Reasoning with Uncertainty

The Jess Language Part 6
CS3025, Knowledge-Based Systems
Lecture 17

Yuting Zhao
Yuting.zhao@gmail.com

2017-11-14

Uncertainty in Inference

- So far:
  - We were reasoning with **certainty** – we assumed that we live in clear-cut world where each fact, hypothesis and conclusion are either true, false or unknown
  - We also made the *closed-world assumption (**CWA**)* – anything that is unknown is regarded as being false
  - With this assumption, we arrived at a rather **binary** world – in our reasoning we draw conclusions that tell us that something is either true or false
- We did not question the validity of such an inference
- What if we cannot draw a particular conclusion from given facts with complete certainty?

Uncertainty in Inference

- What is **uncertainty**?
  - Some data/knowledge we are not 100% sure
  - Real world, not all our data / knowledge is 100% certain
- **Different approaches** to uncertainty
  - What data / knowledge can be represented as uncertain?
  - What is this representation (value)?
  - How are different pieces of evidence combined to come to a particular conclusion and "how good" is this conclusion?

- Knowledge
- Value
- Operation

MYCIN

- **Classic expert system** for the diagnosis and treatment of blood infections
  - Developed at Stanford University Shortliffe, Buchanan *et al.* from 1972
  - Important because the approach used (representation of uncertainty) has been widely copied.
- Uses so-called "**certainty factors**" to represent uncertainty

- Knowledge
- Value
- Operation

MYCIN Knowledge Representation

- Rules have the following structure:

if	the site of the culture is blood
and	the identity of the organism is not definitely known
and	the stain of the organism is gramnegative
and	the morphology of the organism is rod
and	the patient is suffering from burns
then	there is **suggestive evidence (0.4)** that the identity of the organism is pseudomonas

- Question: what does "suggestive evidence (0.4)" mean?

MYCIN Uncertainty

- **Suggestive Evidence**:
  - It means there is some uncertainty regarding the consequent of the rule given those antecedents

- MYCIN uses "certainty factors" **(CF)** to represent uncertainty
  - A certainty factor is a number in [-1, +1]
  - It is associated either with a rule or a fact

Certainty Factors

- Certainty factors are in the range of [-1,1]
  - Certainty factor CF: -1 <= CF <= +1
- For **Facts**

CF = +1	The fact is **certainly true**
CF = 0	We know nothing about whether the fact is true or not (**unknown**)
CF = -1	The fact is **certainly not true**

- ***Knowledge***
- Value
- Operation

Certainty Factors

- Certainty factors are in the range of [-1,1]
  - Certainty factor CF: $-1 <= CF <= +1$
- For **Rules**

CF = +1	If the antecedent is known to be true with certainty then the consequent is known with certainty to be true
CF = 0	The antecedent brings no evidence for or against the consequent
CF = -1	If the antecedent is known with certainty not to be true, the consequent is known with certainty not to be true

- ***Knowledge***
- Value
- Operation

Reasoning with Certainty Factors

- The certainty factor model is not based on **probability** theory
- However, it is a **heuristic** that does seem to work!
- In terms of forward chaining inference, conclusions are drawn from given facts with a particular certainty – this certainty has to be calculated:
  - Facts have a certainty factor
  - Rules have a certainty factor
  - These certainty factors are "combined" during reasoning to assign a certainty factor to the conclusion
  - This is called "combining evidence" for a particular conclusion
    - Antecedent combination
    - Antecedent to consequent propagation
    - Multiple consequent combination

- Knowledge
- Value
- ***Operation***

Combining Evidence

- Combining Evidence for a conclusion is done in three stages
  - Antecedent combination (**LHS**)
    - Find the minimum confidence factor of facts matched at the LHS of a rule
  - Antecedent to consequent propagation (**LHS -> RHS**)
    - Rules have their own confidence factor, combine this with the confidence factor of the antecedent
  - Multiple consequent combination (**RHS**)
    - If two rules produce the same conclusion (maybe with different confidence factors), both conclusions (their confidence factors) have to be taken into account to derive a combined confidence factor

Antecedent Combination (LHS)

- Procedure: “If there are multiple antecedents of a rule, take the minimum”
  - If the rule matches a set of facts with its LHS (each fact with a CF) in a particular activation, choose the minimum CF
  - The **minimum** CF is the CF of the antecedent (LHS) of the rule

```
(deftemplate myfact (slot name)(slot cf))

(assert (myfact (name A)(cf 0.4)))
(assert (myfact (name B)(cf 0.9)))

(defrule R1
      (myfact (name A)(cf ?x1))
      (myfact (name B)(cf ?x2))
    =>
     . . .
)
```

For example: rule R1 matches two facts:
- Fact A has a CF = 0.4
- Fact B has a CF = 0.9

Therefore: the CF for the LHS of R1 is 0.4

Antecedent Combination
Jess codes

```
(deftemplate myfact (slot name)(slot cf))

(assert (myfact (name A)(cf 0.4)))
(assert (myfact (name B)(cf 0.9)))

(defrule R1
      (myfact (name A)(cf ?x1))
      (myfact (name B)(cf ?x2))
     =>
     (bind antecent-cf (min ?x1 ?x2))
)
```

Antecedent to Consequent Propagation **(LHS -> RHS)**

- Each rule also has a confidence factor
- In order to calculate the CF of the consequent of a rule:
  - Once the minimum CF of the LHS of a rule is determined, the consequence is:
    - **Minimum CF of LHS x Rule CF = Consequent CF**
- Example
  - The LHS CF = 0.4
  - The rule CF = 0.5
  - Therefore: the consequent CF = 0.4 x 0.5 = 0.2

Antecedent to Consequent Propagation Jess codes

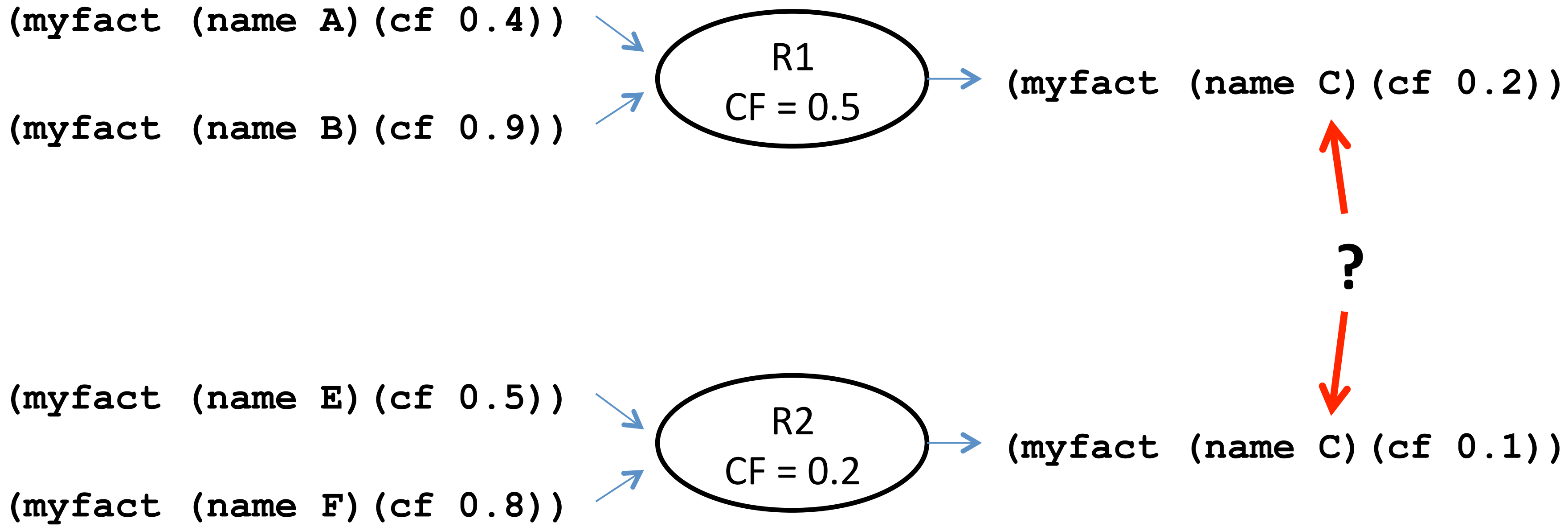
```
(deftemplate myfact (slot name)(slot cf))

(assert (myfact (name A)(cf 0.4)))
(assert (myfact (name B)(cf 0.9)))

(defrule R1
      (myfact (name A)(cf ?x1))
      (myfact (name B)(cf ?x2))
     =>
     (bind antecedent-cf (min ?x1 ?x2))
     (bind rule-CF 0.5)
     (assert
         (myfact (name C)(cf (* rule-cF antecedent-cf))))
)
```

Working Memory:

```
(myfact (name A)(cf 0.4))
(myfact (name B)(cf 0.9))
(myfact (name C)(cf 0.2))
```

Multiple Consequent Combination (RHS)

- This calculation has to be applied, if there are two rules that produce the same consequent



Multiple Consequent Combination

(1) Both Consequents are Positive

- If there exist two consequent evaluations, then the combined confidence factor CF of the consequent is determined in the following way:
  - Given two rules R1, R2
    - R1 produces a consequent with a confidence factor CF_1
    - R2 produces the same consequent with a confidence factor CF_2
  - If both CF_1 and CF_2 are **positive**, then use the following formula to calculate the final CF of the consequent:

$$
\boxed{\begin{aligned} CF &= CF_1 + CF_2 \times (1 - CF_1) \\ &= CF_1 + CF_2 - (CF_1 \times CF_2) \end{aligned}}
$$

Multiple Consequent Combination

(2) Both Consequents are Negative

- If both consequent evaluations are negative:
  - Given two rules R1, R2
    - R1 produces a consequent with a negative confidence factor CF_1
    - R2 produces the same consequent with a negative confidence factor CF_2
  - Use the following formula to calculate the final CF of the consequent:

$$\boxed{\begin{aligned} CF &= CF_1 + CF_2 \times (1 + CF_1) \\ &= CF_1 + CF_2 + (CF_1 \times CF_2) \end{aligned}}$$

Multiple Consequent Combination

(3) Positive and Negative CF

- If one consequent evaluation is positive and one is negative, then use the following formula to calculate the final CF of the consequent:

$$\frac{CF_1 + CF_2}{1 - \min(|CF_1|, |CF_2|)}$$

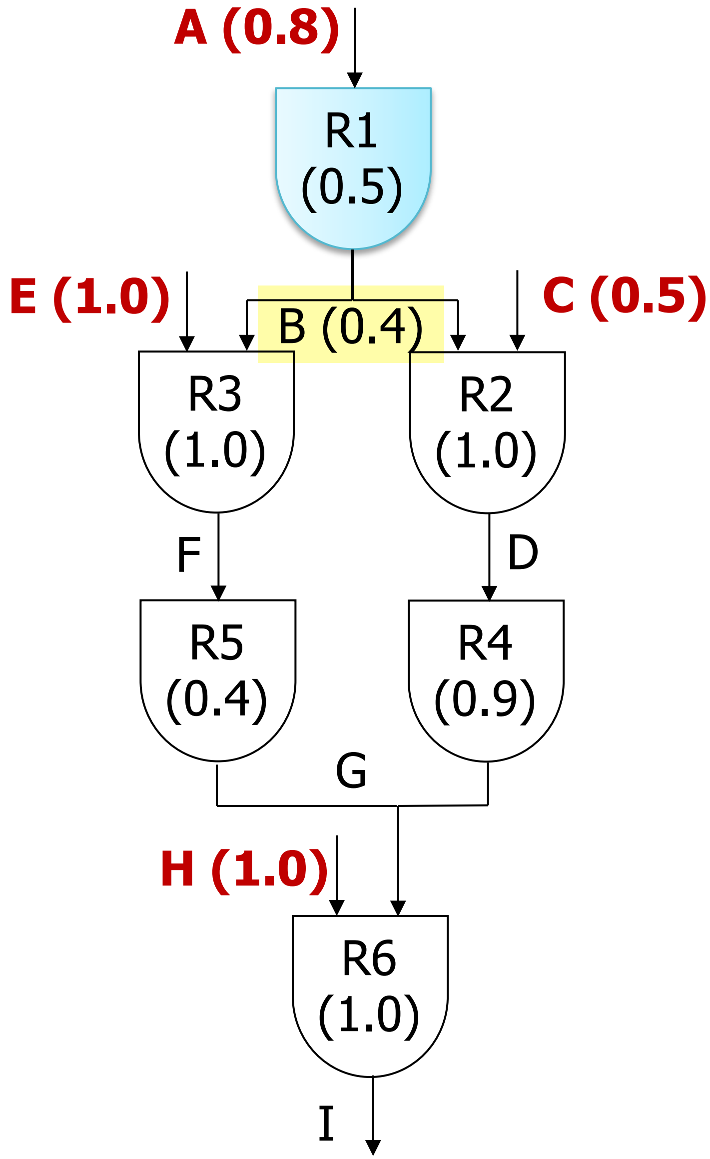
Forward Chaining Inference with Confidence Factors

- Initial Working Memory

A (CF = 0.8)
C (CF = 0.5)
E (CF = 1.0)
H (CF = 1.0)

- Rule R1 fires
- B (CF = 0.8 x 0.5 = 0.4) added to WM
- Extended Working Memory

A (CF = 0.8), B (CF = 0.4)
C (CF = 0.5)
E (CF = 1.0)
H (CF = 1.0)



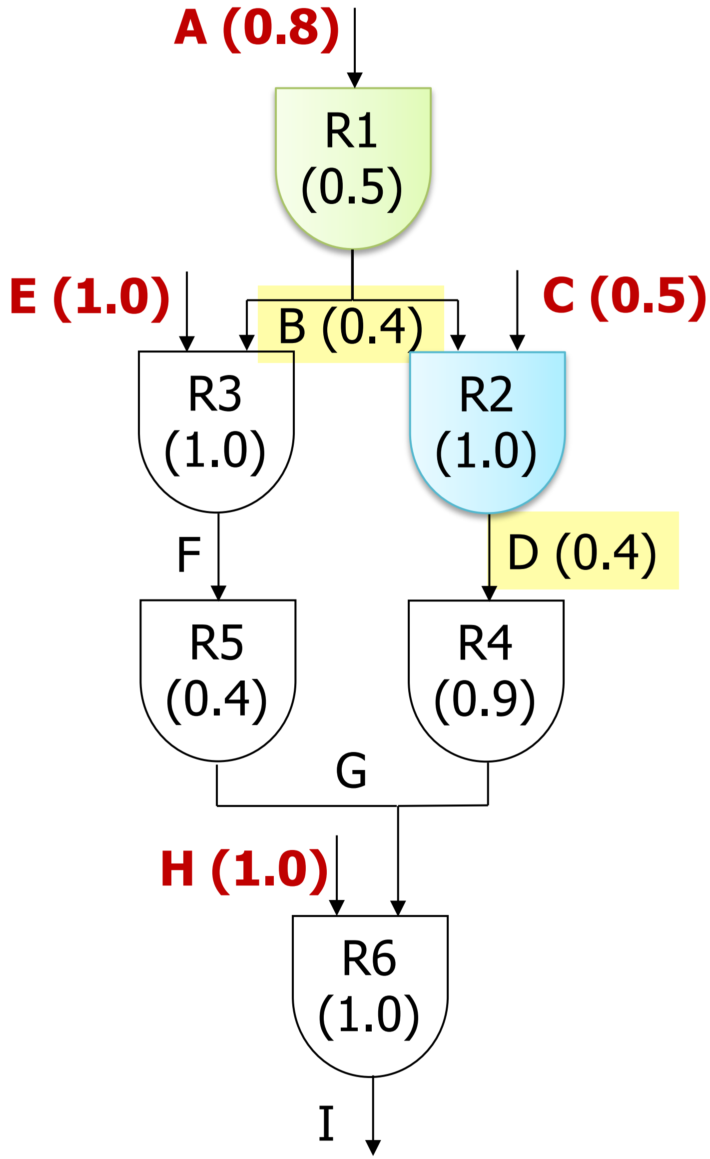
Forward Chaining Inference with Confidence Factors

- Working Memory

A (CF = 0.8), B (CF = 0.4)
C (CF = 0.5)
E (CF = 1.0)
H (CF = 1.0)

- Minimum LHS of R2 is 0.4, Rule R2 fires
- D (CF = 0.4 x 1.0 = 0.4) added to WM
- Working Memory

A (CF = 0.8), B (CF = 0.4)
C (CF = 0.5), D (CF = 0.4)
E (CF = 1.0)
H (CF = 1.0)



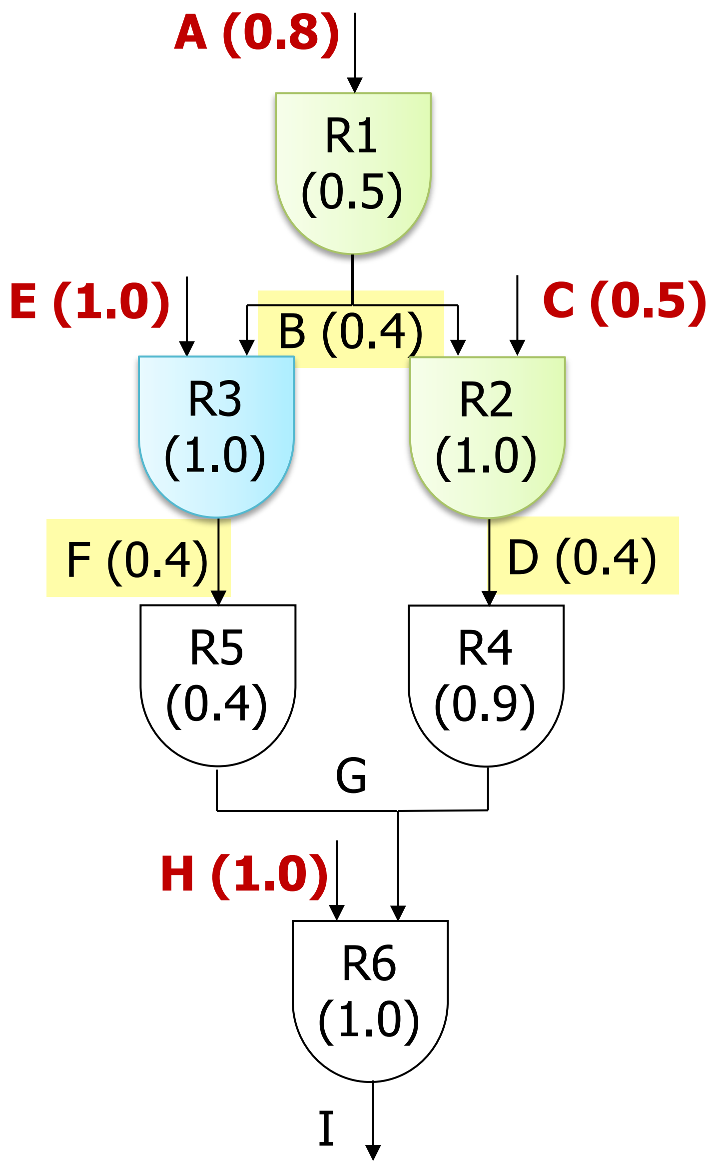
Forward Chaining Inference with Confidence Factors

- Working Memory

A (CF = 0.8), B (CF = 0.4)
C (CF = 0.5), D (CF = 0.4)
E (CF = 1.0)
H (CF = 1.0)

- Minimum LHS of R3 is 0.4, Rule R3 fires
- F (CF = 0.4 x 1.0 = 0.4) added to WM
- Working Memory

A (CF = 0.8), B (CF = 0.4)
C (CF = 0.5), D (CF = 0.4)
E (CF = 1.0), F (CF = 0.4)
H (CF = 1.0)



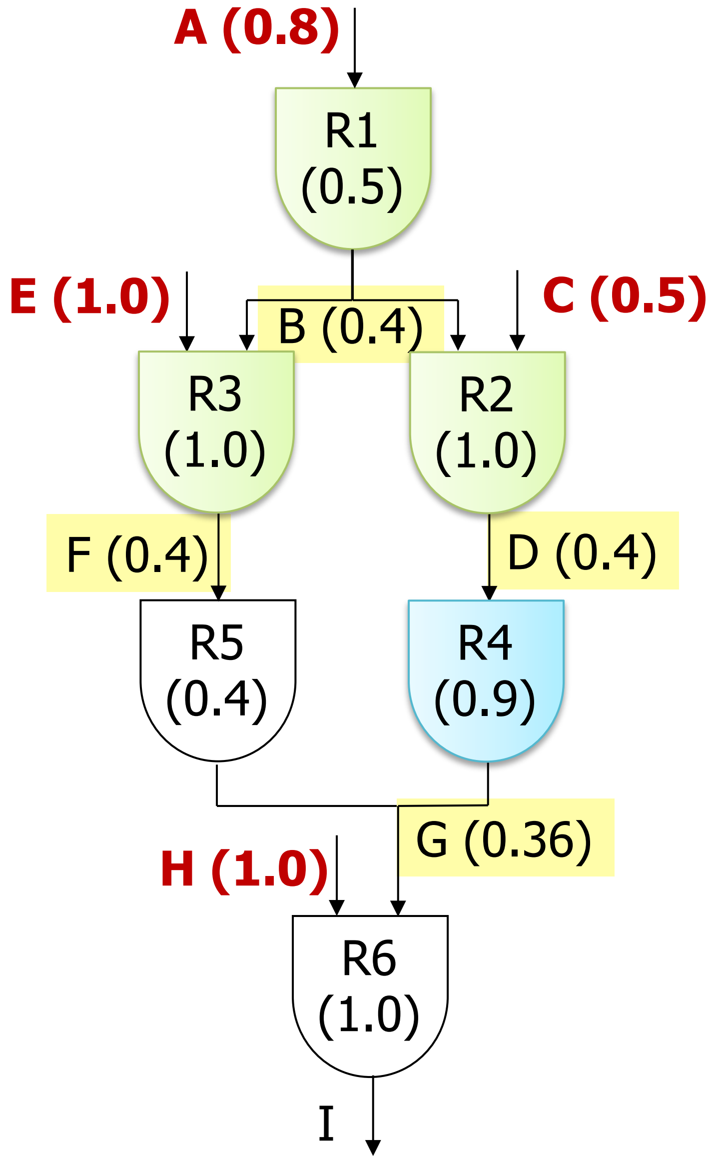
Forward Chaining Inference with Confidence Factors

- Working Memory

A (CF = 0.8), B (CF = 0.4)
C (CF = 0.5), D (CF = 0.4)
E (CF = 1.0) , F (CF = 0.4)
H (CF = 1.0)

- Minimum LHS of R4 is 0.4, Rule R4 fires
- G (CF = 0.4 x 0.9 = 0.36) added to WM
- Working Memory

A (CF = 0.8), B (CF = 0.4)
C (CF = 0.5), D (CF = 0.4)
E (CF = 1.0), F (CF = 0.4)
H (CF = 1.0), G (CF = 0.36)



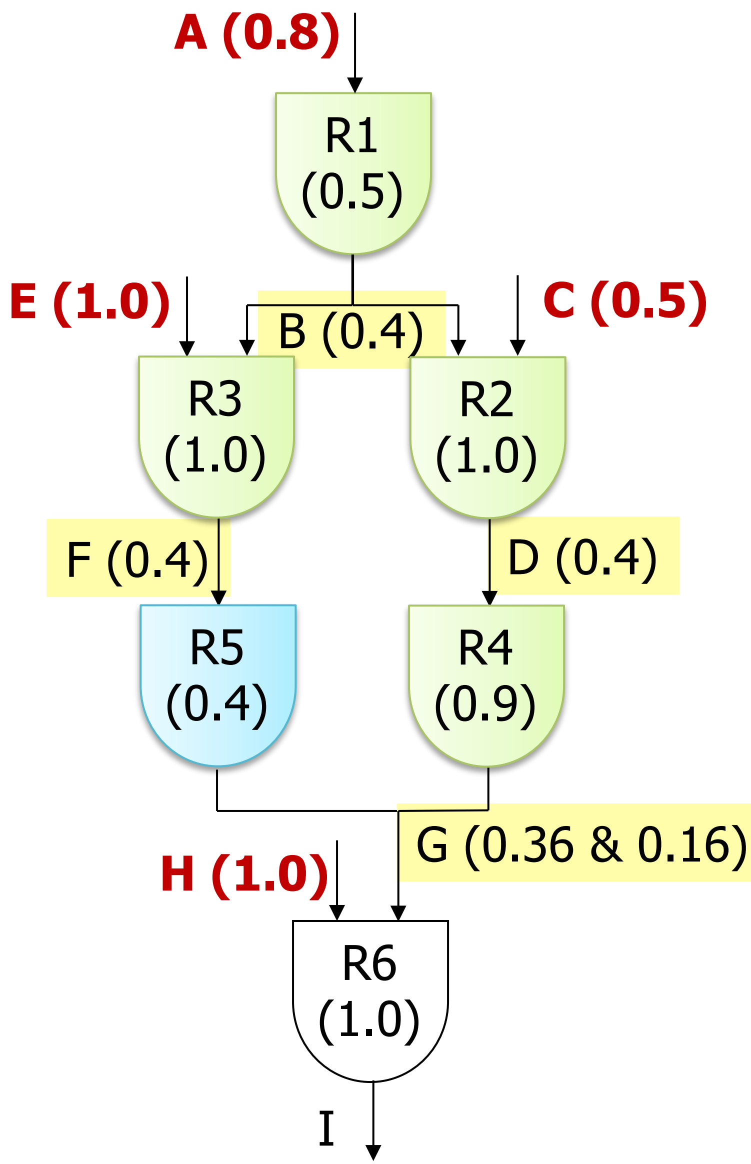
Forward Chaining Inference with Confidence Factors

- Working Memory

A (CF = 0.8), B (CF = 0.4)
C (CF = 0.5), D (CF = 0.4)
E (CF = 1.0) , F (CF = 0.4)
H (CF = 1.0), G (CF = 0.36)

- Minimum LHS of R5 is 0.4, Rule R5 fires
- G (CF = 0.4 x 0.4 = 0.16) added to WM
- Working Memory

A (CF = 0.8), B (CF = 0.4)
C (CF = 0.5), D (CF = 0.4)
E (CF = 1.0), F (CF = 0.4)
H (CF = 1.0), G (CF = 0.36 & 0.16)



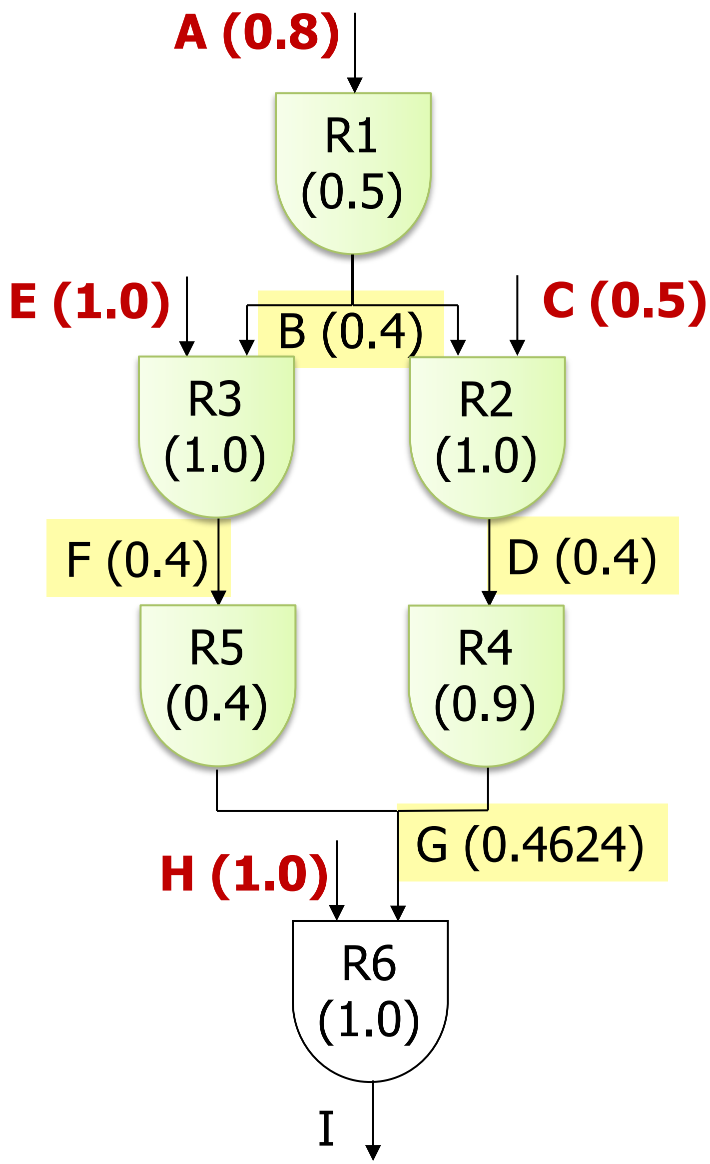
Multiple Consequent Combination

- Working Memory

A (CF = 0.8), B (CF = 0.4)
C (CF = 0.5), D (CF = 0.4)
E (CF = 1.0) , F (CF = 0.4)
H (CF = 1.0), G (CF = 0.36 & 0.16)

- Multiple Consequent Combination for G
  - CF1 = 0.36 (from rule R4)
  - CF2 = 0.16 (from rule R5)
- Both are positive, therefore calculate
  - CF = CF1 + CF2 – (CF1 x CF2)

    = 0.36 + 0.16 – (0.36 x 0.16)
- Resulting Confidence Factor for G
  - G (CF = 0.4624)



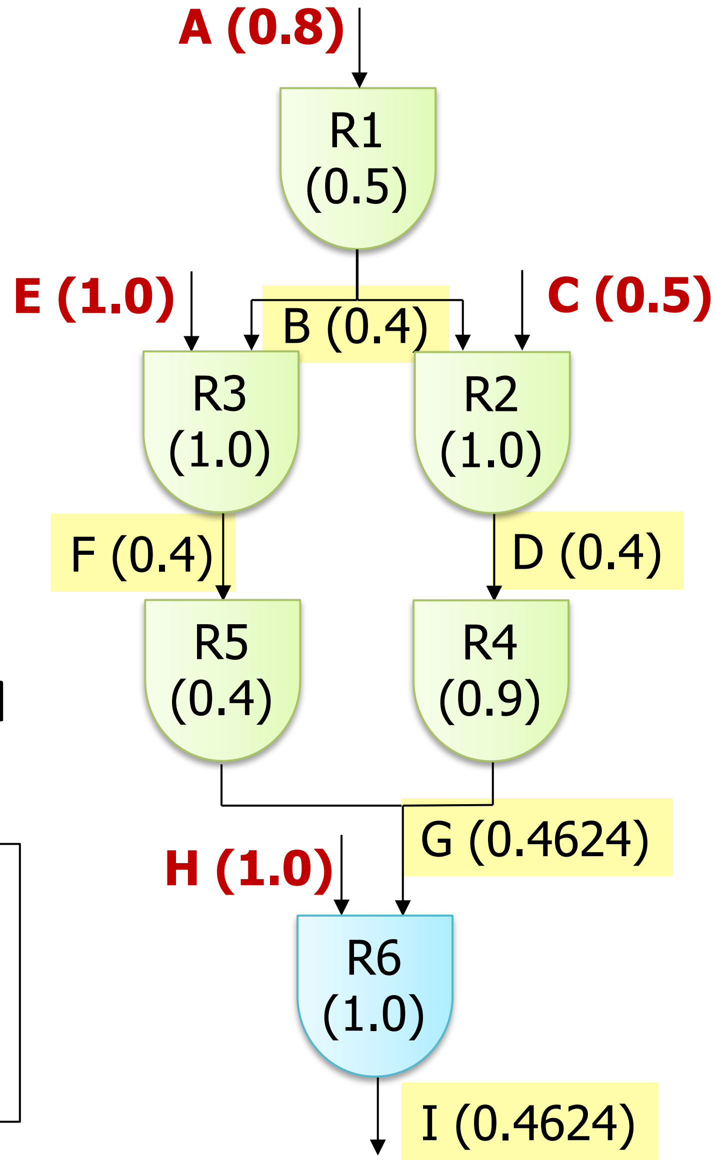
Forward Chaining Inference with Confidence Factors

- Working Memory

A (CF = 0.8), B (CF = 0.4)
C (CF = 0.5), D (CF = 0.4)
E (CF = 1.0) , F (CF = 0.4)
H (CF = 1.0), G (CF = 0.4624)

- Minimum LHS of R5 is 0.4, Rule R5 fires
- G (CF = 0.4 x 0.4 = 0.16) added to WM
- Working Memory

A (CF = 0.8), B (CF = 0.4)
C (CF = 0.5), D (CF = 0.4)
E (CF = 1.0), F (CF = 0.4)
H (CF = 1.0), G (CF = 0.4624), I (CF = 0.4624)



Multiple Consequent Combination
Both Consequents are Positive

```
(deftemplate myfact (slot name)(slot cf))

(assert (myfact (name A)(cf 0.4)))
(assert (myfact (name B)(cf 0.9)))

(defrule antecedent-to-consequent
     ?f1 <- (myfact (name ?name)(cf ?x1))
     ?f2 <- (myfact (name ?name)(cf ?x2 & :(<> ?x1 ?x2))
     (test (and (> ?x1 0)(> ?x2 0)))
    =>
    (retract ?f1)
    (retract ?f2)
    (assert
       (myfact (name C)(cf (-(+ ?x1 ?x2)(* ?x1 ?x2))))))

)
```

Multiple Consequent Combination Both Consequents are Negative

```
(deftemplate myfact (slot name)(slot cf))

(assert (myfact (name A)(cf 0.4)))
(assert (myfact (name B)(cf 0.9)))

(defrule antecedent-to-consequent
      ?f1 <- (myfact (name ?name)(cf ?x1))
      ?f2 <- (myfact (name ?name)(cf ?x2 & :(<> ?x1 ?x2))
      (test (and (<= ?x1 0)(<= ?x2 0)))
     =>
     (retract ?f1)
     (retract ?f2)
     (assert
        (myfact (name C)(cf (+(+ ?x1 ?x2)(* ?x1 ?x2))))))

)
```

Multiple Consequent Combination Positive and Negative CF

```
(deftemplate myfact (slot name)(slot cf))

(assert (myfact (name A)(cf 0.4)))
(assert (myfact (name B)(cf 0.9)))

(defrule antecedent-to-consequent
     ?f1 <- (myfact (name ?name)(cf ?x1))
     ?f2 <- (myfact (name ?name)(cf ?x2 & :(<> ?x1 ?x2))
     (test (or(and (> ?x1 0)(<= ?x2 0))
               (and (> ?x2 0)(<= ?x1 0)))
    =>
    (retract ?f1)
    (retract ?f2)
    (assert
       (myfact (name C)
               (cf (/(+ ?x1 ?x2)
                     (- 1 (min(abs ?x1)(abs ?x2))))))
)
```

Summary

- Not all knowledge is 100% certain
- Using confidence factors to represent uncertainty
  - Antecedent combination
  - Antecedent to consequent propagation
  - Multiple consequent combination
- Both forward and backward chaining with confidence factors possible

- Question?

General issues in the assessment

- Typo
  - " ?PersonObject1 " or " http://www.cs4021/bookstore.owl#?PersonObject1 "
- Ontology problem
  - Some results should be shown but not
  - "Purchase_1" "hasBuyer" "Jeff", but when checking "Jeff", "Purchase_1" is not under "hasMadePurchase"
  - Solution: (1) manually fix (2) InverseFunctional

General issues in the assessment

- Can not find all purchases
  - (http:// … #hasMadePurchase $? ?Purch1 $?)
  - What about (http:// … #hasMadePurchase ?Purch1 $?)
  - Only check the 1st purchase
- Build your own rule, by using what you need
  - (is-a   http://www.cs4021/bookstore.owl#Person) or

  (is-a http://www.cs4021/bookstore.owl#Customer)

General issues in the assessment

- Debugging:
  - Check the facts: are they all right?
  - Write testing programs
    - (defrule list-all-customer / list-all-person / list-all-book ……
    - (defrule list-all-purchase ….
    - (defrule list-all-customer-AND-purchase …
  - Divide and conquer

Write testing programs

```
(defrule list-all-purchase-strict

    (object (is-a http...#Purchase) (OBJECT ?Purch1) (http...#hasBuyer
            ?PersonObject1) (http...#hasBook ?BookObjectA))

    (object (is-a http...#Book) (OBJECT ?BookObjectA) (http...#hasTitle ?titleA))

    (object (is-a http...#Customer) (OBJECT ?PersonObject1)
            (http...#hasName ?Person1) (http...#hasMadePurchase $? ?Purch1  $?))

        =>
        (printout t "Purchase: " ?Person1 " has bought " ?titleA   crlf)
)
```

What about we remove this constraint?

Write testing programs

```
(defrule list-all-purchase-easy

      (object (is-a http...#Purchase) (OBJECT ?Purch1) (http...#hasBuyer
               ?PersonObject1) (http...#hasBook ?BookObjectA))

      (object (is-a http...#Book) (OBJECT ?BookObjectA) (http...#hasTitle ?titleA))

      (object (is-a http...#Customer) (OBJECT ?PersonObject1)
               (http...#hasName ?Person1)  )

          =>
          (printout t "Purchase: " ?Person1 " has bought " ?titleA   crlf)
)
```

Divide and conquer

```
(defrule recommend-1
   (object (is-a http...#Person) (OBJECT ?PersonObject1) (http...#hasName ?Person1) (http...#hasMadePurchase $? ?Purch1 $?
   (object (is-a http...#Purchase) (OBJECT ?Purch1) (http...#hasBuyer ?PersonObject1) (http...#hasBook ?BookObjectA))
   (object (is-a http...#Person) (OBJECT ?PersonObject2) (http...#hasName ?Person2) (http...#hasMadePurchase $? ?Purch2 $?
   (object (is-a http...#Person) (OBJECT ?PersonObject2) (http...#hasName ?Person2) (http...#hasMadePurchase $? ?Purch3 $?
   (test (neq ?Person1 ?Person2))
   (test (neq ?Purch2 ?Purch3))
   (object (is-a http...#Purchase) (OBJECT ?Purch2) (http...#hasBuyer ?PersonObject2) (http...#hasBook ?BookObjectA))
   (object (is-a http...#Purchase) (OBJECT ?Purch3) (http...#hasBuyer ?PersonObject2) (http...#hasBook ?BookObjectB))
   (not (object (is-a http...#Purchase) (http...#hasBuyer ?PersonObject1) (http...#hasBook ?BookObjectB)))
   (not (object (is-a http...#Recommendation) (http...#hasBuyer ?PersonObject1) (http...#hasBook ?BookObjectB)))
   (object (is-a http...#Book) (OBJECT ?BookObjectA) (http...#hasTitle ?titleA))
   (object (is-a http...#Book) (OBJECT ?BookObjectB) (http...#hasTitle ?titleB))
   =>
   (printout t "Recommendation for " ?Person1 ": other people that bought \"" ?titleA "\" also bought \"" ?titleB "\"" crlf)
   (make-instance of http...#Recommendation (http...#hasBuyer ?PersonObject1) (http...#hasBook ?BookObjectB))
)
```