COMP3608 Assignment 2

Tuesday, March 26, 2019 10:07 PM

816000772
COMP3608
Assignment 2

Christopher Sahadeo

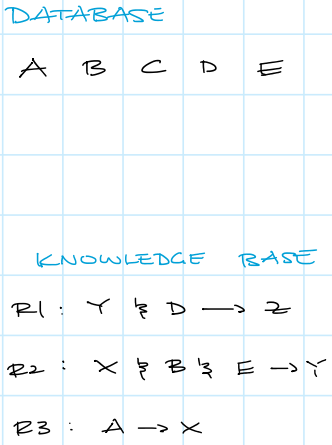
Question 1 Part b

Tuesday, March 26, 2019 1:08 AM

Note: I substituted the (watch activations) statement for the (agenda) statement in order to see the activations for each cycle

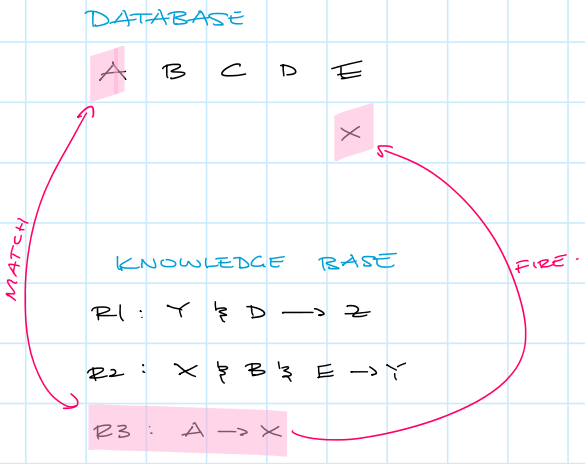
Dialog Window

```
CLIPS> (load "D:/DELL STORAGE/UWI/18-19-S2-Intelligent-Systems-
Code/Assignments/Assignment_2/Fig5.clp")
Defining defrule: R1 +j+j+j
Defining defrule: R2 +j+j+j+j
Defining defrule: R3 +j+j

CLIPS> (watch activations)

CLIPS> (assert (A TRUE) (B TRUE) (C TRUE) (D TRUE) (E TRUE))

==> Activation 0      R3: f-1
<Fact-5>

CLIPS> (run)
==> Activation 0      R2: f-6,f-2,f-5
X is true
==> Activation 0      R1: f-7,f-4
Y is true
Z is true
```

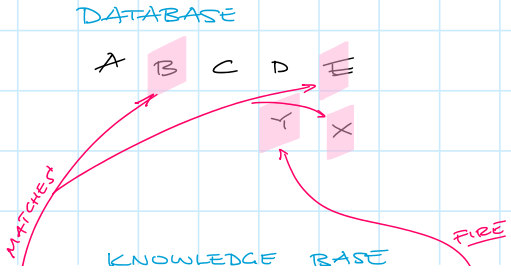


Inference Process Explanation

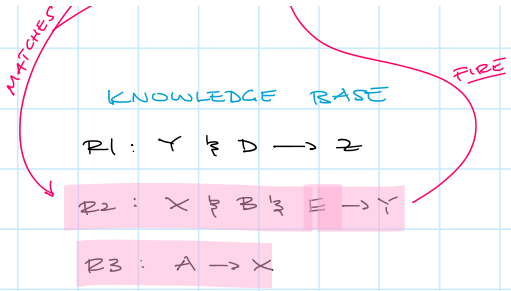
CLIPS uses the forward chaining technique to infer the fact Z, starting with the bottom most rule.

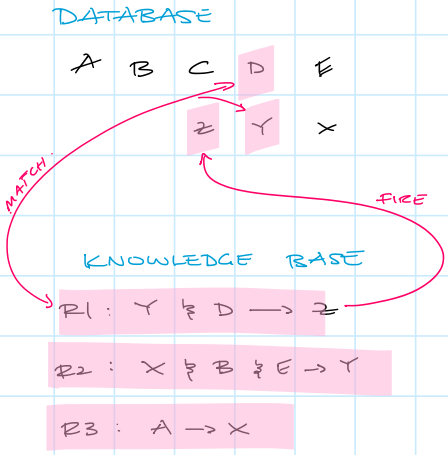


In the first cycle, only Rule 3 matches a fact in the database.
Rule 3 is fired.
The IF part of this rule matches fact A in the database.
Its THEN part is executed and the new fact X is added to the database.

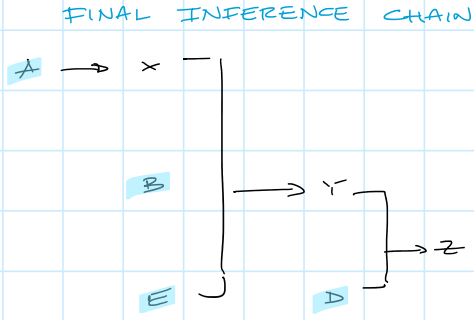


In the second cycle, Rule 2 is fired because the facts B, E and X are already in the database.
Fact Y Is inferred and put in the database.





In the third cycle, Rule 1 is fired since the facts Y and D are in the database. Fact Z is then inferred and put in the database



Question 1 Part d

Tuesday, March 26, 2019 2:38 AM

SWI Prolog Console

```
?- trace.
true.


[trace]  ?- r1(Z).
   Call: (8) r1(_3036) ? Creep % goal: Z
   Call: (9) r2(_3248) ? Creep  % subgoal: Y
   Call: (10) r3(_3248) ? Creep % subgoal: X
   Call: (11) f1(_3248) ? Creep % f1(A)
   Exit: (11) f1(_3248) ? Creep % f1(A)
   Call: (11) writeln("R3: Asserted X") ? creep
R3: Asserted X
   Exit: (11) writeln("R3: Asserted X") ? creep
^  Call: (11) assertz(f1(_3242)) ? creep
^  Exit: (11) assertz(f1(_3242)) ? Creep
   Exit: (10) r3(_3242) ? Creep % subgoal X is obtained
   Call: (10) f1(_3260) ? Creep % B
   Exit: (10) f1(_3260) ? Creep % B
   Call: (10) f1(_3260) ? Creep % E
   Exit: (10) f1(_3260) ? Creep % E
   Call: (10) writeln("R2: Asserted Y") ? Creep
R2: Asserted Y
   Exit: (10) writeln("R2: Asserted Y") ? creep
^  Call: (10) assertz(f1(_3254)) ? creep
^  Exit: (10) assertz(f1(_3254)) ? creep
   Exit: (9) r2(_3254) ? Creep % subgoal Y is obtained
   Call: (9) f1(_3272) ? Creep % D
   Exit: (9) f1(_3272) ? Creep % D
   Call: (9) writeln("R1: Asserted Z") ? creep
R1: Asserted Z
   Exit: (9) writeln("R1: Asserted Z") ? creep
^  Call: (9) assertz(f1(_3036)) ? creep
^  Exit: (9) assertz(f1(_3036)) ? creep
   Exit: (8) r1(_3036) ? Creep % goal Z is obtained
true .
```

DATABASE

A B C D E

KNOWLEDGE BASE

R1 : Y ∧ D → Z

R2 : X ∧ B ∧ E → Y

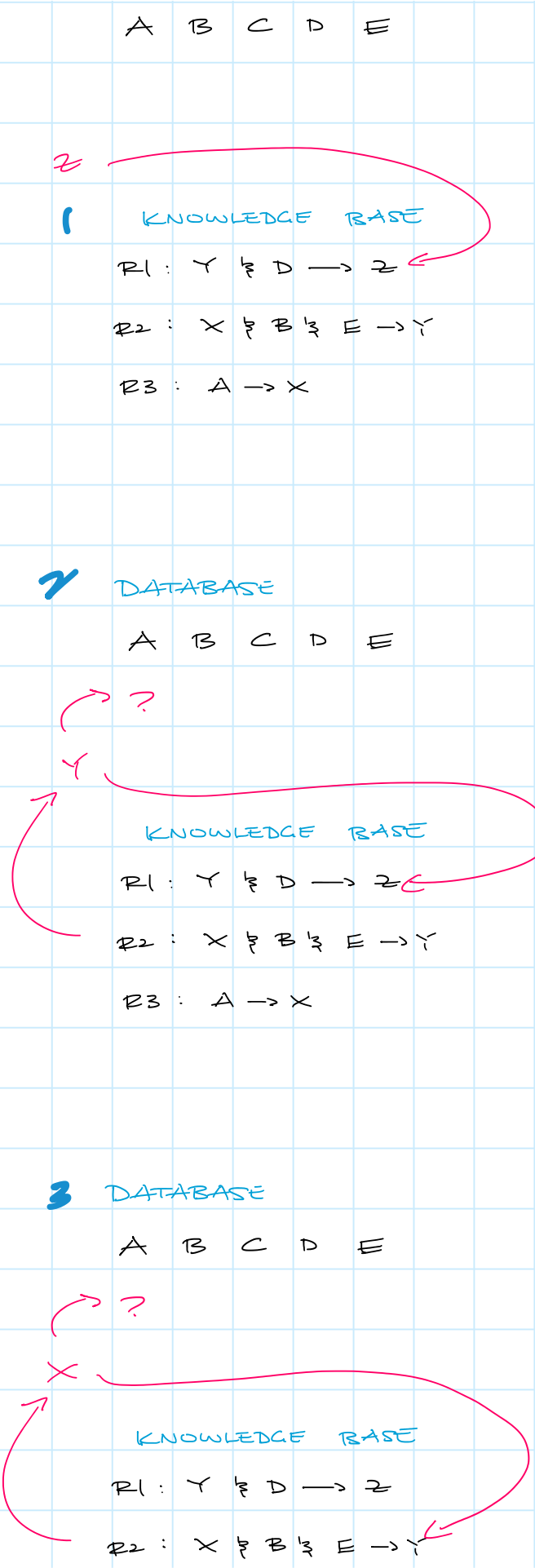
R3 : A → X

Inference Process Explanation

Prolog uses backward chaining to infer the fact Z

When more than one rule/fact with the same functor is present in the database, they are tried in the order that they appear in the database.

DATABASE

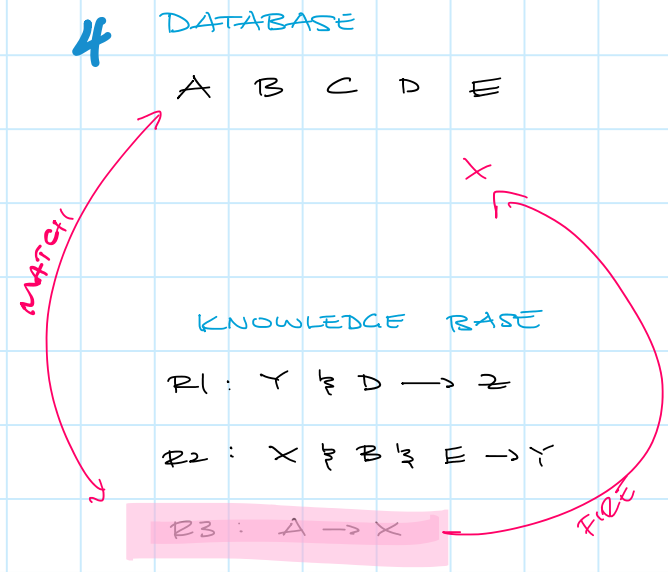


Pass 1:
Inference engine attempts to infer fact Z
It searches the knowledge base to find the rule that has fact Z in its THEN part
The engines finds and stacks r1
The IF part of r1 (Y and D) must be established

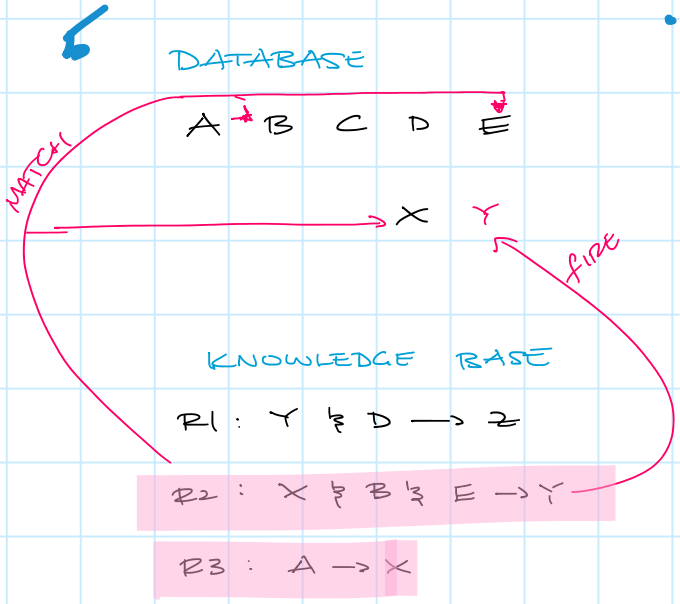
Pass 2
Inference engine sets up the subgoal Y
It checks the database but the fact Y is not there
The knowledge base is searched again for a rule with Y in its then part.
The engine located and stacks R2

Pass 3
The inference engine sets up the subgoal X
It checks the database for fact X and fails to find it
It searches for the rule that infer X (call this r3)

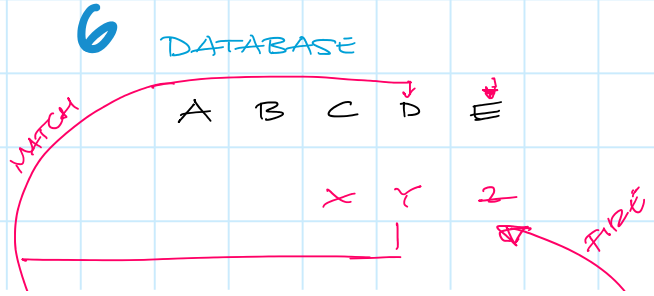
R3 : A → X



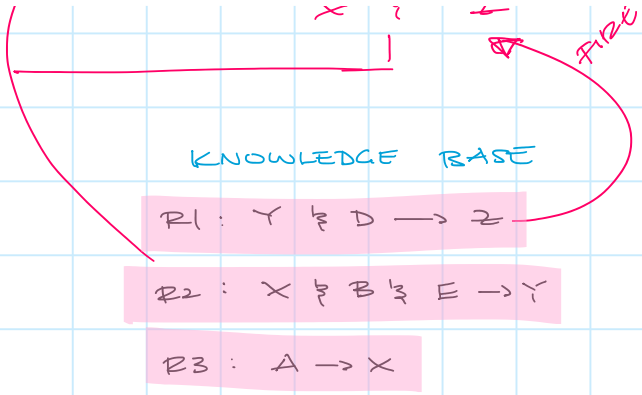
Pass 4
Fact A is found and r3 is fired
X is asserted into the database



Pass 5
r2 matches and fires
Y is added to the database



Pass 6
The inference engine then returns to rule 1
The IF part in rule 1 matches all the facts in the database
Rule 1 is executed and Z (final goal) is asserted into the database



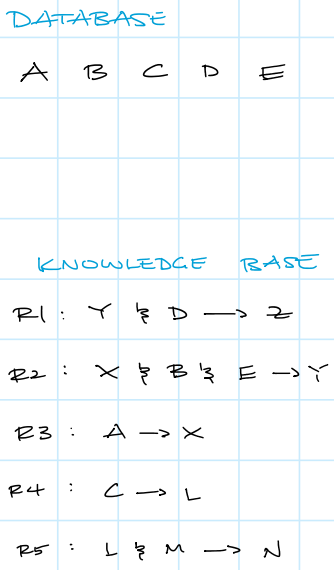
Rule 1 is executed and Z (final goal) is asserted into the database

Question 2 Part b

Tuesday, March 26, 2019 1:45 AM

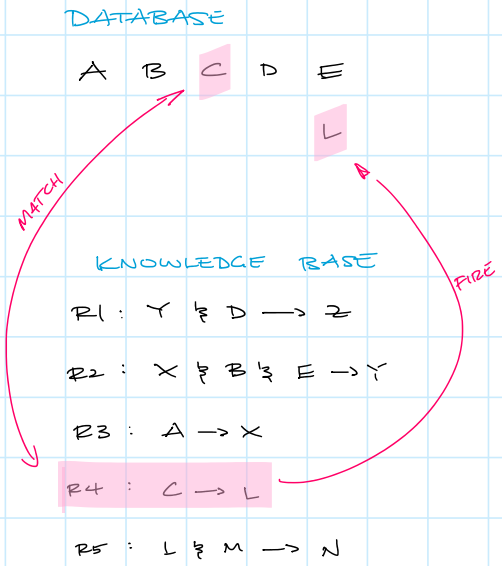
Dialogue Window

```
CLIPS> (load "D:/DELL STORAGE/UWI/18-19-S2-Intelligent-Systems-
Code/Assignments/Assignment_2/Fig6.clp")
Defining defrule: R1 +j+j+j
Defining defrule: R2 +j+j+j+j
Defining defrule: R3 +j+j
Defining defrule: R4 +j+j
Defining defrule: R5 +j+j+j
TRUE

CLIPS> (assert (A TRUE) (B TRUE) (C TRUE) (D TRUE) (E TRUE))
==> Activation 0      R3: f-1
==> Activation 0      R4: f-3
<Fact-5>

CLIPS> (run)
L is true
==> Activation 0      R2: f-7,f-2,f-5
X is true
==> Activation 0      R1: f-8,f-4
Y is true
Z is true
```

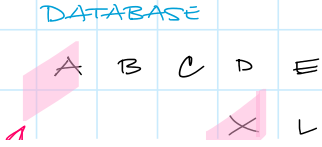


Inference Process Explanation

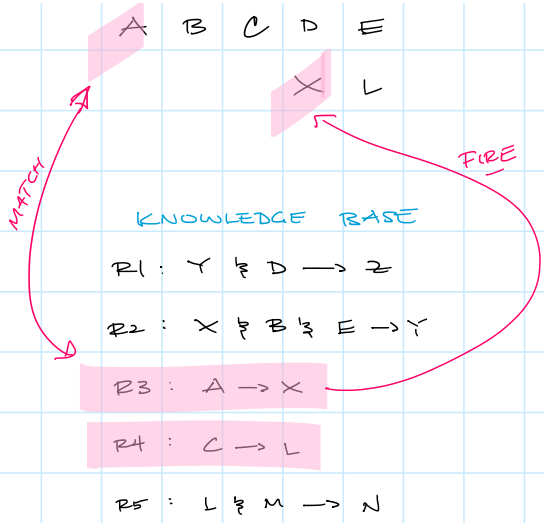
CLIPS uses the forward chaining technique to infer the fact Z, starting with the bottom most rule.



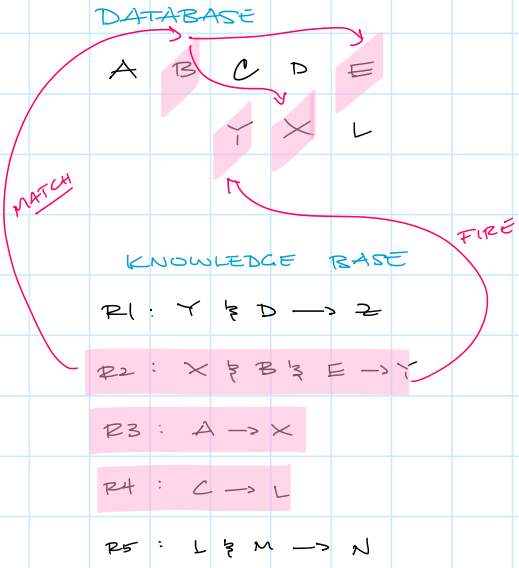
In the first cycle, both R3 and R4 match facts in the database.
They are both activated.
R4 fires first.
The IF part of R4 matches the fact C in the database
Its THEN part is executed and the new fact L is added to the database



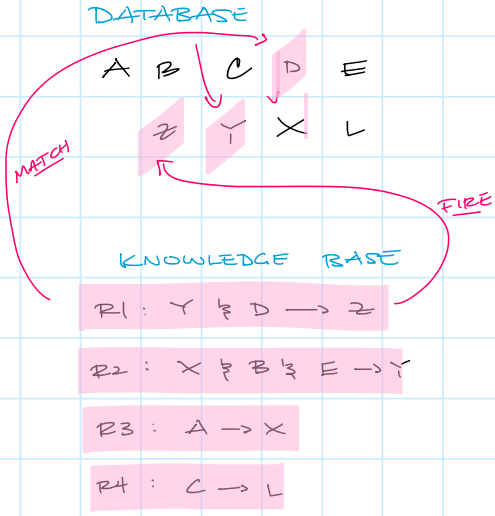
Still in cycle 1, R3 is the next to fire.
The IF part of R3 matches the fact A in the database
Its THEN part is executed and the new fact X is added to the database
The addition of X activates R2



Still in cycle 1, R3 is the next to fire.
The IF part of R3 matches the fact A in the database
Its THEN part is executed and the new fact X is added to the database
The addition of X activates R2



In cycle 2, R2 is the next to fire.
The IF part of R2 matches the facts X, B and E in the database
Its THEN part is executed and the new fact Y is added to the database
The addition of Y activates R1

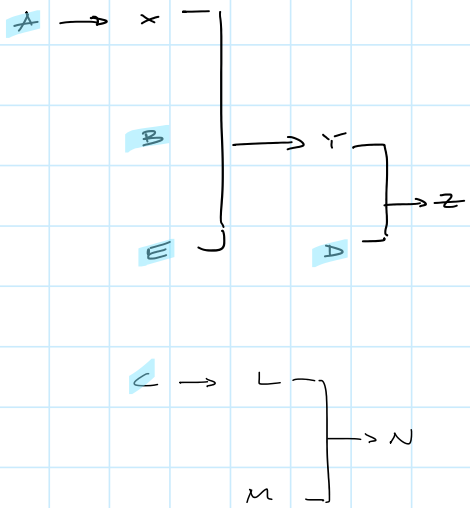


In cycle 3, R1 is the next to fire.
The IF part of R1 matches the facts Y and D in the database
Its THEN part is executed and the new fact Z is added to the database

The match-fire cycles stop because the IF part of R5 does not match all the facts in the database.

R5 : L & M → N

FINAL INFERENCE CHAIN.



Question 2 Part d

Tuesday, March 26, 2019 9:53 PM

SWI Prolog Console

Note: Backward chaining is a goal-driven technique. This means the process starts at the goal state Z and works it way backwards though the chain to find sub goals. This means that any rules that are not directly connected to the goal state is NEVER fired. For this reason, R4 and R5 are never fired in this scenario. This implies that the solution for Question 2 Part d is essentially the same as Question 1 Part d.

```
?- trace.
true.


[trace]  ?- r1(Z).
   Call: (8) r1(_3036) ? Creep % goal: Z
   Call: (9) r2(_3248) ? Creep  % subgoal: Y
   Call: (10) r3(_3248) ? Creep % subgoal: X
   Call: (11) f1(_3248) ? Creep % f1(A)
   Exit: (11) f1(_3248) ? Creep % f1(A)
   Call: (11) writeln("R3: Asserted X") ? creep
R3: Asserted X
   Exit: (11) writeln("R3: Asserted X") ? creep
^  Call: (11) assertz(f1(_3242)) ? creep
^  Exit: (11) assertz(f1(_3242)) ? Creep
   Exit: (10) r3(_3242) ? Creep % subgoal X is obtained
   Call: (10) f1(_3260) ? Creep % B
   Exit: (10) f1(_3260) ? Creep % B
   Call: (10) f1(_3260) ? Creep % E
   Exit: (10) f1(_3260) ? Creep % E
   Call: (10) writeln("R2: Asserted Y") ? Creep
R2: Asserted Y
   Exit: (10) writeln("R2: Asserted Y") ? creep
^  Call: (10) assertz(f1(_3254)) ? creep
^  Exit: (10) assertz(f1(_3254)) ? creep
   Exit: (9) r2(_3254) ? Creep % subgoal Y is obtained
   Call: (9) f1(_3272) ? Creep % D
   Exit: (9) f1(_3272) ? Creep % D
   Call: (9) writeln("R1: Asserted Z") ? creep
R1: Asserted Z
   Exit: (9) writeln("R1: Asserted Z") ? creep
^  Call: (9) assertz(f1(_3036)) ? creep
^  Exit: (9) assertz(f1(_3036)) ? creep
   Exit: (8) r1(_3036) ? Creep % goal Z is obtained
true .
```

Inference Process Explanation

Same explanation as Question 1 Part d.
The only difference is this version has the two extra rules (r4 and r5) that are never fired since they do not help in inferring the fact Z.