.
```

Print Out Answer

```
(defrule Done-Complete
        (declare (salience 2000))
        ?complete <- (complete no-parent)
        ?old-list <- (list ?first $?moves ?last)
=>
        (printout t "Solution:" crlf ?moves crlf)
        (retract ?complete ?old-list))
```

The Solutions

```
CLIPS> (run 1)
Solution:
("Add Two Gallons" "Add One Gallon")
Solution:
("Add Two Gallons" "Add One Gallon")
Solution:
("Add One Gallon" "Add Two Gallons")
Solution:
("Add One Gallon" "Add Two Gallons")
Solution:
("Add One Gallon" "Add One Gallon" "Add One Gallon")
Solution:
("Add One Gallon" "Add One Gallon" "Add One Gallon")
```

The Nodes of the Tree

```
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-5     (status (water-jug 0) (depth 0) (node 1)
                (parent no-parent) (operation "")) CF 1.00
f-8     (status (water-jug 1) (depth 1) (node 2)
                (parent <Fact-5>) (operation "Add One Gallon")) CF 1.00
f-11    (status (water-jug 2) (depth 1) (node 3)
                (parent <Fact-5>) (operation "Add Two Gallons")) CF 1.00
f-15    (status (water-jug 2) (depth 2) (node 4)
                (parent <Fact-8>) (operation "Add One Gallon")) CF 1.00
f-26    (status (water-jug 3) (depth 2) (node 5)
                (parent <Fact-11>) (operation "Add One Gallon")) CF 1.00
f-45    (status (water-jug 3) (depth 2) (node 6)
                (parent <Fact-8>) (operation "Add Two Gallons")) CF 1.00
f-56    (status (water-jug 4) (depth 2) (node 7)
                (parent <Fact-11>) (operation "Add Two Gallons")) CF 1.00
f-70    (status (water-jug 3) (depth 3) (node 8)
                (parent <Fact-15>) (operation "Add One Gallon")) CF 1.00
f-82    (nodes 10) CF 1.00
f-83    (status (water-jug 4) (depth 3) (node 9)
                (parent <Fact-15>) (operation "Add Two Gallons")) CF 1.00
f-84    (depth 3 10) CF 1.00
For a total of 12 facts.
```

When No More Nodes Added In Depth: Halt

```
(defrule finished
        (declare (salience 100))
        ?depth <- (depth ?current ?last)
        ?node <- (nodes ?nodes)
        (test (= ?nodes ?last))
=>
        (printout t "A Total of " (- ?nodes 1) " Nodes in the Tree" crlf)
        (retract ?depth)
        (halt)
        )
```

```
CLIPS> (agenda)
100    finished: f-84,f-82
For a total of 1 activation.
CLIPS> (run 1)
A Total of 9 Nodes in the Tree
CLIPS> (agenda)
```