Rules and Inference

Creating Knowledge-based Systems
CS3025, Knowledge-Based Systems
Lecture 04

Yuting Zhao
yuting.zhao@gmail.com

Outline

- What is Knowledge?
- Representation of Knowledge
- Knowledge-based Systems:

  Knowledge + Inference
- ***Rules*** and ***Facts***
- ***Inference***

Knowledge for Human

Knowledge for AI / robot

Knowledge

– facts and how to process them

- Facts
  - Humans observe facts or data about the world
    - "The sky is blue"
    - "It is raining"
  - How do they process this information?
- Cognitive psychologists have used the concept of a "Rule" to explain human information processing
  - Humans store knowledge how to reason about these facts in the form of rules:
    - Most human problem solving or "cognition" can be expressed in the form of "**IF … THEN …"** sentences:
    - E.g.:

IF "it is raining" **THEN** "carry an umbrella"

IF "it is raining" **THEN** "sell umbrellas"

Human Cognition
Processing Knowledge

- A **rule** corresponds to a small modular collection of knowledge, also called a "**chunk**"
  - Think of a Chess master: may know
    Guess how many?
    or more chunks of knowledge / rules about chess patterns and how to react to certain situations
- Basic idea:
  - Sensory input (observations / facts) provides stimuli to the brain
  - These stimuli trigger the processing of rules that produce some response by a human being
    - New insights, that lead to the triggering of further rules
    - Some action by the human being
- This is a good model for automated reasoning in knowledge-based systems

Knowledge-Based Systems

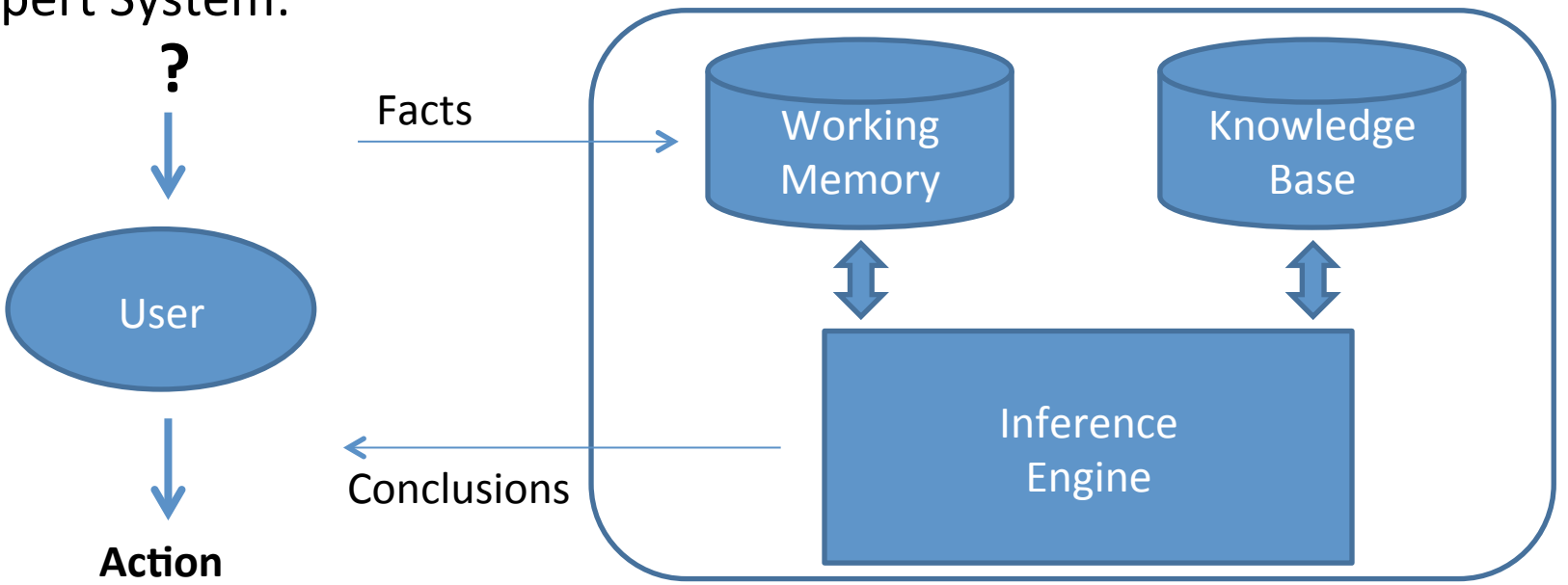
AKA Expert Systems

- What is a *Knowledge-based System*:
  - A software system that "emulates" a human expert in its decision making
    - It seems to ***reason*** like a human expert and draw (***infer***) **conclusions** from known ***facts*** using expert ***knowledge***
  - A software system that solves problems by using ***inference procedures*** to draw **conclusions** from a symbolic representation of ***knowledge***
- Also called an *Expert System*

Knowledge-Based Systems

Expert Systems

- Expert System:



Knowledge + Inference = Knowledge-based System

- Basic Concepts:
  - User provides facts to expert system
  - The inference engine draws conclusions based on given facts and domain-specific knowledge
  - User receives conclusions inferred by expert system

Knowledge Representation

- Knowledge expressed as IF ... THEN ... rules:

  - IF
    - **Premise, antecedent, left-hand side (LHS) is true**
  - THEN
    - **Conclusion, consequent, right-hand side (RHS) is true**
- E.g.:
  - "IF *Socrates is a man* THEN *Socrates is mortal*"

- Knowledge expressed as simple facts:
  - *"Socrates is a man"*

Inference
Reasoning about Knowledge

Knowledge Base = Set of sentences in a formal language

- Example Knowledge Base:
  - A Rule: "IF *Socrates is a man* THEN *Socrates is mortal*"
  - A Fact: "*Socrates is a man*"
- We create this Knowledge Base to *infer* additional knowledge:
  - It seems natural that we can conclude (or *infer*) from the content of this Knowledge Base:

    "*Socrates is mortal*"
- But how? (and, in particular how can software make such a conclusion?)

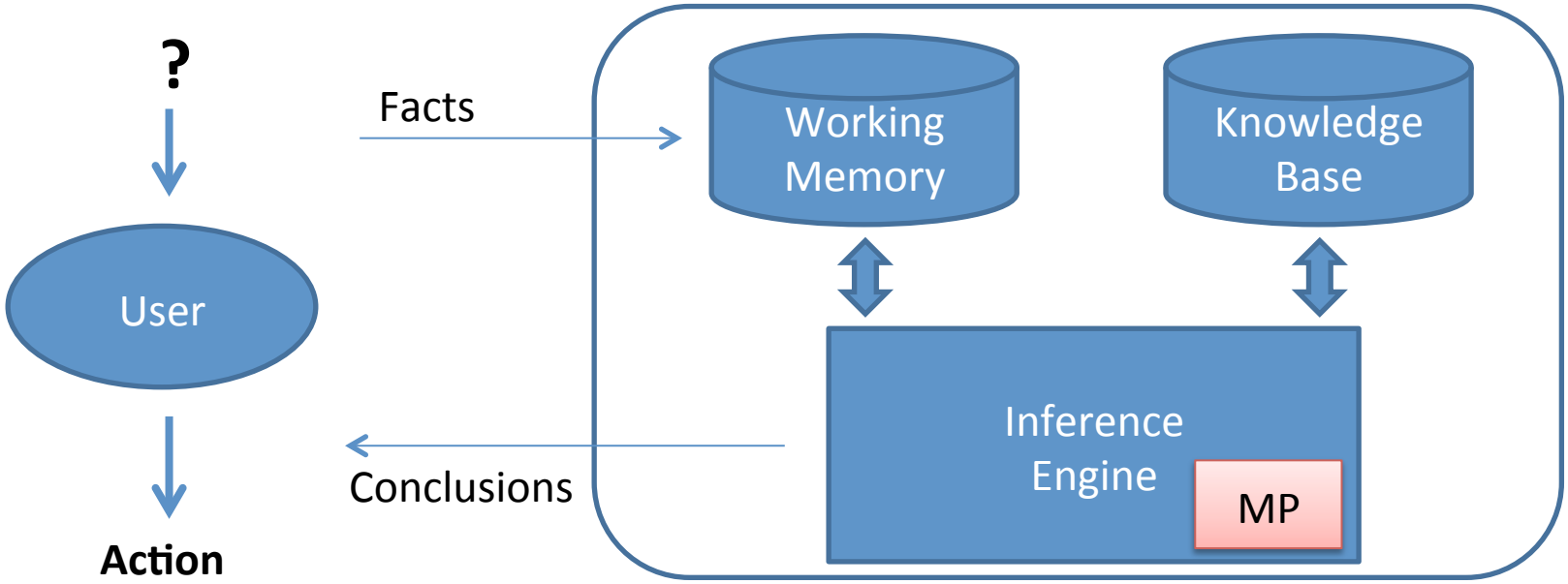
Inference: Modus Ponens (MP)

- Theorem from Propositional Logic:
  Modus Ponens (MP):

  From: if x then y
  x
  Infer: y

- Use Modus Ponens to make Inferences
  - Assume to be TRUE (Axioms):
    - "IF Socrates is a man THEN Socrates is mortal" (Rule)
    - "Socrates is a man" (Fact / Proposition)
  - Apply Modus Ponens / infer / conclude
    - "Socrates is mortal"
- This is an example of "valid inference" because given true axioms, the inference (conclusion) drawn will be also be true.
- Modus Ponens is the basis for implementing Inference Engines

Expert System Shell (like JESS)



- Expert System Shells are used to implement a Knowledge-based System
  - Separation between knowledge and inference engine
  - Knowledge represented as a set of rules (the production system)
  - Inference engine is the execution mechanism for production systems and manages rule activation, execution and the facts produced

Inference Engine and Production System

- Production System
  - A Production System is the set of rules specified by a knowledge engineer – it is the "Knowledge" of an Expert System, implemented by the knowledge engineer
  - A single rule is called a "production rule", it produces some "effect" (e.g. writing a new fact into working memory)
- Inference Engine
  - Is the execution environment for rules
    - Manages the activation of rules, the resolution of conflicts between them and the maintenance of the working memory

Specifying Rules

IF [**LHS** ANTECEDENT] **then** [**RHS** CONSEQUENT]

IF "the traffic light shows red"
THEN "wait 1min",
"switch on yellow light"

IF "the traffic light shows red **AND** yellow"
THEN "wait 3seconds",
"switch off red light",
"switch off yellow light",
"switch on green light"

- Constructing the Antecedent
  - use logical operators to combine propositions in antecedent and consequent of a rule
    - and (∧)
    - or (∨)
    - not (¬)
- Constructing the Consequent
  - Specify a sequence of actions

Rules: Conjunctive Antecedents

IF "the traffic light shows red
AND yellow"
THEN "wait 3seconds",
"switch off red light",
"switch off yellow light",
"switch on green light"

If $a_1 \wedge a_2 \wedge \ldots \wedge a_n$
then $c_1 \wedge c_2 \wedge \ldots \wedge c_m$

a_i is antecedent i
c_j is consequent j
"$\wedge$" represents "**AND**" (conjunction)

Pay fine in either case

- Examples:
  - **If** book is overdue **then** pay fine
  - **If** book is lost **then** pay fine **and** pay for replacement book
  - **If** book is due today **and** library is open **then** return book
- We can also split a rule with conjunctive consequents into two separate rules:
  - If book is lost then pay fine
  - If book is lost then pay for replacement book

Rules: Disjunctive Antecedents

IF $a_1 \vee a_2 \vee \ldots \vee a_n$

THEN $c_1 \wedge c_2 \wedge \ldots \wedge c_m$

a_i is antecedent i

c_j is consequent j

$\vee$ represents **OR** (disjunction)

- Examples:
  - **IF** book is overdue **OR** book is lost **THEN** pay fine
  - **IF** book is lost **THEN** pay for replacement book
- We can also split a rule with disjunctive antecedent into separate rules:
  - **IF** book is overdue **THEN** pay fine
  - **IF** book is lost **THEN** pay fine

Rules: General Form

IF $(a_{11} \wedge a_{12} \wedge \ldots \wedge a_{1b}) \vee$
$(a_{21} \wedge a_{22} \wedge \ldots \wedge a_{2c}) \vee$
…
$(a_{d1} \wedge a_{d2} \wedge \ldots \wedge a_{de})$

THEN $c_1 \wedge c_2 \wedge \ldots \wedge c_m$

- We can now write really complicated and difficult to understand rules!
- *Keep each rule as simple as possible.*

Inference

- How does automated reasoning / inference work?
  - Example: Traffic light control

IF "the traffic light shows red"
THEN "wait 1min",
"switch on yellow light"

IF "the traffic light shows green"
THEN "wait 1min",
"switch off green light",
"switch on yellow light"

IF "the traffic light shows red **AND** yellow"
THEN "wait 3seconds",
"switch off red light",
"switch off yellow light",
"switch on green light"

IF "the traffic light shows yellow **only**"
THEN "wait 3sec",
"switch off yellow light",
"switch on red light"

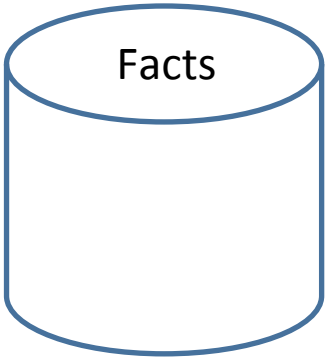
Inference

R1

IF "red"
THEN "wait 1min",
"add yellow"

R4

IF "yellow" **AND NOT** "red"
THEN "wait 3sec",
"remove yellow",
"add red"



R2

IF "red" **AND** "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"

- We use a shared "memory" containing facts
- Rules "look" into this memory whether their antecedent matches one or more of those facts
- A rule with a matching antecedent will perform actions as a consequence
- How to start?
  - Have to add an initial fact

R3

IF "green"
THEN "wait 1min",
"remove green",
"add yellow"

Inference

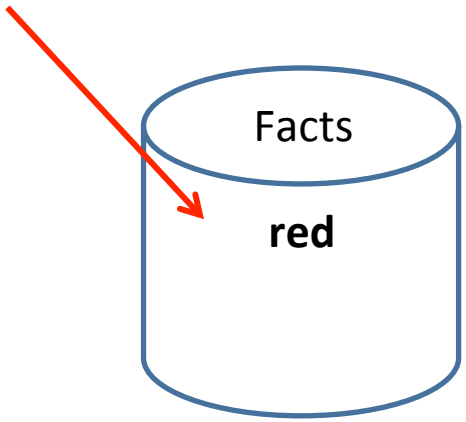
1. **Initialisation**: Manually write the first fact into our memory

R1

IF "red"
THEN "wait 1min",
"add yellow"

R4

IF "yellow" **AND NOT** "red"
THEN "wait 3sec",
"remove yellow",
"add red"



R2

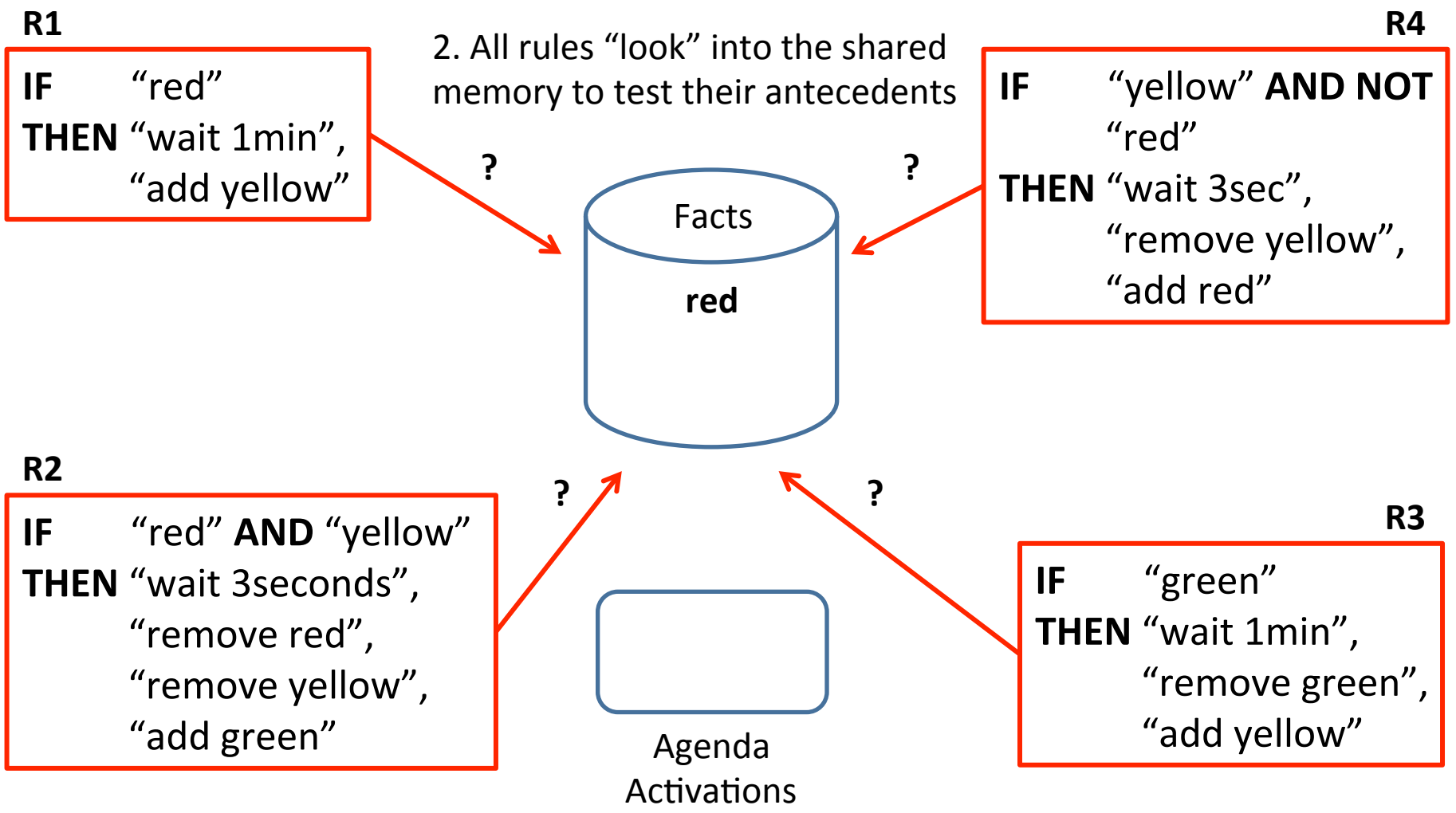
IF "red" **AND** "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"

Agenda
Activations

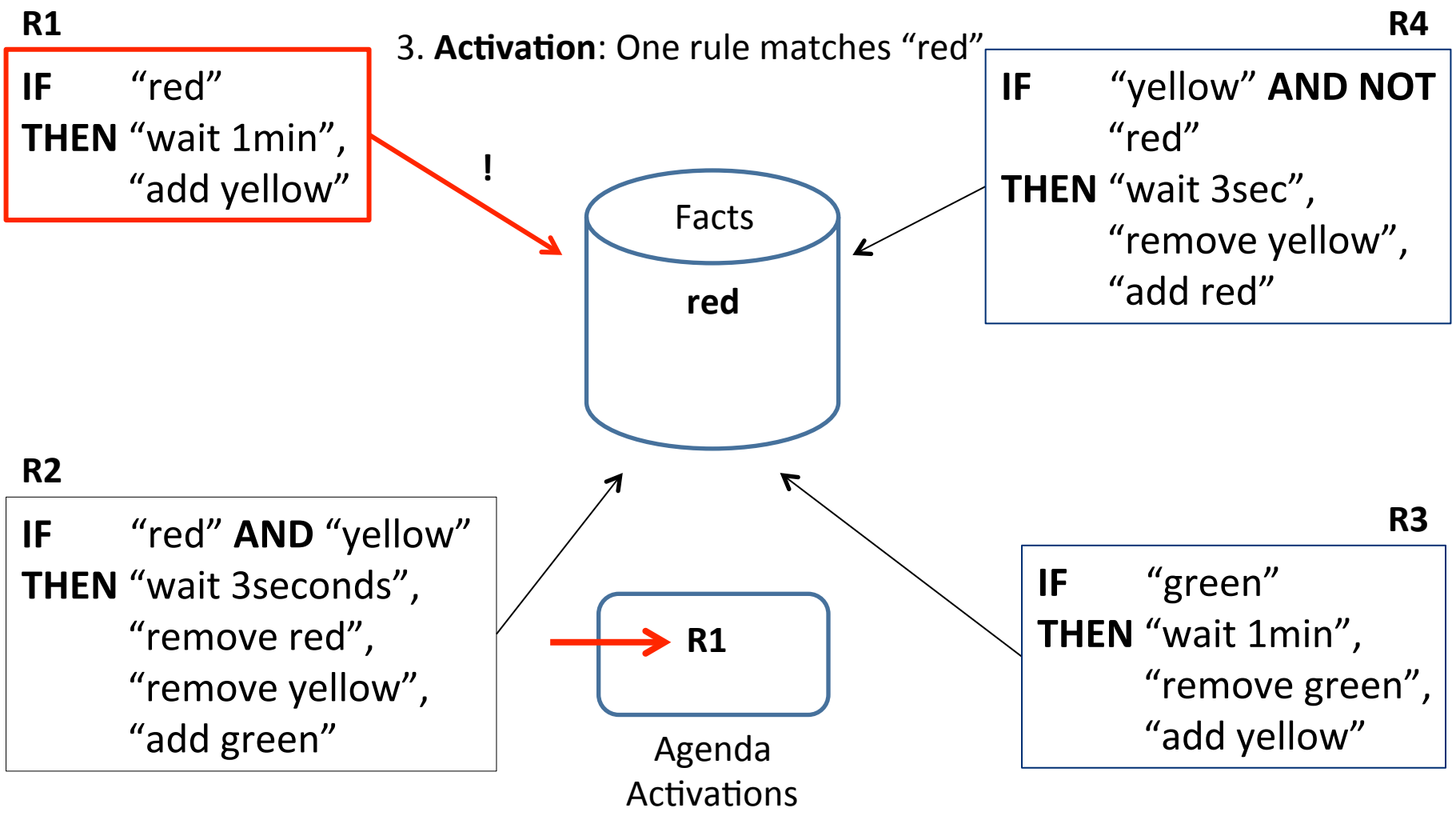
R3

IF "green"
THEN "wait 1min",
"remove green",
"add yellow"

Inference


R1
IF "red"
THEN "wait 1min",
"add yellow"
2. All rules "look" into the shared memory to test their antecedents
?
?
Facts
red
R4
IF "yellow" AND NOT "red"
THEN "wait 3sec",
"remove yellow",
"add red"
R2
IF "red" AND "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"
?
?
Agenda
Activations
R3
IF "green"
THEN "wait 1min",
"remove green",
"add yellow"

Inference

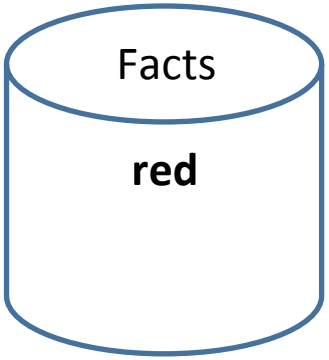

R1
IF "red"
THEN "wait 1min",
"add yellow"
3. Activation: One rule matches "red"
!
Facts
red
R4
IF "yellow" AND NOT
"red"
THEN "wait 3sec",
"remove yellow",
"add red"
R2
IF "red" AND "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"
R1
Agenda
Activations
R3
IF "green"
THEN "wait 1min",
"remove green",
"add yellow"

Inference

R1

IF "red"
THEN "**wait 1min**",
"add yellow"

4a. **Make "conclusions"**: perform action

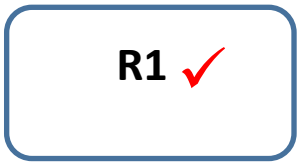


R4

IF "yellow" **AND NOT** "red"
THEN "wait 3sec",
"remove yellow",
"add red"

R2

IF "red" **AND** "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"



Agenda Activations

R3

IF "green"
THEN "wait 1min",
"remove green",
"add yellow"

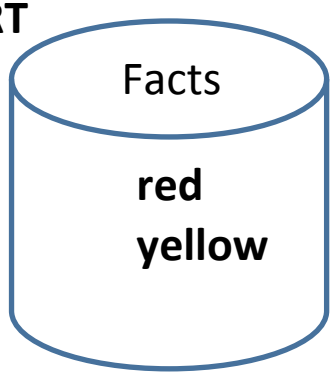
Inference

R1

IF "red"
THEN "wait 1min",
"**add yellow**"

4b. **Make "conclusions"**: adds a new fact "yellow" to shared memory

ASSERT

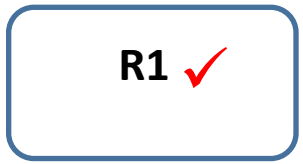


R4

IF "yellow" **AND NOT** "red"
THEN "wait 3sec",
"remove yellow",
"add red"

R2

IF "red" **AND** "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"

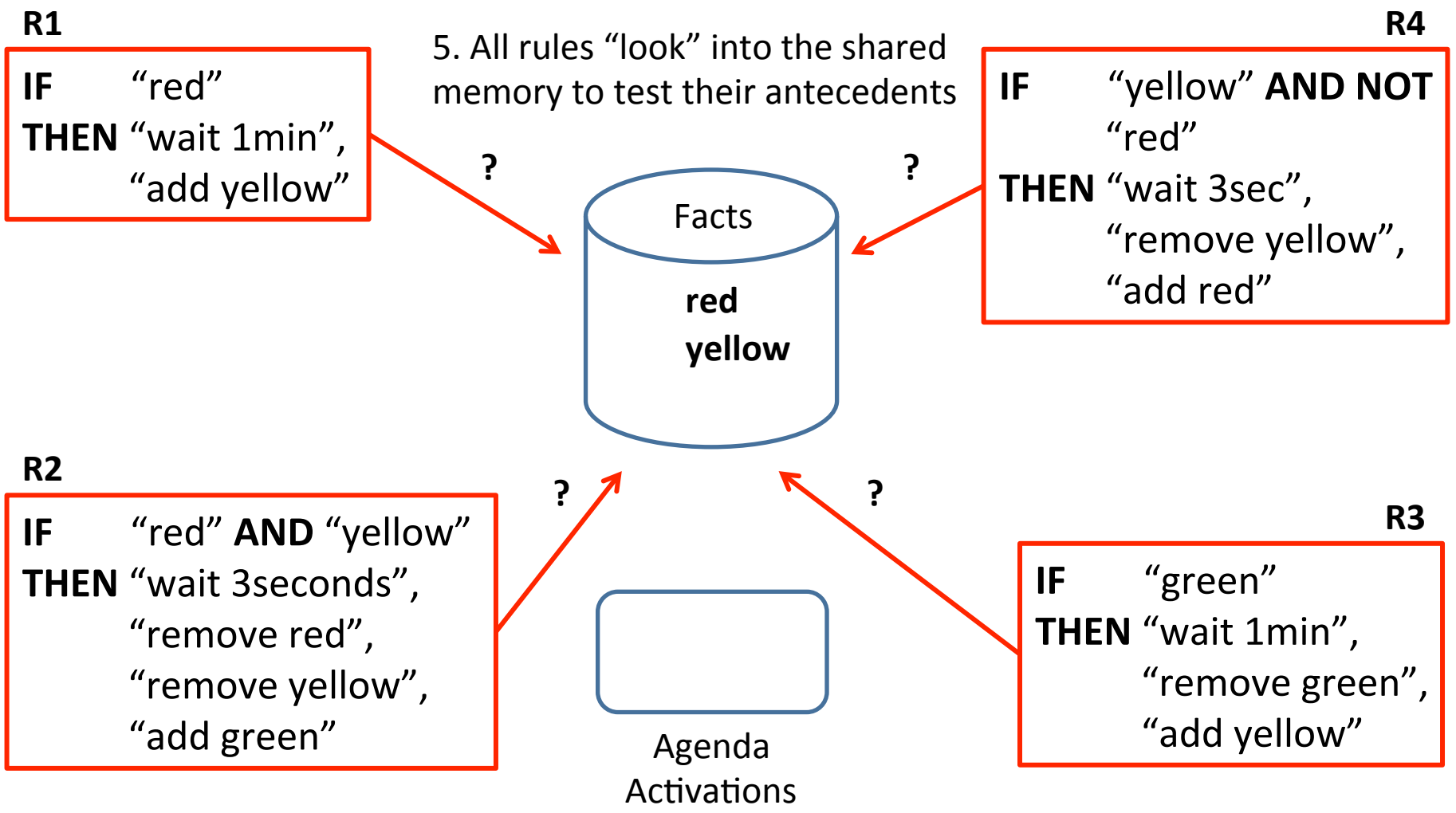


Agenda Activations

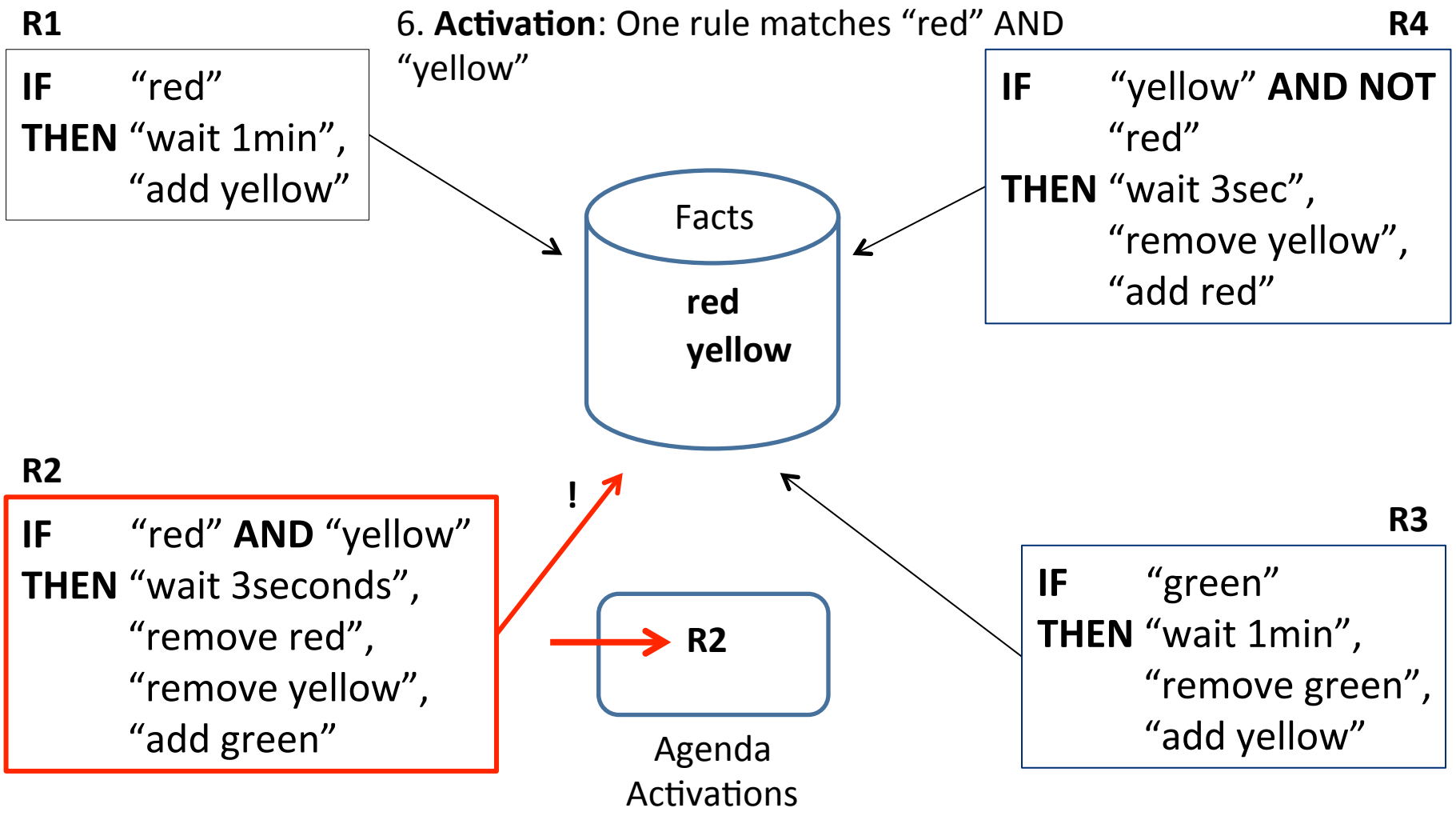
R3

IF "green"
THEN "wait 1min",
"remove green",
"add yellow"

Inference


R1
IF "red"
THEN "wait 1min",
"add yellow"
5. All rules "look" into the shared memory to test their antecedents
R4
IF "yellow" AND NOT "red"
THEN "wait 3sec",
"remove yellow",
"add red"
?
?
Facts
red
yellow
R2
IF "red" AND "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"
?
?
R3
IF "green"
THEN "wait 1min",
"remove green",
"add yellow"
Agenda
Activations

Inference

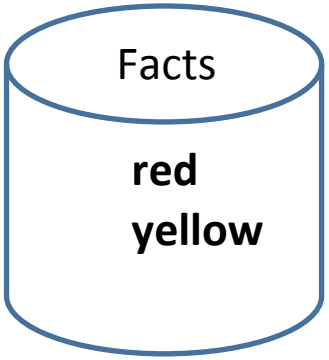

R1
IF "red"
THEN "wait 1min",
"add yellow"
6. **Activation**: One rule matches "red" AND "yellow"
R4
IF "yellow" **AND NOT** "red"
THEN "wait 3sec",
"remove yellow",
"add red"
Facts
red
yellow
R2
IF "red" **AND** "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"
!
R2
Agenda
Activations
R3
IF "green"
THEN "wait 1min",
"remove green",
"add yellow"

Inference

R1

IF "red"
THEN "wait 1min",
"add yellow"

7a. **Make "conclusions"**: perform action

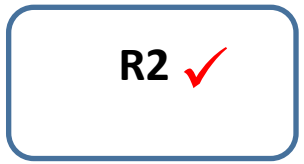


R4

IF "yellow" **AND NOT** "red"
THEN "wait 3sec",
"remove yellow",
"add red"

R2

IF "red" **AND** "yellow"
THEN "**wait 3seconds**",
"remove red",
"remove yellow",
"add green"



Agenda Activations

R3

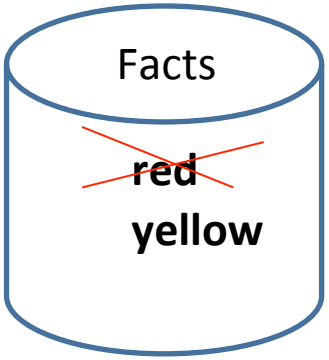
IF "green"
THEN "wait 1min",
"remove green",
"add yellow"

Inference

R1

IF "red"
THEN "wait 1min",
"add yellow"

7b. **Make "conclusions"**: remove fact "red" from shared memory



R4

IF "yellow" **AND NOT** "red"
THEN "wait 3sec",
"remove yellow",
"add red"

R2

IF "red" **AND** "yellow"
THEN "wait 3seconds",
"**remove red**",
"remove yellow",
"add green"

REMOVE

R2 ✓

Agenda Activations

R3

IF "green"
THEN "wait 1min",
"remove green",
"add yellow"

Inference

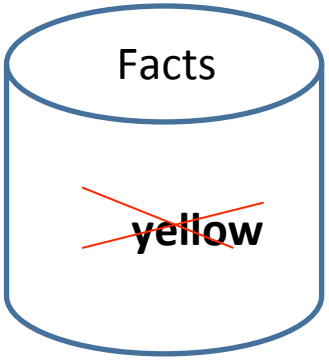
R1

IF "red"
THEN "wait 1min",
"add yellow"

7c. **Make "conclusions"**: remove fact "yellow" from shared memory

R4

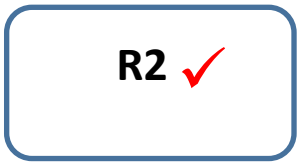
IF "yellow" **AND NOT** "red"
THEN "wait 3sec",
"remove yellow",
"add red"



R2

IF "red" **AND** "yellow"
THEN "wait 3seconds",
"remove red",
"**remove yellow**",
"add green"

REMOVE



Agenda Activations

R3

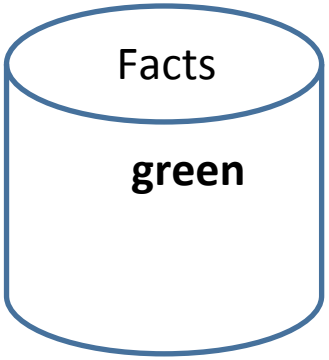
IF "green"
THEN "wait 1min",
"remove green",
"add yellow"

Inference

R1

IF "red"
THEN "wait 1min",
"add yellow"

7d. **Make "conclusions"**: add fact "green" to shared memory



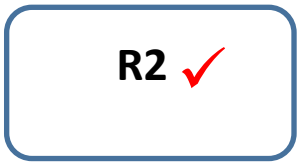
R4

IF "yellow" **AND NOT** "red"
THEN "wait 3sec",
"remove yellow",
"add red"

R2

IF "red" **AND** "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"**add green**"

ASSERT

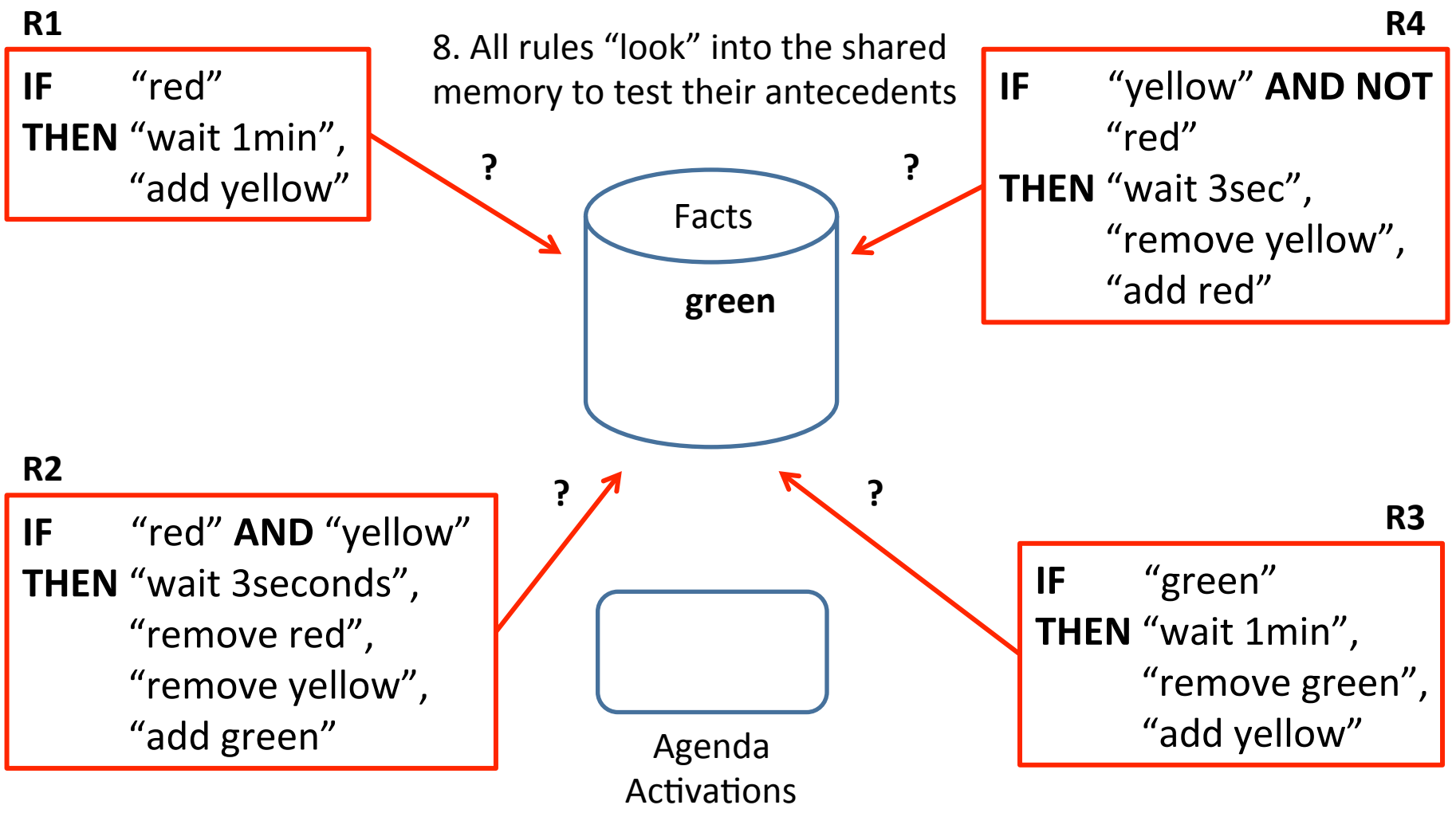


Agenda Activations

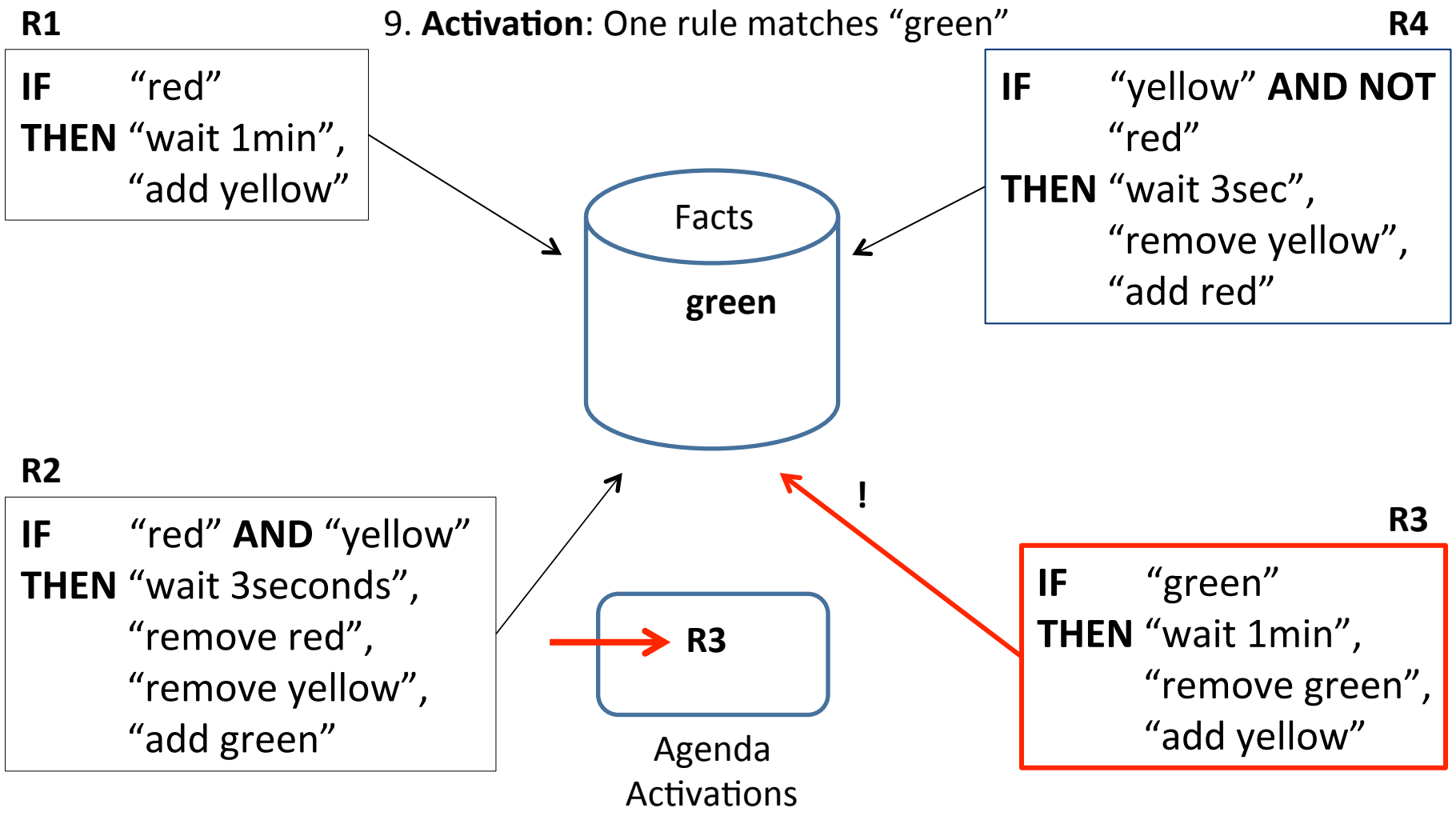
R3

IF "green"
THEN "wait 1min",
"remove green",
"add yellow"

Inference


R1
IF "red"
THEN "wait 1min",
"add yellow"
8. All rules "look" into the shared memory to test their antecedents
R4
IF "yellow" AND NOT "red"
THEN "wait 3sec",
"remove yellow",
"add red"
?
?
Facts
green
R2
IF "red" AND "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"
?
?
R3
IF "green"
THEN "wait 1min",
"remove green",
"add yellow"
Agenda
Activations

Inference

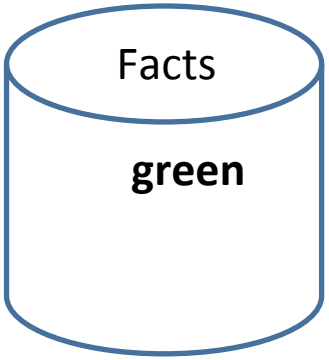

R1
IF "red"
THEN "wait 1min",
"add yellow"
9. **Activation**: One rule matches "green"
R4
IF "yellow" **AND NOT**
"red"
THEN "wait 3sec",
"remove yellow",
"add red"
Facts
green
R2
IF "red" **AND** "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"
!
R3
IF "green"
THEN "wait 1min",
"remove green",
"add yellow"
R3
Agenda
Activations

Inference

R1

IF "red"
THEN "wait 1min",
"add yellow"

10a. **Make "conclusions"**: perform action



R4

IF "yellow" **AND NOT** "red"
THEN "wait 3sec",
"remove yellow",
"add red"

R2

IF "red" **AND** "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"

Agenda Activations

R3

IF "green"
THEN "**wait 1min**",
"remove green",
"add yellow"

Inference

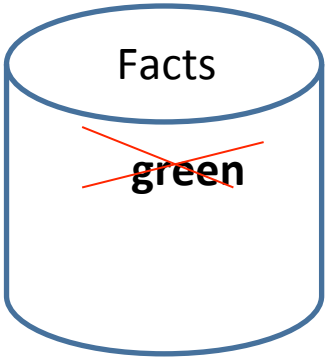
R1

IF "red"
THEN "wait 1min",
"add yellow"

10b. **Make "conclusions"**: remove fact "green" from shared memory

R4

IF "yellow" **AND NOT** "red"
THEN "wait 3sec",
"remove yellow",
"add red"



R2

IF "red" **AND** "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"

REMOVE

R3

IF "green"
THEN "wait 1min",
"**remove green**",
"add yellow"

Agenda
Activations

Inference

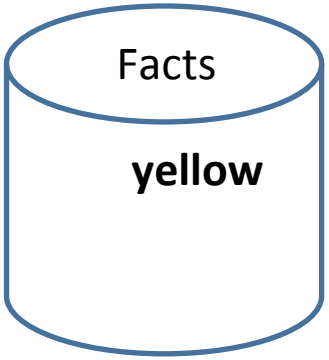
R1

IF "red"
THEN "wait 1min",
"add yellow"

10c. **Make "conclusions"**: add fact "yellow" to shared memory

R4

IF "yellow" **AND NOT** "red"
THEN "wait 3sec",
"remove yellow",
"add red"



R2

IF "red" **AND** "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"

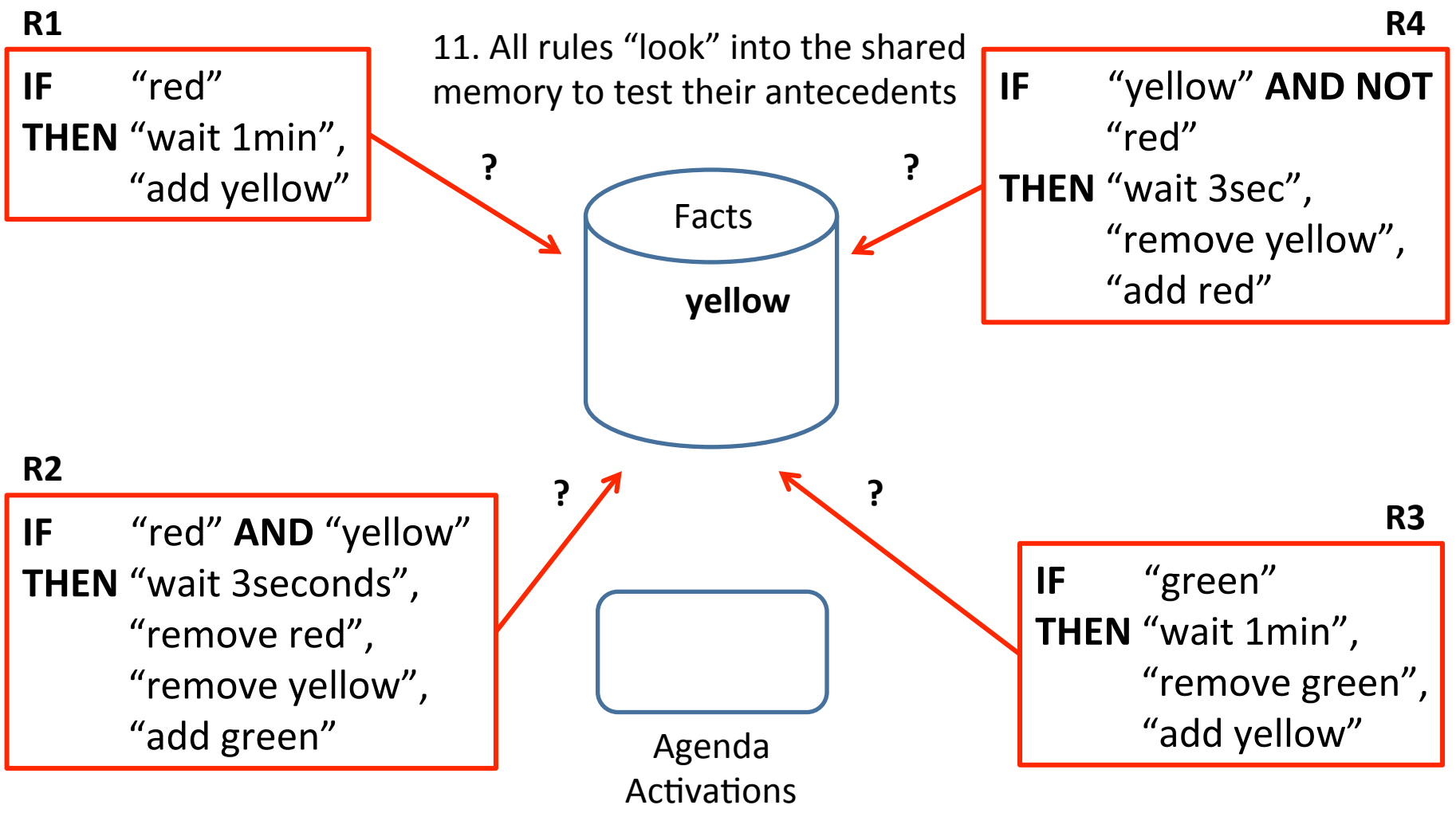
ASSERT

R3

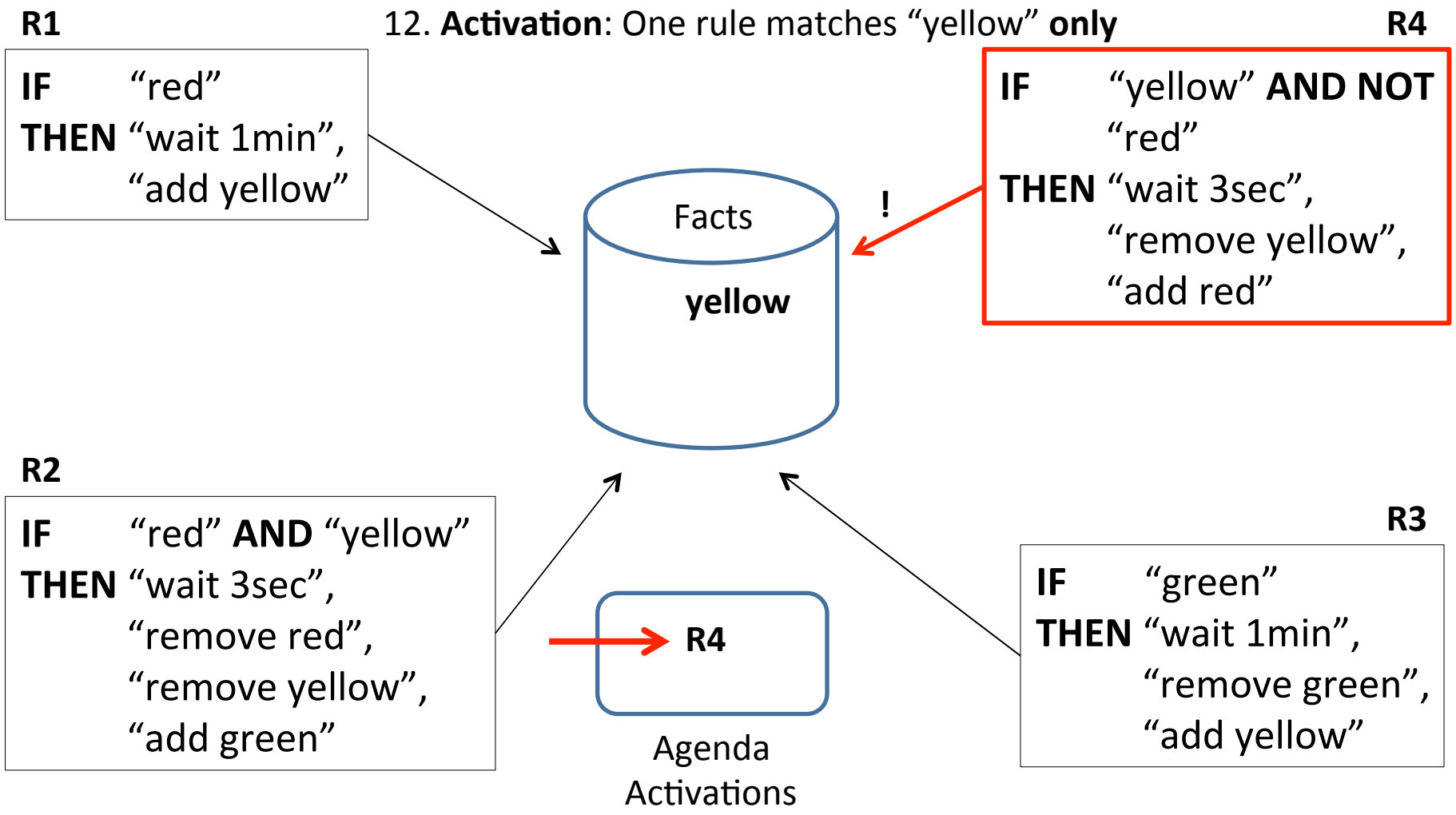
IF "green"
THEN "wait 1min",
"remove green",
"**add yellow**"

Agenda
Activations

Inference


R1
IF "red"
THEN "wait 1min",
"add yellow"
11. All rules "look" into the shared memory to test their antecedents
R4
IF "yellow" AND NOT "red"
THEN "wait 3sec",
"remove yellow",
"add red"
?
?
Facts
yellow
R2
IF "red" AND "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"
?
?
R3
IF "green"
THEN "wait 1min",
"remove green",
"add yellow"
Agenda
Activations

Inference

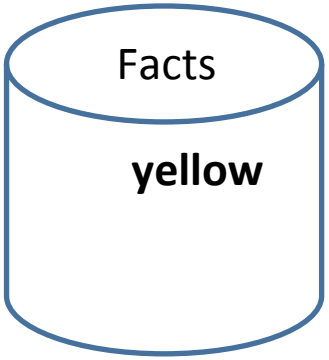

R1
IF "red"
THEN "wait 1min",
"add yellow"
12. Activation: One rule matches "yellow" only
R4
IF "yellow" AND NOT
"red"
THEN "wait 3sec",
"remove yellow",
"add red"
!
Facts
yellow
R2
IF "red" AND "yellow"
THEN "wait 3sec",
"remove red",
"remove yellow",
"add green"
R4
Agenda
Activations
R3
IF "green"
THEN "wait 1min",
"remove green",
"add yellow"

Inference

R1

IF "red"
THEN "wait 1min",
"add yellow"

13a. **Make "conclusions"**: perform action

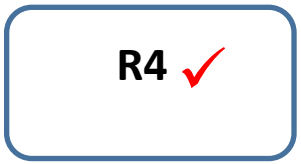


R4

IF "yellow" **AND NOT** "red"
THEN "**wait 3sec**",
"remove yellow",
"add red"

R2

IF "red" **AND** "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"



Agenda Activations

R3

IF "green"
THEN "wait 1min",
"remove green",
"add yellow"

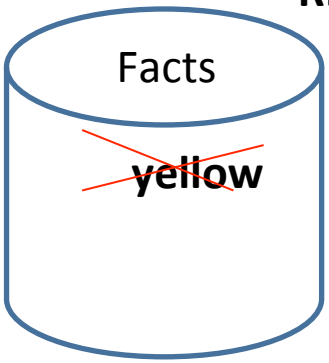
Inference

R1

IF "red"
THEN "wait 1min",
"add yellow"

13b. **Make "conclusions"**: remove fact "yellow" from shared memory

REMOVE

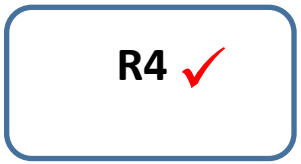


R4

IF "yellow" **AND NOT** "red"
THEN "wait 3sec",
"**remove yellow**",
"add red"

R2

IF "red" **AND** "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"



Agenda Activations

R3

IF "green"
THEN "wait 1min",
"remove green",
"add yellow"

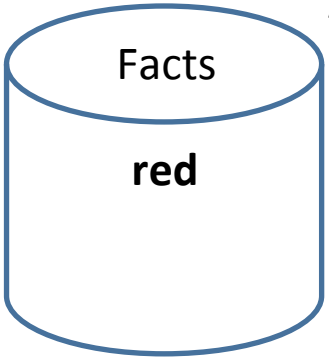
Inference

R1

IF "red"
THEN "wait 1min",
"add yellow"

13c. **Make "conclusions"**: add fact "red" to shared memory

ASSERT

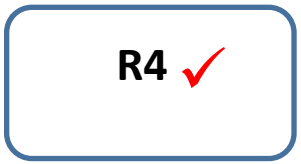


R4

IF "yellow" **AND NOT** "red"
THEN "wait 3sec",
"remove yellow",
"**add red**"

R2

IF "red" **AND** "yellow"
THEN "wait 3seconds",
"remove red",
"remove yellow",
"add green"

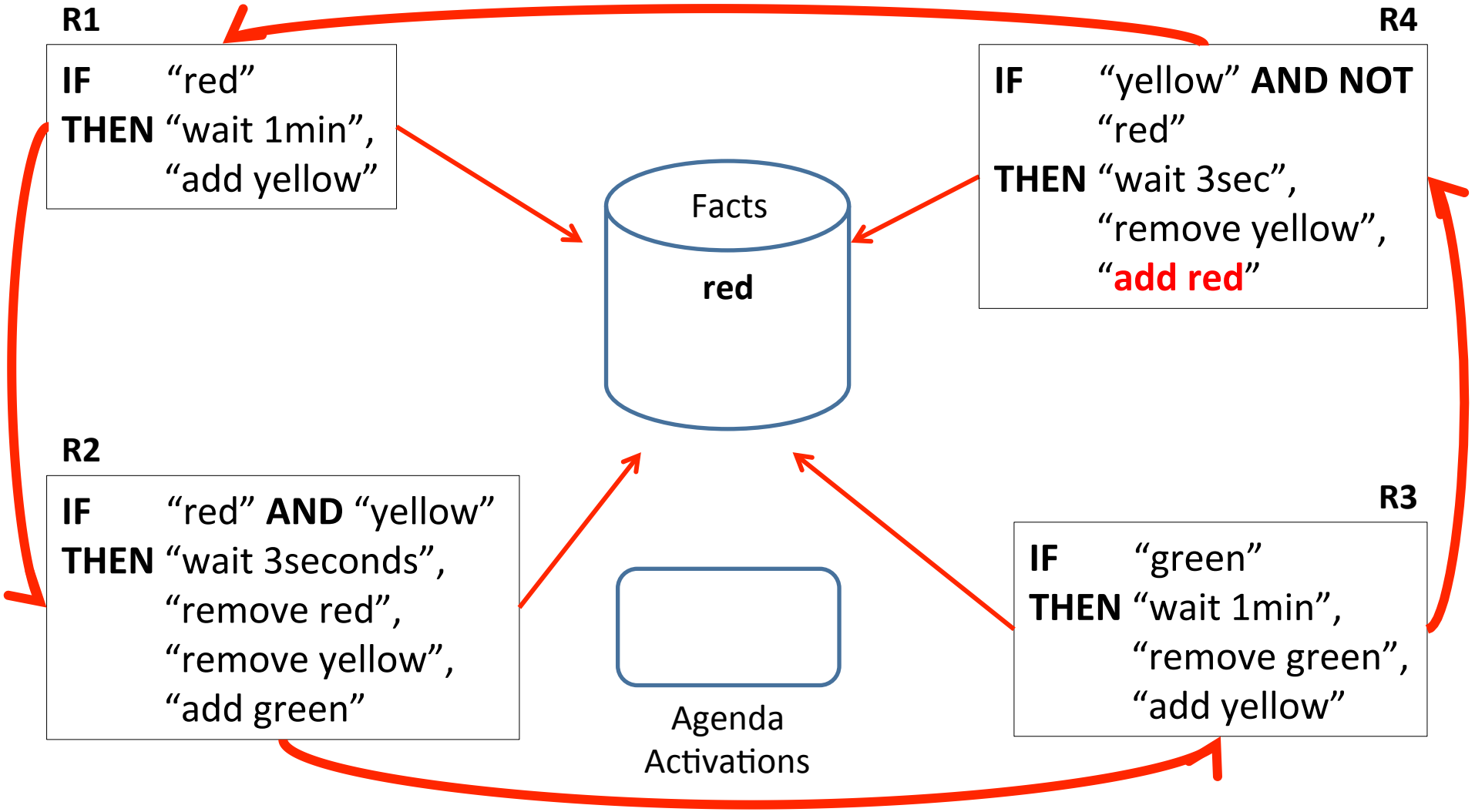


Agenda Activations

R3

IF "green"
THEN "wait 1min",
"remove green",
"add yellow"

Rule chaining

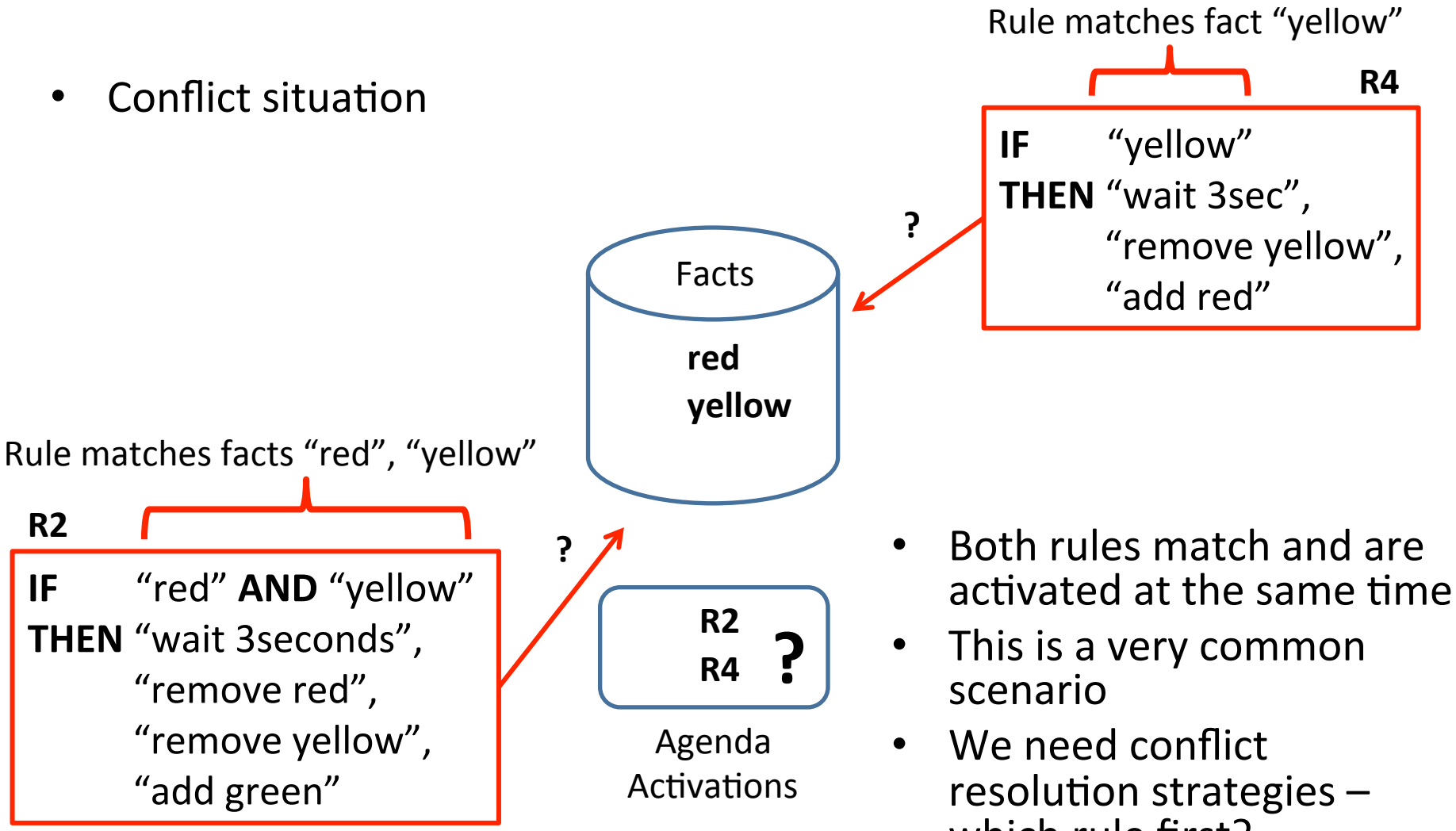


Conflict Between Rules

- More than one rule may be activated at the same time
- Inference Engine must handle such a situation
  - Maintain a list of currently active rules – the so-called “Agenda”
  - Prioritise rules on this agenda

Rule Activations

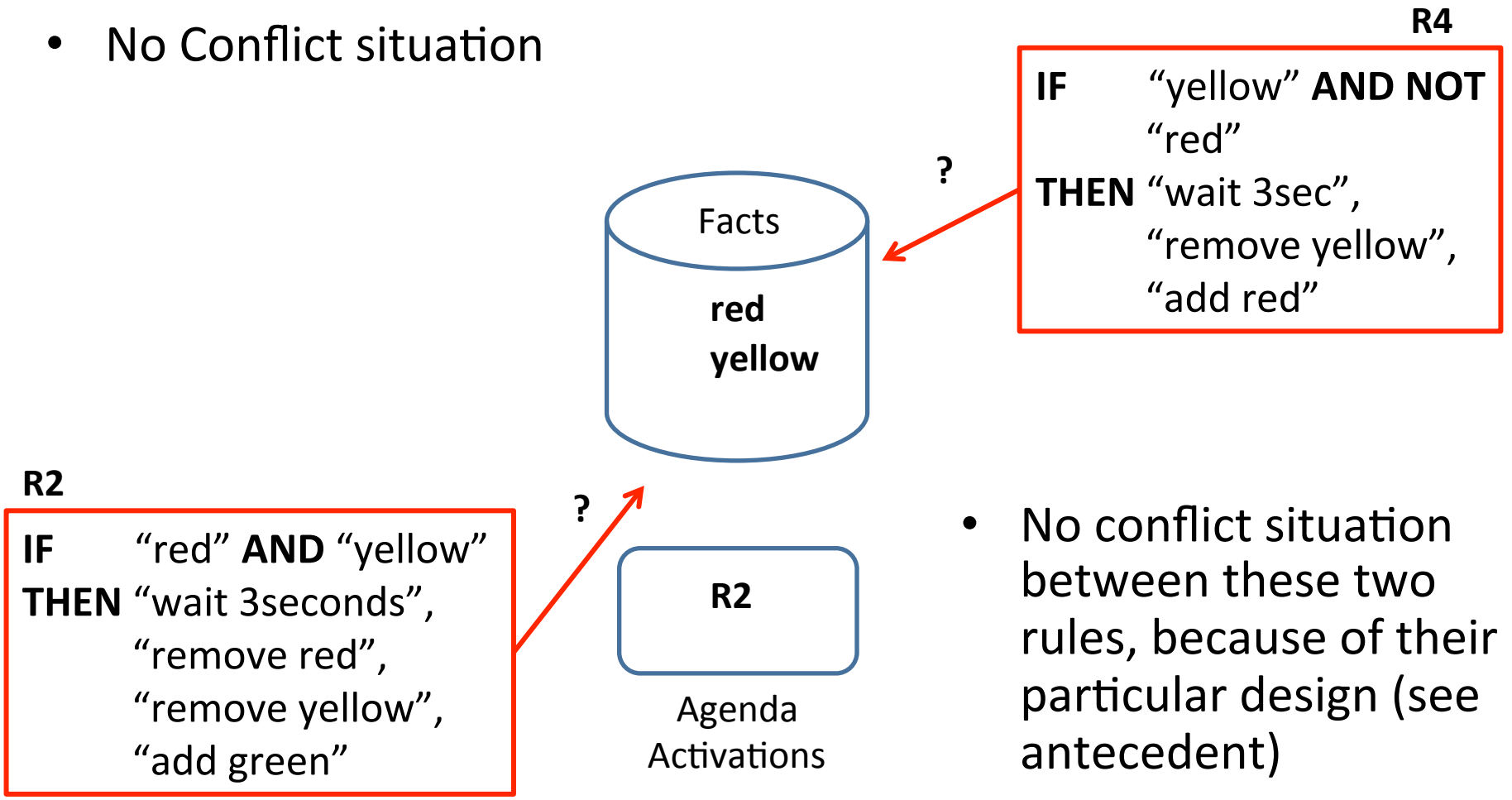
- Conflict situation



- Both rules match and are activated at the same time
- This is a very common scenario
- We need conflict resolution strategies – which rule first?

Rule Activations

- No Conflict situation



- No conflict situation between these two rules, because of their particular design (see antecedent)

Observations

- Rules fire when their antecedents / preconditions are true (satisfied/matched by known facts)
- When rules fire, we gain additional knowledge – the consequents of the rules are facts that become part of our (working) memory
- When rules fire, they may create a situation where other rules can fire as well
- More than one rule may be ready to fire, because their antecedents are true (we may have to choose which one fires first)
- This form of inference is called **Forward Chaining**

Inference: Forward Chaining

- ***Forward chaining*** works from known facts, applying **Modus Ponens** to these facts and the rules in the KB
- Given a set of rules and a number of facts, ***forward chaining*** inference attempts to prove whatever it can
- Backward chaining – will be discussed later

Forward Chaining Algorithm

- Let the WM (Working Memory) contain all known facts
- REPEAT
  - **Match and Activate**: Obtain the list of all rules whose antecedents match known facts in WM (this is the conflict set), add them to the agenda
  - **Conflict Resolution**: Select a rule from this conflict set (this is conflict resolution)
  - **Execute (fire)**: carry out actions of selected rule specified as its conclusion – usually adding facts to, or deleting them from, the WM
- UNTIL no more applicable rules on the agenda

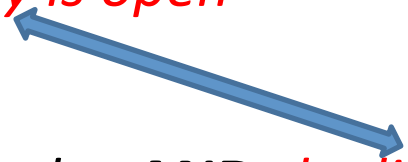
Observation

- Rule Chaining
  - One rule is changing the working memory in such a way (adding/removing facts) that other rules become activated
  - That means: one rule is constructed in a way that it creates the preconditions for other rules to be activated (fire).

Rule Chaining

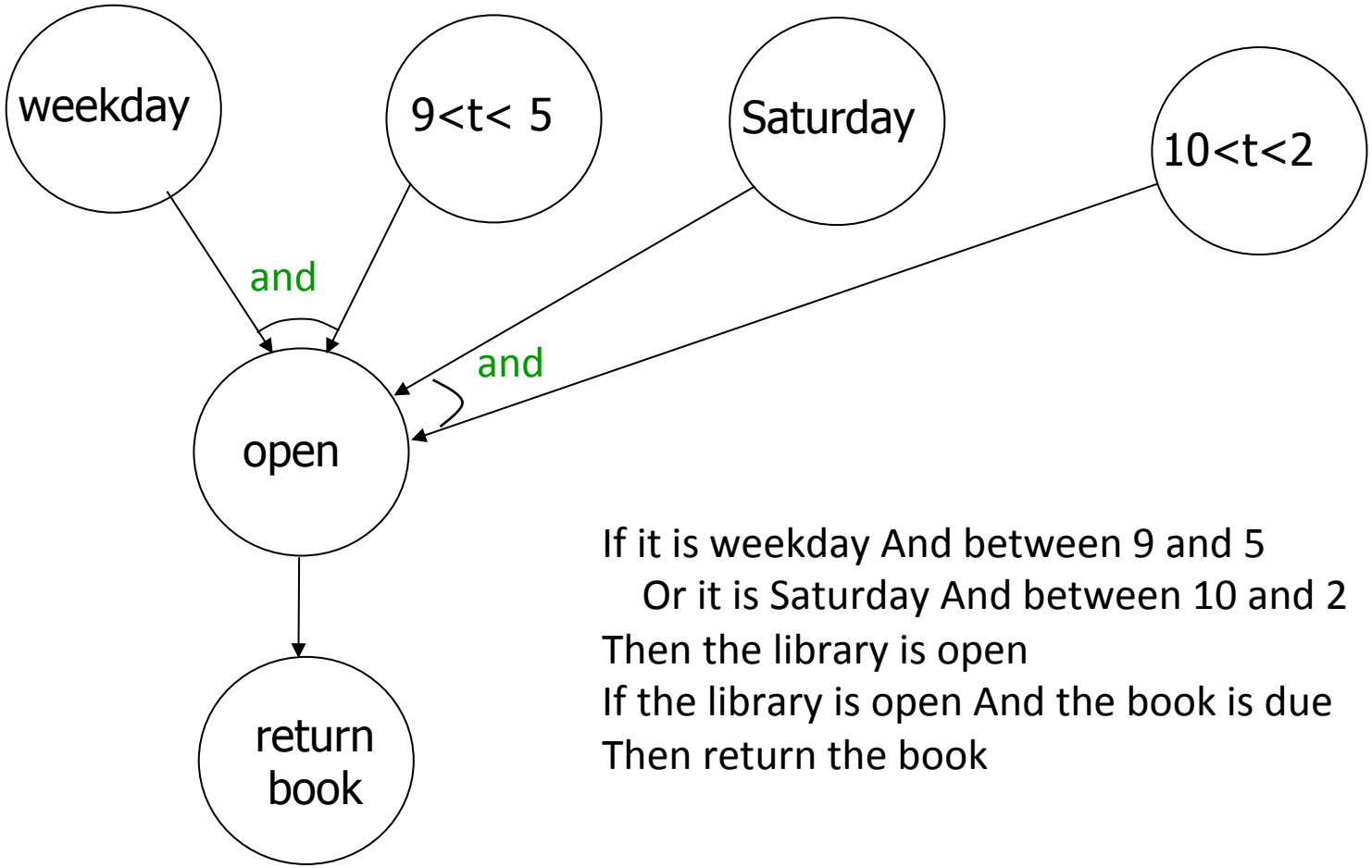
- The consequent of one rule can assert a fact that is (part of) the antecedent of another rule
- Rule chaining is an important design principle for knowledge bases
- Example:

IF *it is a weekday* **AND** *it is between 9am and 5pm*
THEN *the library is open*



IF *book is due today* **AND** *the library is open*
THEN *return book*

Rule Chaining

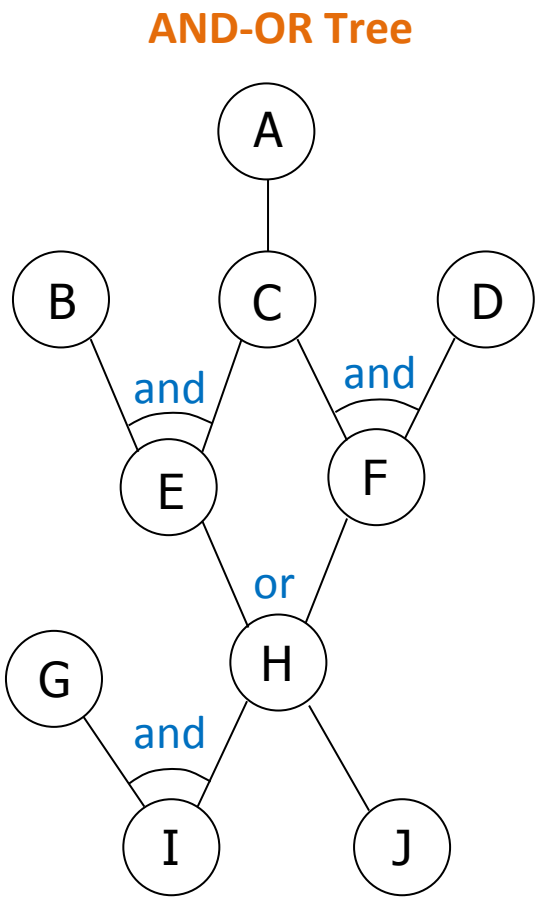


If it is weekday And between 9 and 5
Or it is Saturday And between 10 and 2
Then the library is open
If the library is open And the book is due
Then return the book

Forward Chaining

We have the following set of rules:

- R1: If A then C
- R2: If B and C then E
- R3: If C and D then F
- R4: If E then H
- R5: If F then H
- R6: If G and H then I
- R7: If H then J



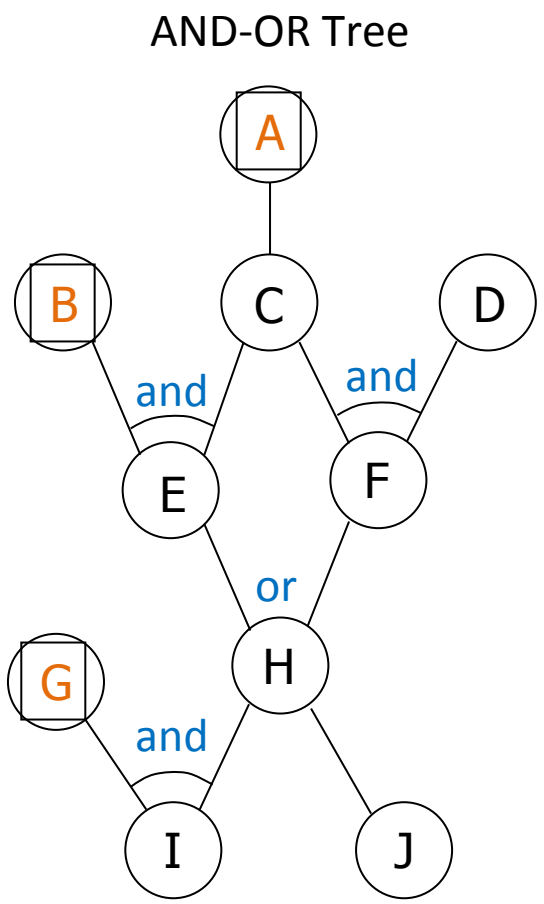
Forward Chaining

We have the following set of rules:

- R1: If A then C
- R2: If B and C then E
- R3: If C and D then F
- R4: If E then H
- R5: If F then H
- R6: If G and H then I
- R7: If H then J

We assume to be TRUE:

- A, B, G



Forward Chaining

Rule R1 becomes “active” and “fires”:

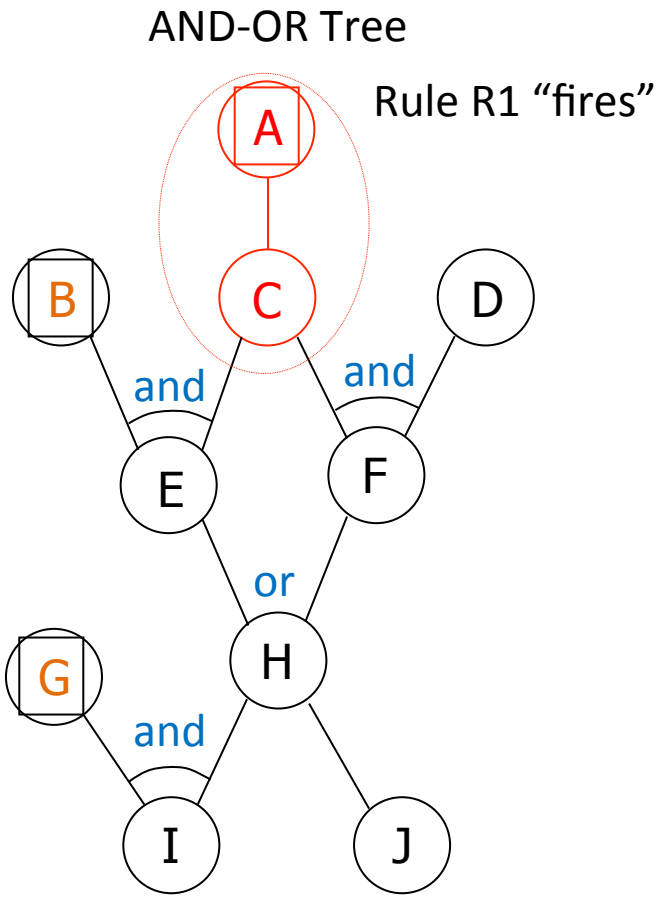
- R1: If A then C
- R2: If B and C then E
- R3: If C and D then F
- R4: If E then H
- R5: If F then H
- R6: If G and H then I
- R7: If H then J

Known Facts (Working Memory):

- A, B, G

R1 allows us to conclude:

- C is TRUE



Forward Chaining

Rule R2 becomes “active” and “fires”:

- R1: If A then C
- R2: If B and C then E
- R3: If C and D then F
- R4: If E then H
- R5: If F then H
- R6: If G and H then I
- R7: If H then J

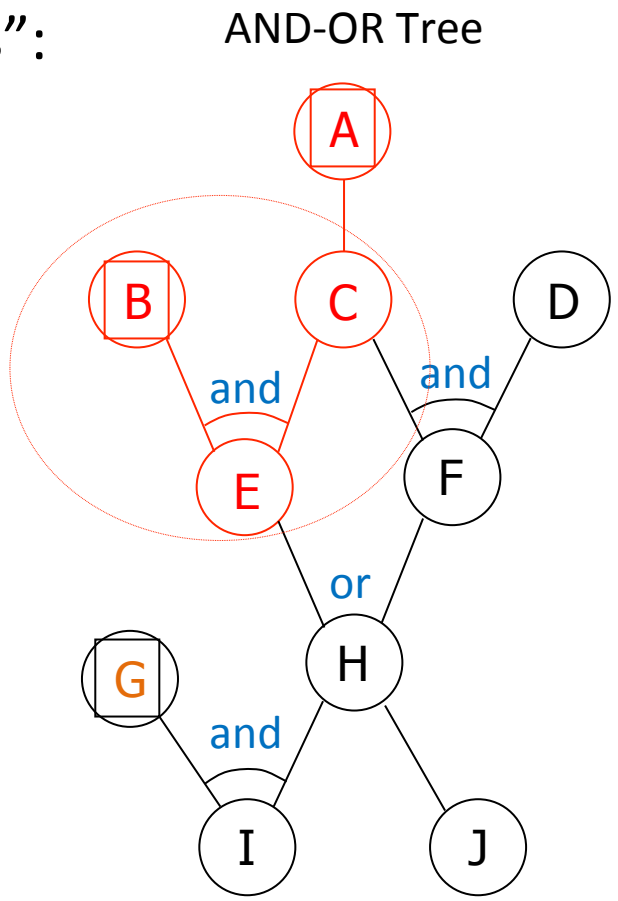
Known Facts (Working Memory):

- A, B, G
- C

R2 allows us to conclude:

- E is TRUE

Rule R2 “fires”



Forward Chaining

Rule R4 becomes “active” and “fires”:

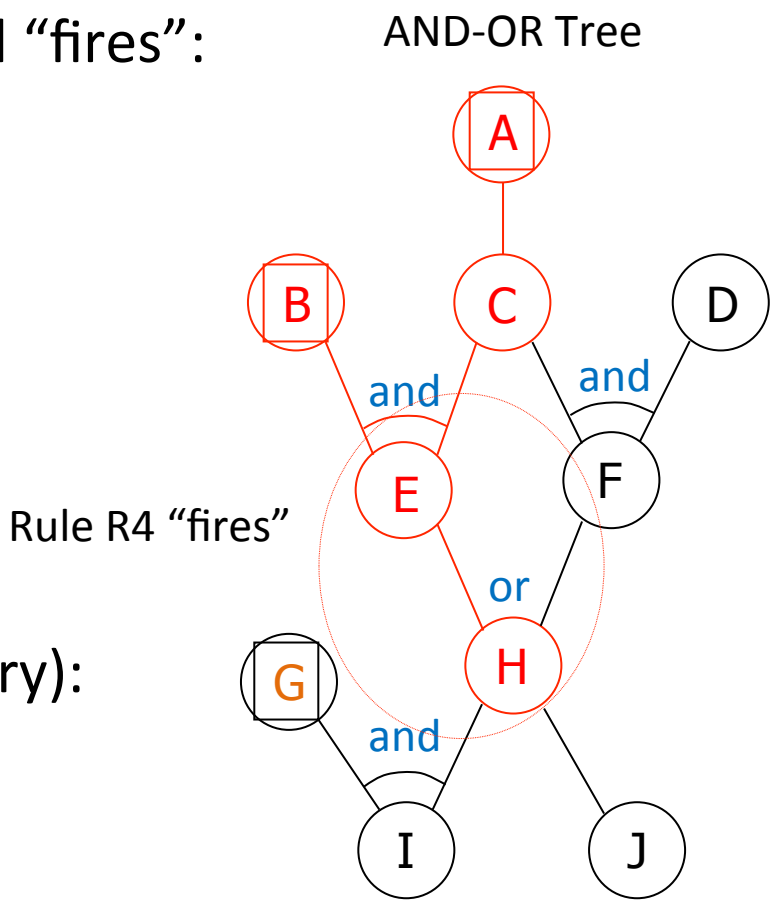
- R1: If A then C
- R2: If B and C then E
- R3: If C and D then F
- R4: If E then H
- R5: If F then H
- R6: If G and H then I
- R7: If H then J

Known Facts (Working Memory):

- A, B, G
- C,E

R4 allows us to conclude:

- H is TRUE



Forward Chaining

Rules R6,R7 become “active” and “fire”:

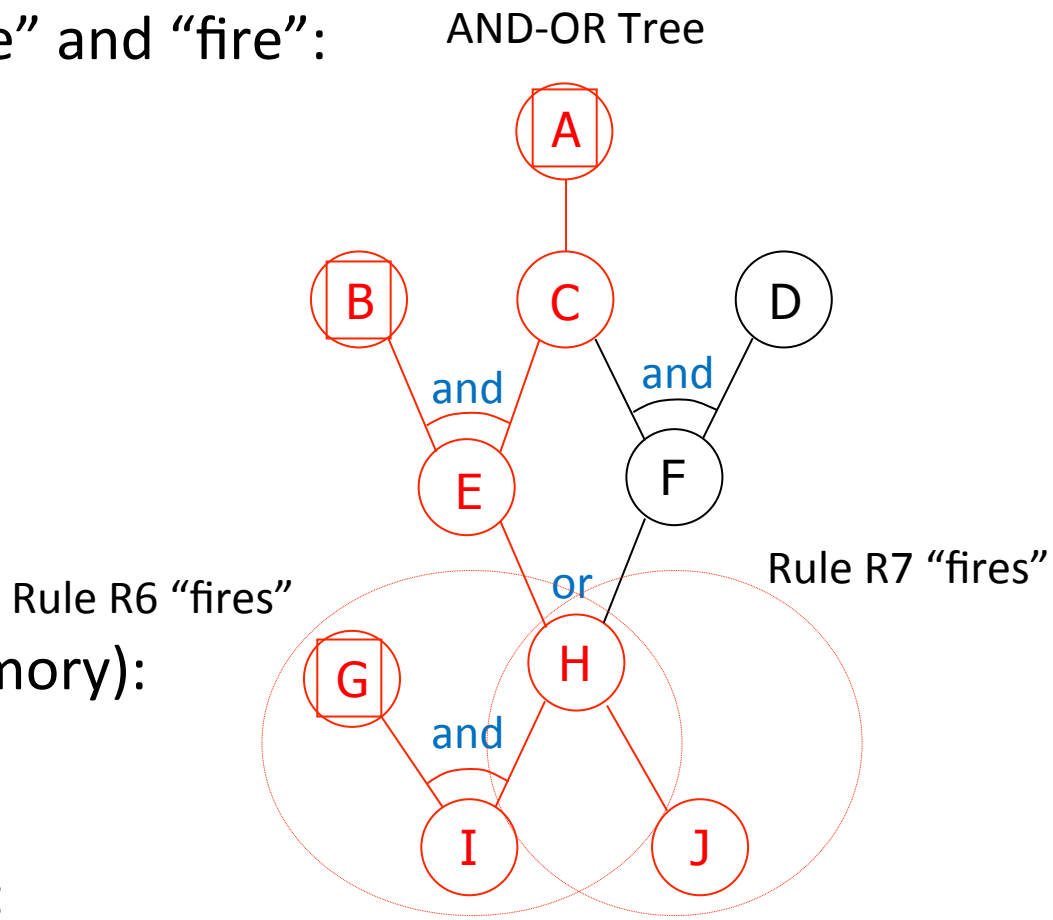
- R1: If A then assert C
- R2: If B and C then assert E
- R3: If C and D then assert F
- R4: If E then assert H
- R5: If F then assert H
- R6: If G and H then assert I
- R7: If H then assert J

Known Facts (Working Memory):

- A, B, G
- C,E,H

R6,R7 allow us to conclude:

- I,J are TRUE

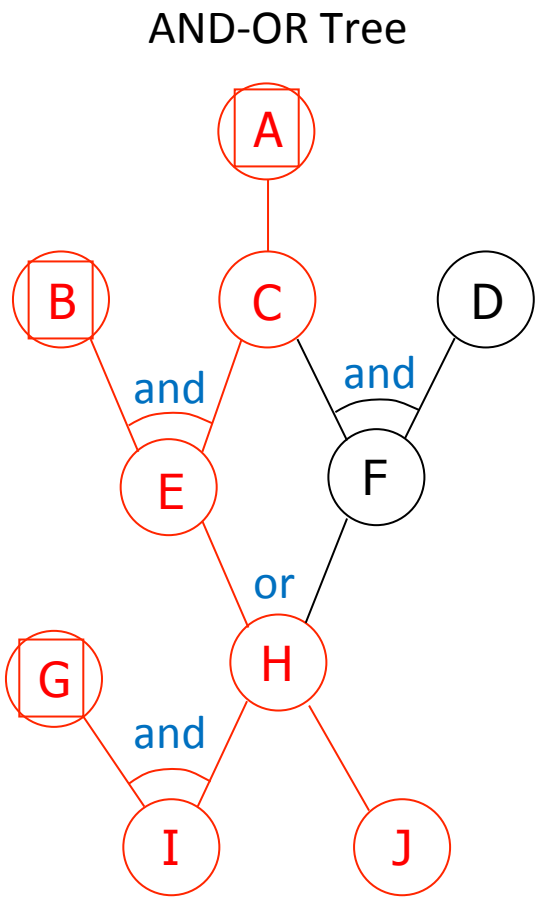


Forward Chaining

- R1: If A then assert C
- R2: If B and C then assert E
- R3: If C and D then assert F
- R4: If E then assert H
- R5: If F then assert H
- R6: If G and H then assert I
- R7: If H then assert J

Known Facts (Working Memory):

- A, B, G
- C, E, H, I, J



Result: we know now more about the world – that facts C, E, H, I, J are true as well

Non-monotonic Inference in JESS

Non-monotonic Inference

- Inference does not just mean new facts asserted by rules
  - Non-monotonic inference – we also remove ("**retract**") facts that were asserted previously
- The RHS of a rule allows us to
  - Assert new facts
  - Retract facts
  - Modify facts
- This allows us to model a changing world, for example exploring different options in search of an optimal solution:
  - A robot moving blocks around and/or stacking them
  - Putting goods into bags (finding the best combination per bag)

Retracting Facts

- In order to retract a fact from working memory in Jess, we have to find/ retrieve it first
- Facts have unique identifiers or “addresses” managed by Jess

```
f-0     (MAIN::initial-fact)
f-1     (MAIN::initial-state A on table clear)
f-2     (MAIN::initial-state B on table not-clear)
f-3     (MAIN::initial-state C on B clear)
f-4     (MAIN::goal C on B clear)
f-5     (MAIN::goal B on A not-clear)
f-6     (MAIN::goal A on table not-clear)
```

- A fact is found in (retrieved from) working memory, when it is matched by the LHS of a rule
- With a specially constructed LHS of a rule, we can bind a variable with the Jess-internal address of a fact:

```
(defrule check-goals
        ?factid   <-   (goal ?x on ?y clear)
        =>
        (printout t "We found fact " ?factid " where " ?x " is clear" crlf ))
```

Retract Facts

- Bind variables to fact addresses:
  - We have the following fact in WM
    - `f-4  (MAIN::goal C on B clear)`
  - We use the following LHS of a rule to match it and also record its Jess-internal address:

```
(defrule check-goals
        ?factid   <-   (goal ?x on ?y clear)
        =>
        (retract ?factid ) )
```

  - `?factid = 4`
  - `?x = C, ?y = B`
- (retract ?factid) removes fact 4 from working memory

f-1 (MAIN:: Black Horse G 3)
Jess> (retract (fact-id 1))
remove (Black Horse G 3) !!!

- Questions?