Machine Reasoning
Day 3

Course Manager: GU Zhan (Sam)

zhan.gu@nus.edu.sg

Machine Reasoning

Day 3

Day 3 Agenda

3.1 Deductive Reasoning by Logical Inference

- 3.1.1 Rules & Logical Inference
- 3.1.2 Conflict Resolution
- 3.1.3 Exercise

3.2 Reasoning under Uncertainty

- 3.2.1 Analogical Reasoning (Similarity)
- 3.2.2 Fuzzy Logic Reasoning
- 3.2.3 Abductive Reasoning (Probability)
- 3.2.4 Exercise

3.3 Deductive Reasoning (under Uncertainty) [Workshop]

- 3.3.1 Analogical Reasoning (Similarity)
- 3.3.2 Fuzzy Logic Reasoning
- 3.3.3 Abductive Reasoning (Probability)
- 3.3.4 Workshop Submission

3.1 Deductive Reasoning by Logical Inference

3.1.1 Rules & Logical Inference

3.1.2 Conflict Resolution

3.1.3 Exercise

3.1 Deductive Reasoning by Logical Inference

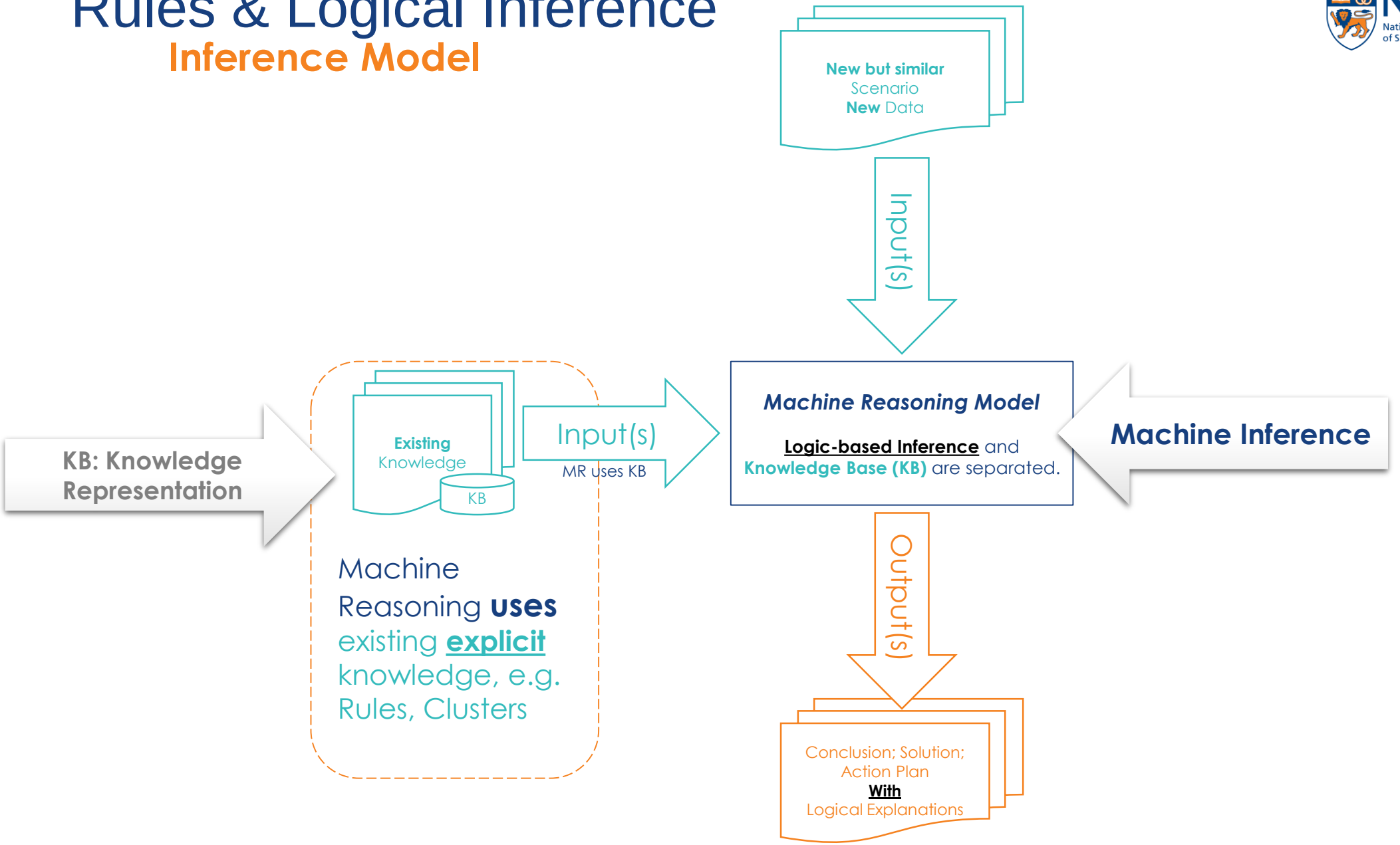
3.1.1 Rules & Logical Inference

3.1.2 Conflict Resolution

3.1.3 Exercise

Rules & Logical Inference

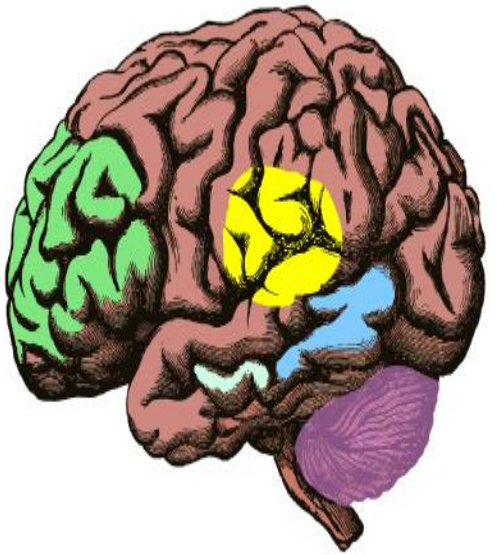
Inference Model



Rules & Logical Inference

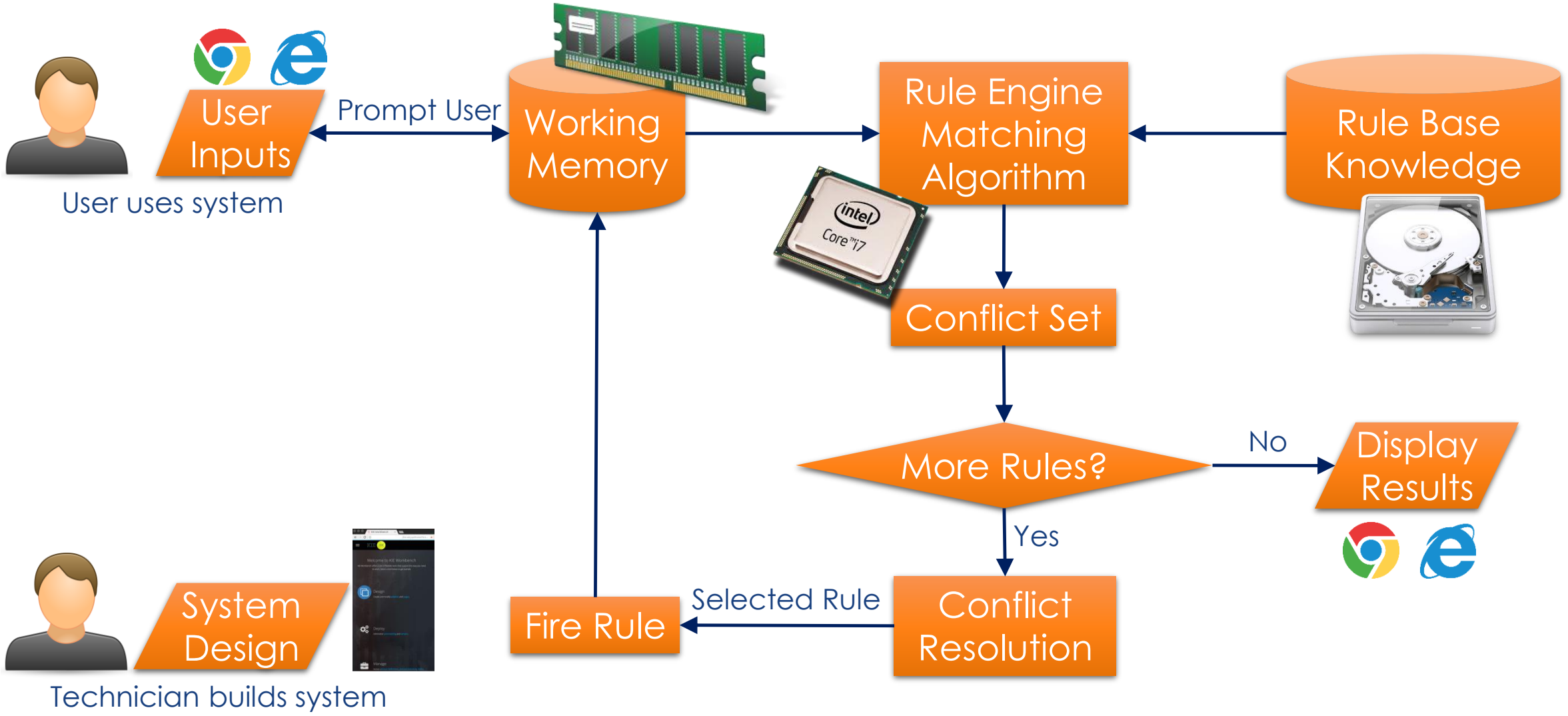
[Psychology] Information Processing Theory

Information Processing Model



Rules & Logical Inference

[AI] Recognise-Act Control Cycle



Rules & Logical Inference

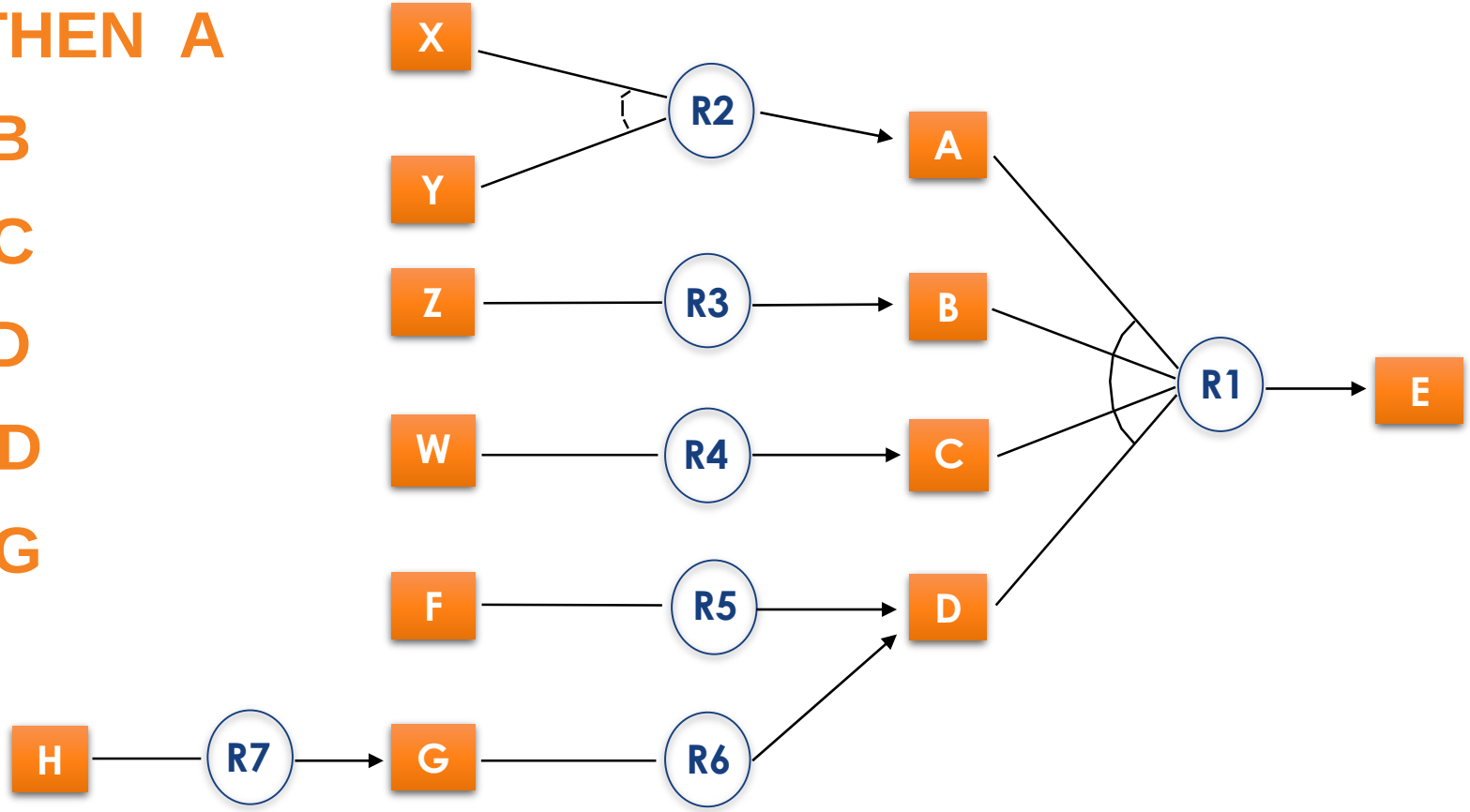
Three main components of Rule Based System (RBS)

- **A set of domain rules/facts (Knowledge Base)** **[long term memory]**
  - Rules/Facts represent a chunk of problem-solving knowledge
- **A working memory (WM)** **[short term memory]**
  - Contains new observation's Data (current state of program execution)
- **A rule/inference engine** **[pattern matching controller]**
  - A computational system that implements the control strategy and applies (fire) the rules
  - The patterns in WM are matched against the conditions of the rules
  - Matched rules are called the conflict set
  - The control strategy determines the order in which the rules are fired and resolves any rule conflicts
  - Uses the Recognise-Act cycle

Rules & Logical Inference

Rules form a Search Tree

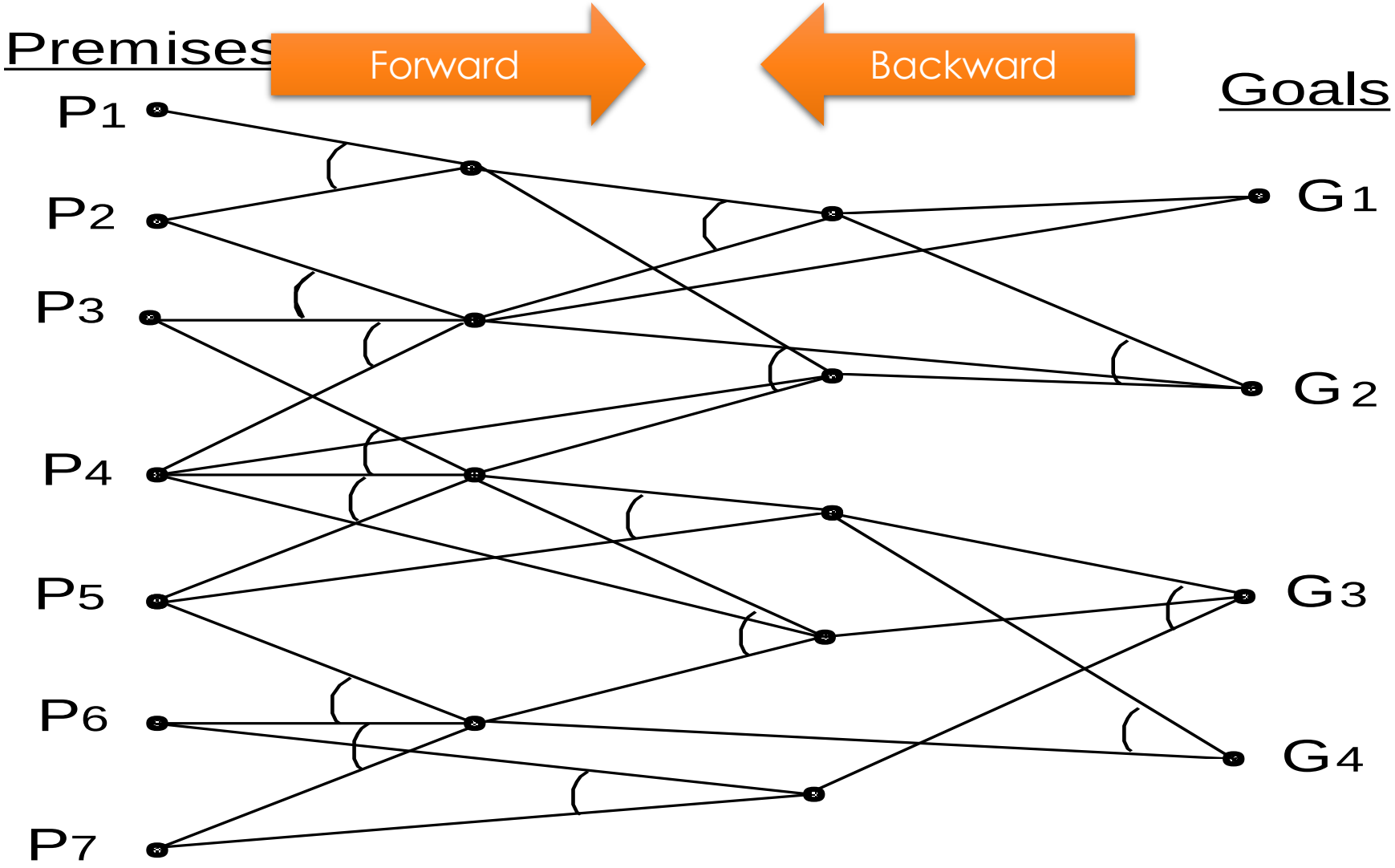
- **R1** IF A and B and C and D THEN E
- **R2** IF X and Y THEN A
- **R3** IF Z THEN B
- **R4** IF W THEN C
- **R5** IF F THEN D
- **R6** IF G THEN D
- **R7** IF H THEN G



- Inference strategy is known as "**chaining**".

Rules & Logical Inference

Forward Chaining & Backward Chaining



Rules & Logical Inference

Forward Chaining Example

Rule 1:

IF	the patient has a sore throat
AND	we suspect a Bacterial infection
THEN	we believe the Illness is a strep throat

Rule 2:

IF	the patient's temperature is > 100
THEN	the patient has a fever

Rule 3:

IF	the patient has been sick for over a month
AND	the patient has a fever
THEN	we suspect a bacterial infection

Facts:	Temperature = 104°F (40°C)	F1
	Patient has been sick for 2 months	F2
	Patient has a sore throat	F3

Rules & Logical Inference

Forward Chaining Example

Rule 1:

IF the patient has a sore throat
AND we suspect a Bacterial infection
THEN we believe the Illness is a strep throat

Rule 2:

IF the patient's temperature is > 100
THEN the patient has a fever

Rule 3:

IF the patient has been sick for over a month
AND the patient has a fever
THEN we suspect a bacterial infection

Facts:

Temperature = 104°F (40°C)	F1
Patient has been sick for 2 months	F2
Patient has a sore throat	F3
patient has a fever	F4

Rules & Logical Inference

Forward Chaining Example

Rule 1:

IF the patient has a sore throat
AND we suspect a Bacterial infection
THEN we believe the Illness is a strep throat

Rule 2:

IF the patient's temperature is > 100
THEN the patient has a fever

Rule 3:

IF the patient has been sick for over a month
AND the patient has a fever
THEN we suspect a bacterial infection

Facts:

Temperature = 104°F (40°C)	F1
Patient has been sick for 2 months	F2 ←
Patient has a sore throat	F3
patient has a fever	F4 ←
suspect a bacterial infection →	F5

Rules & Logical Inference

Forward Chaining Example

Rule 1:

IF the patient has a sore throat
AND we suspect a Bacterial infection
THEN we believe the illness is a strep throat

Rule 2:

IF the patient's temperature is > 100
THEN the patient has a fever

Rule 3:

IF the patient has been sick for over a month
AND the patient has a fever
THEN we suspect a bacterial infection

Facts:

Temperature = 104°F (40°C)	F1
Patient has been sick for 2 months	F2
Patient has a sore throat	F3 ←
patient has a fever	F4
suspect a bacterial infection	F5 ←
illness is a strep throat →	F6

Rules & Logical Inference

Forward Chaining Example

Initial state of working memory WM: [F1, F2, F3]

Rule-2 [F1]

Add F4: “patient has a fever”

Rule-3 [F2, F4]

Add F5: “suspect a bacterial infection”

Rule-1 [F3, F5]

Add F6: “illness is a strep throat”

No more rules to fire → halt

Conclusion → illness is a strep throat

Rules & Logical Inference

Forward Chaining Definition

- **A forward chaining system**
  - Begin with a set of facts in the Working Memory, then apply rules to generate new facts until the desired goal is reached.
- **Rules whose premise (IF..) is known to be true are fired, and their conclusions (THEN..) are declared true.**
  - This process continues until no more rules can be triggered/fired. The system then reports its conclusions.

Rules & Logical Inference

Backward Chaining Example

Rule 1:

IF the patient has a sore throat
AND we suspect a Bacterial infection
THEN we believe the illness is a strep throat

Rule 2:

IF the patient's temperature is > 100°F
THEN the patient has a fever

Rule 3:

IF the patient has been sick for over a month
AND the patient has a fever
THEN we suspect a bacterial infection

F1: Temperature = 104°F (40°C)
F2: Patient has been sick for 2 months
F3: Patient has a sore throat

Rules & Logical Inference

Backward Chaining Example

Rule 1:

IF the patient has a sore throat
AND we suspect a Bacterial infection
THEN we believe the illness is a strep throat

Rule 2:

IF the patient's temperature is > 100°F
THEN the patient has a fever

Rule 3:

IF the patient has been sick for over a month
AND the patient has a fever
THEN we suspect a bacterial infection

F1: Temperature = 104°F (40°C)
F2: Patient has been sick for 2 months
F3: Patient has a sore throat
F4: Is the illness a strep throat?

Rules & Logical Inference

Backward Chaining Example

Rule 1:

IF the patient has a sore throat
AND we suspect a Bacterial infection
THEN we believe the illness is a strep throat

Rule 2:

IF the patient's temperature is > 100°F
THEN the patient has a fever

Rule 3:

IF the patient has been sick for over a month
AND the patient has a fever
THEN we suspect a bacterial infection

F1: Temperature = 104°F (40°C)
F2: Patient has been sick for 2 months
F3: Patient has a sore throat
F4: Is the illness a strep throat?

Rules & Logical Inference

Backward Chaining Example

Rule 1:

IF the patient has a sore throat ☑
AND we suspect a Bacterial infection [?]
THEN we believe the illness is a strep throat

Rule 2:

IF the patient's temperature is > 100°F
THEN the patient has a fever

Rule 3:

IF the patient has been sick for over a month
AND the patient has a fever
THEN we suspect a bacterial infection

F1: Temperature = 104°F (40°C)
F2: Patient has been sick for 2 months
F3: Patient has a sore throat
F4: Is the illness a strep throat?

Rules & Logical Inference

Backward Chaining Example

Rule 1:

IF the patient has a sore throat ☑
AND we suspect a Bacterial infection [?]
THEN we believe the illness is a strep throat

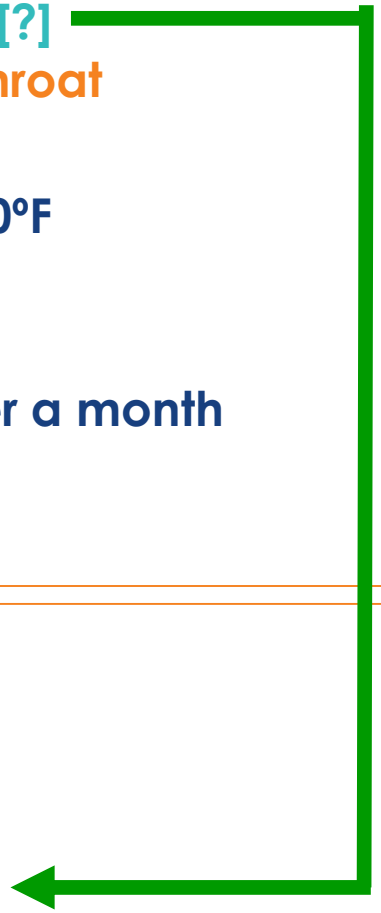
Rule 2:

IF the patient's temperature is > 100°F
THEN the patient has a fever

Rule 3:

IF the patient has been sick for over a month
AND the patient has a fever
THEN we suspect a bacterial infection

F1: Temperature = 104°F (40°C)
F2: Patient has been sick for 2 months
F3: Patient has a sore throat
F4: Is the illness a strep throat?
F5: Do we suspect a bacterial infection?



Rules & Logical Inference

Backward Chaining Example

Rule 1:

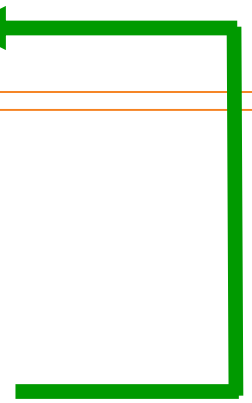
IF the patient has a sore throat ☑
AND we suspect a Bacterial infection [?]
THEN we believe the illness is a strep throat

Rule 2:

IF the patient's temperature is > 100°F
THEN the patient has a fever

Rule 3:

IF the patient has been sick for over a month
AND the patient has a fever
THEN we suspect a bacterial infection



F1: Temperature = 104°F (40°C)
F2: Patient has been sick for 2 months
F3: Patient has a sore throat
F4: Is the illness a strep throat?
F5: Do we suspect a bacterial infection?

Rules & Logical Inference

Backward Chaining Example

Rule 1:

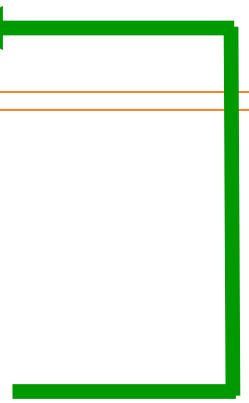
IF the patient has a sore throat ☑
AND we suspect a Bacterial infection [?]
THEN we believe the illness is a strep throat

Rule 2:

IF the patient's temperature is > 100°F
THEN the patient has a fever

Rule 3:

IF the patient has been sick for over a month ☑
AND the patient has a fever [?]
THEN we suspect a bacterial infection



F1: Temperature = 104°F (40°C)
F2: Patient has been sick for 2 months
F3: Patient has a sore throat
F4: Is the illness a strep throat?
F5: Do we suspect a bacterial infection?

Rules & Logical Inference

Backward Chaining Example

Rule 1:

IF the patient has a sore throat ☑
AND we suspect a Bacterial infection [?]
THEN we believe the illness is a strep throat

Rule 2:

IF the patient's temperature is > 100°F
THEN the patient has a fever

Rule 3:

IF the patient has been sick for over a month ☑
AND the patient has a fever [?]
THEN we suspect a bacterial infection

F1: Temperature = 104°F (40°C)
F2: Patient has been sick for 2 months
F3: Patient has a sore throat
F4: Is the illness a strep throat?
F5: Do we suspect a bacterial infection?
F6: Does the patient have a fever?

Rules & Logical Inference

Backward Chaining Example

Rule 1:

IF the patient has a sore throat ☑
AND we suspect a Bacterial infection [?]
THEN we believe the illness is a strep throat

Rule 2:

IF the patient's temperature is > 100°F
THEN the patient has a fever

Rule 3:

IF the patient has been sick for over a month ☑
AND the patient has a fever [?]
THEN we suspect a bacterial infection

F1: Temperature = 104°F (40°C)
F2: Patient has been sick for 2 months
F3: Patient has a sore throat
F4: Is the illness a strep throat?
F5: Do we suspect a bacterial infection?
F6: Does the patient have a fever?

Rules & Logical Inference

Backward Chaining Example

Rule 1:

IF the patient has a sore throat ☑
AND we suspect a Bacterial infection [?]
THEN we believe the illness is a strep throat

Rule 2:

IF the patient's temperature is > 100°F ☑
THEN the patient has a fever

Rule 3:

IF the patient has been sick for over a month ☑
AND the patient has a fever [?]
THEN we suspect a bacterial infection

Proved: Patient has a strep throat.

F1: Temperature = 104°F (40°C)
F2: Patient has been sick for 2 months
F3: Patient has a sore throat
F4: Is the illness a strep throat?
F5: Do we suspect a bacterial infection?
F6: Does the patient have a fever?

Rules & Logical Inference

Backward Chaining Example

- **Initial state of working memory WM: [Facts + Hypothesis]**

Rule-1 [Hypothesis F4]

Goal: we suspect a strep throat? **{new goal to pursue Add F4}**

Check: patient has a sore throat? **{proved by F3}**

Check: we suspect bacterial infection? **{new goal to pursue Add F5}**

Rule-3 [F5]

Check: patient has been sick for over a week? **{proved by F2}**

Check: patient has a fever? **{new goal to pursue Add F6}**

Rule-2 [F6]

Check: patient's temperature is > 100ºF (37.8ºC)? **{proved by F1}**

No more rules to fire {all proved} → halt

Conclusion → yes, illness is a strep throat

Rules & Logical Inference

Backward Chaining Definition

- **A backward chaining inference engine starts from a goal or hypothesis.**
  - It works through the rules trying to match the goal with the action clauses (THEN part) of a rule.
  - When a match is found, the condition clauses (IF part) of the matching rule become "sub-goals".
  - The cycle is repeated until a verifiable set of condition clauses is found.

Rules & Logical Inference

Forward Chaining vs. Backward Chaining

- **FC is data-driven**
  - The focus of attention starts form known data & business rules
    - e.g., object recognition, routine decisions
    - May do lots of work that is irrelevant to the goal
- **BC is goal-driven**
  - Appropriate for problem-solving & investigation
    - e.g., Where are my keys? How do I get into a PhD program?

[Optional] Rules & Logical Inference

Rules : Drools inference engine

a driving license application.

```
public class Applicant {
    private String name;
    private int age;
    private boolean valid;
    // getter and setter methods here
}
```

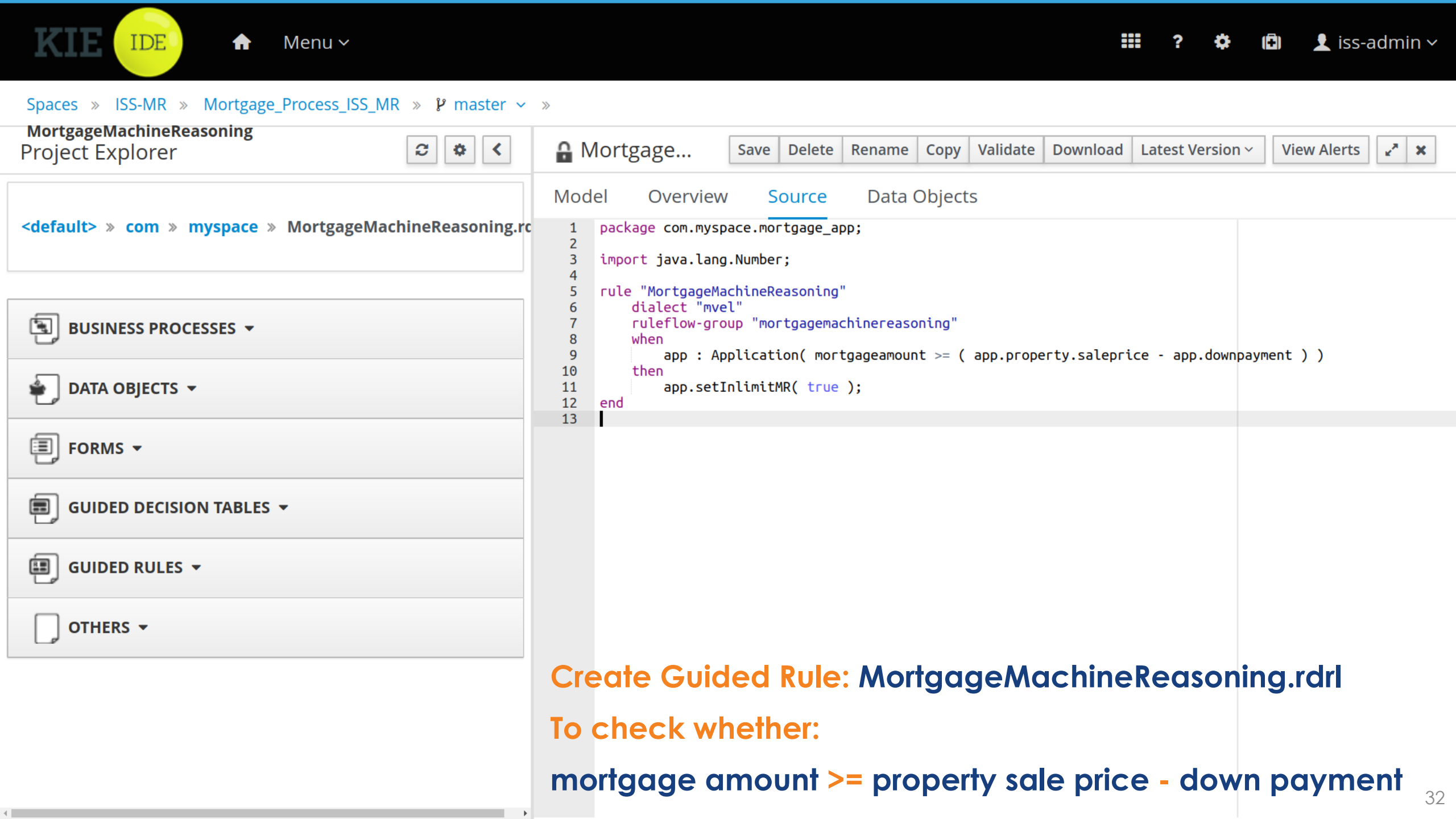
Now that we have our data model we can write our first rule. We assume that the application uses rules to reject invalid applications. As this is a simple validation use case we will add a single rule to disqualify any applicant younger than 18.

```
package com.company.license

rule "Is of valid age"
when
    $a : Applicant( age < 18 )
then
    $a.setValid( false );
end
```

To make the Drools engine aware of data, so it can be processed against the rules, we have to ***insert*** the data, much like with a database. When the Applicant instance is inserted into the Drools engine it is evaluated against the constraints of the rules, in this case just two constraints for one rule. We say ***two*** because the type **Applicant** is the first object type constraint, and **age < 18** is the second field constraint. **An object type constraint plus its zero or more field constraints is referred to as a pattern**. When an inserted instance satisfies both the object type constraint and all the field constraints, it is said to be matched. The **$a** is a binding variable which permits us to reference the matched object in the consequence. There its properties can be updated. The dollar character ('$') is optional, but it helps to differentiate variable names from field names. The process of matching patterns against the inserted data is, not surprisingly, often referred to as ***pattern matching***.

Source: https://docs.jboss.org/drools/release/latest/drools-docs/html_single/index.html#_methods_versus_rules



Create Guided Rule: MortgageMachineReasoning.rdrl

To check whether:

mortgage amount >= property sale price - down payment

3.1 Deductive Reasoning by Logical Inference

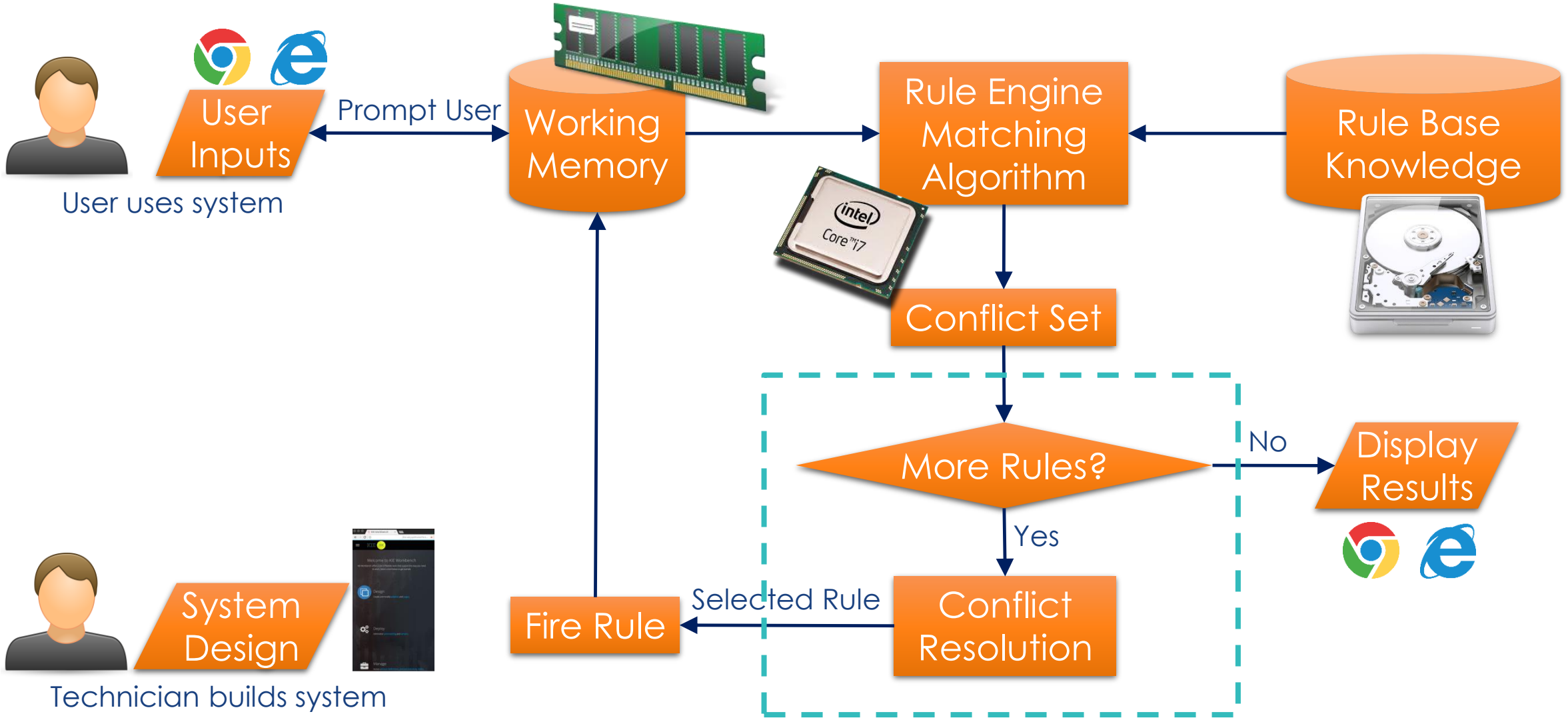
3.1.1 Rules & Logical Inference

3.1.2 Conflict Resolution

3.1.3 Exercise

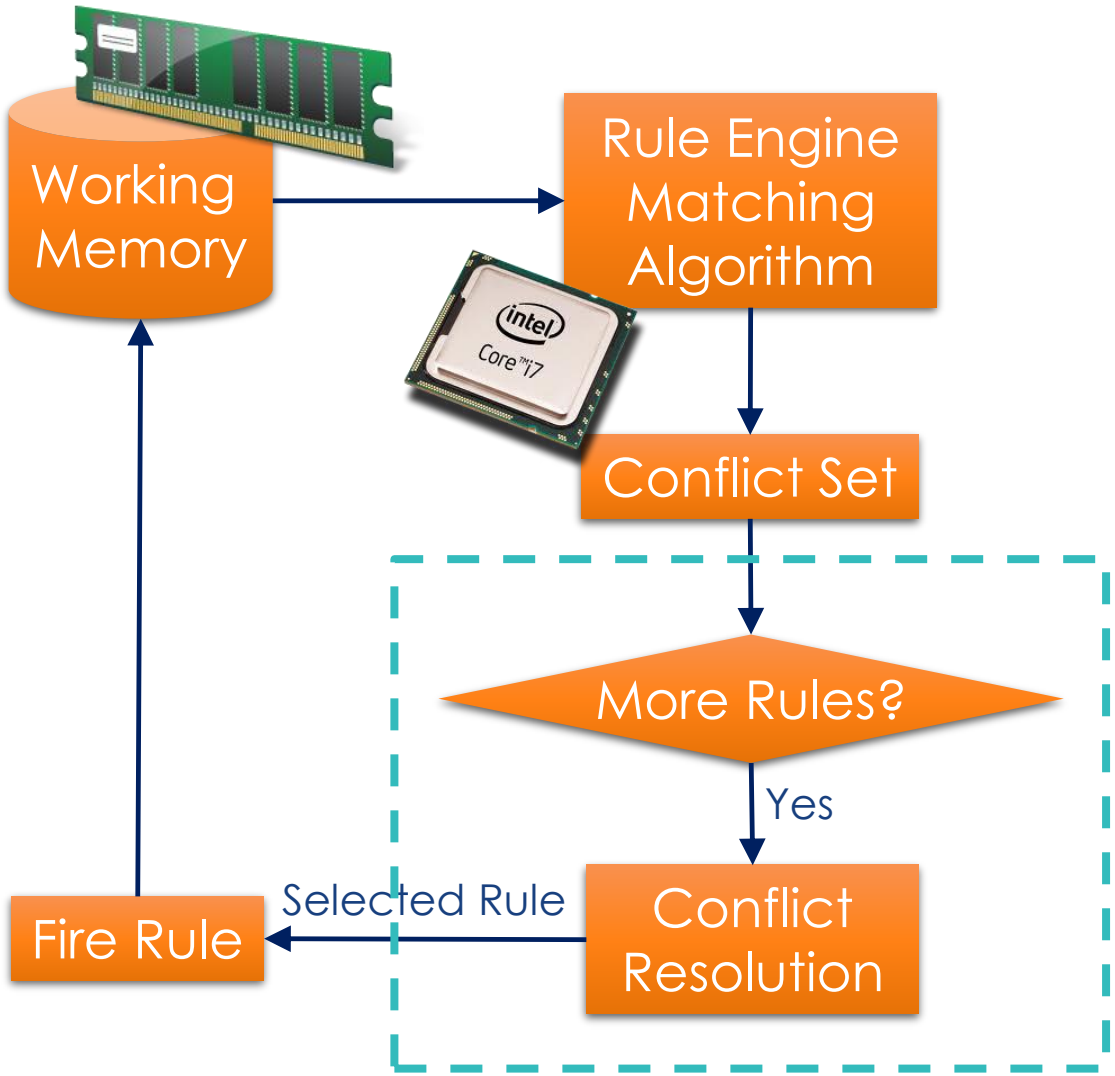
Conflict Resolution

[AI] Recognise-Act Control Cycle



Conflict Resolution

- **Conflict Set**
  - More than one rule can fire based on the facts in WM. That is, the fact in WM can match more than one rule at a specific time. The matched activated rules represents a **conflict set**.
- **Conflict Resolution**
  - A method for choosing a rule to fire when more than one rule can be fired in a given cycle.



Conflict Resolution

Example

Rule 1: **IF** I have at least $20 AND Man-United is playing today
THEN I should go to the football game.

Rule 2: **IF** it is raining today AND I don't have school
THEN I should stay home

Rule 3: **IF** I have at least $20
THEN I should go to the cinema.

Rule 4: **IF** I should go to the cinema
THEN I should call my friends

Initial Facts: **Fact-1**: I have $20
Fact-2: Man-United is playing today

Conflict Set: **<R1: Fact-1, Fact-2>, <R3: Fact-1>**

Conflict Resolution Strategy

- **The order in which the rule fires depends on facts in Working Memory, (in general) not the order of rules.**
- **Rule firing priority:**
  - Default is last in first out (**LIFO**) based on facts in WM
  - Attach priority to rules, and select the rule with the highest priority (**Salience**)
  - Order the facts by the length of time they have been in working memory, and select the most recent. (**Recency**)
  - Select rules which required the lowest/highest number of facts/rule-conditions (**Specificity**).

[Optional] Conflict Resolution

Strategy : Drools inference engine control parameters

- **Salience / Specificity / Recency / LIFO / FIFO**

  Individual rule's priority **in Agenda**

- **AgendaGroup**

  It allow you to place rules into groups, and to place those groups onto a stack (**rule set's priority**). The stack has push/pop behaviour.

- **ActivationGroup**

  It is a set of rules bound together by the same "activation-group" rule attribute. In this group **only one rule can fire**, and after that rule has fired all the other rules are cancelled from the agenda.

- **RuleFlowGroup**

  It is a group of rules associated by the "ruleflow-group" rule attribute. These rules can only fire when the group is activated. (**jBPM Business Rule Task**)

```
rule "Print balance for AccountPeriod"
        salience -50
    when
        ap : AccountPeriod()
        acc : Account()
    then
        System.out.println( acc.accountNo + " : " + acc.balance );
end
```

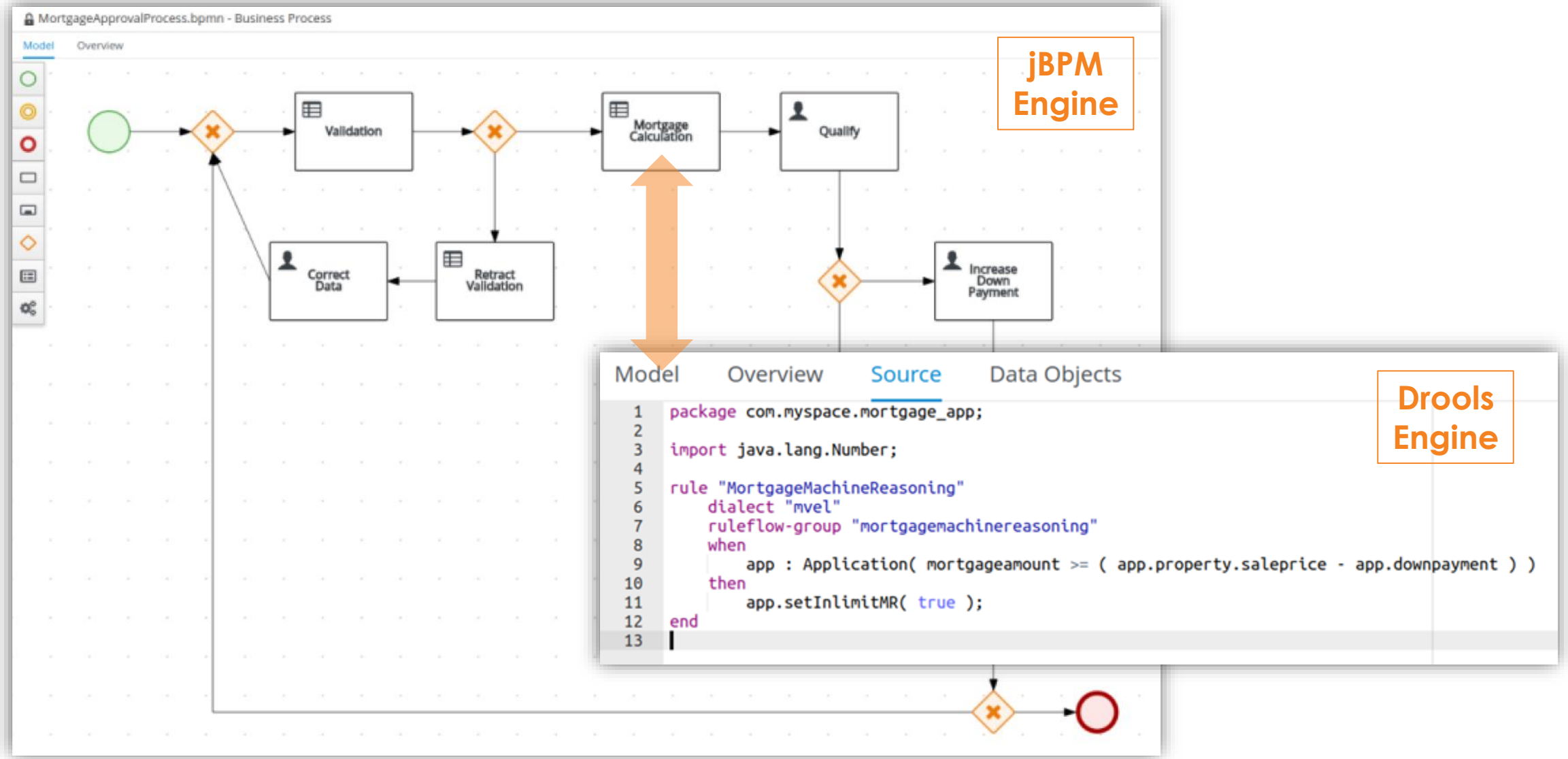
```
rule "increase balance for credits"
  agenda-group "calculation"
when
  ap : AccountPeriod()
  acc : Account( $accountNo : accountNo )
  CashFlow( type == CREDIT,
            accountNo == $accountNo,
            date >= ap.start && <= ap.end,
            $amount : amount )
then
  acc.balance  += $amount;
end
```

```
rule "Print balance for AccountPeriod"
  agenda-group "report"
when
  ap : AccountPeriod()
  acc : Account()
then
  System.out.println( acc.accountNo +
                      " : " + acc.balance );
end
```

Source: https://docs.jboss.org/drools/release/latest/drools-docs/html_single/index.html#_conflict_resolution_2

[Optional] Conflict Resolution

KIE BPMS/BRMS Suite – Mortgage application systems



3.1 Deductive Reasoning by Logical Inference

3.1.1 Rules & Logical Inference

3.1.2 Conflict Resolution

3.1.3 Exercise

[Exercise] Deductive Reasoning by Logical Inference

Refer to <Guess the Animal> case, work out the following problems:

- **Q1: Given these facts in Working Memory:**

  the animal gives milk, the animal eats grass, the animal has long legs, the animal has a long neck

  Goal: To establish by **forward chaining** that the animal is a giraffe. If you are not able to establish this, what is the rule(s) that you can add into the Knowledge/Rule Base to successfully perform the chaining (inference)?

- **Q2: Given these facts in Working Memory:**

  the animal has hair, the animal has claws, the animal has pointed teeth, the animal's eyes point forward, the animal has a tawny color, the animal has dark spots

  Goal: To establish by **backward chaining** that the animal is a cheetah

[Exercise] Deductive Reasoning by Logical Inference

<Guess the Animal> Knowledge Base:

1. animals with **hair** as their **body covering** are **mammals**
2. animals that **feed** their young with **milk** are **mammals**
3. animals with **feathers** as their **body covering** are **birds**
4. animals that **fly** and **reproduce** by **eggs** are **birds**
5. **mammals** that **eat meat** are **carnivores**
6. **mammals** with pointed **teeth**, **claws** on their **feet**, and **eyes** that **point forward** are **carnivores**
7. **mammals** that **eat grass** are **herbivores**
8. **mammals** with **hooves** on their **feet** are **herbivores**
9. **carnivores** that have a **tawny colour** and **dark spots** as their **marking** are **cheetahs**
10. **carnivores** that have a **tawny colour** and **dark stripes** as their **marking** are **tigers**
11. **herbivores** that have a **tawny colour** and **dark spots** as their **marking** and **long necks** are **giraffes**
12. **herbivores** that have a **black and white colour** are **zebras**
13. **birds** that **walk** and are **black and white** and have a **long neck** are **ostriches**
14. **birds** that **swim** and are **black and white** are **penguins**
15. **birds** that **fly** and are **black and white** are **albatrosses**

[Exercise] Deductive Reasoning by Logical Inference

1. WHEN BodyCoverType = HasHair THEN AnimalType = Mammal
2. WHEN FeedType = FeedMilk THEN AnimalType = Mammal
3. WHEN BodyCoverType = HasFeather THEN AnimalType = Bird
4. WHEN MoveType = CanFly AND ReproduceType = LayEgg THEN AnimalType = Bird
5. WHEN AnimalType = Mammal AND FeedType = EatMeat THEN AnimalType = Carnivore
6. WHEN AnimalType = Mammal AND ToothType = HasPointedTeeth AND FootType = HasClaws AND EyeType = HasForwardEyes THEN AnimalType = Carnivore
7. WHEN AnimalType = Mammal AND FeedType = EatGrass THEN AnimalType = Herbivore
8. WHEN AnimalType = Mammal AND FootType = HasHooves THEN AnimalType = Herbivore
9. WHEN AnimalType = Carnivore AND ColorType = HasColorTawny AND MarkingType = HasDarkSpots THEN AnimalName = Cheetah
10. WHEN AnimalType = Carnivore AND ColorType = HasColorTawny AND MarkingType = HasDarkStripes THEN AnimalName = Tiger
11. WHEN AnimalType = Herbivore AND ColorType = HasColorTawny AND MarkingType = HasDarkSpots AND NeckType = HasLongNeck THEN AnimalName = Giraffe
12. WHEN AnimalType = Herbivore AND ColorType = HasColorBlackWhite THEN AnimalName = Zebra
13. WHEN AnimalType = Bird AND MoveType = CanWalk AND ColorType = HasColorBlackWhite AND NeckType = HasLongNeck THEN AnimalName = Ostrich
14. WHEN AnimalType = Bird AND MoveType = CanWwim AND ColorType = HasColorBlackWhite THEN AnimalName = Penguin
15. WHEN AnimalType = Bird AND MoveType = CanFly AND ColorType = HasColorBlackWhite THEN AnimalName = Albatross

[Exercise] Deductive Reasoning by Logical Inference

X is an (instance of) animal (class).

1. HasHair(X) → Mammal(X)
2. FeedMilk(X) → Mammal(X)
3. HasFeather(X) → Bird(X)
4. CanFly(X) ^ LayEgg(X) → Bird(X)
5. Mammal(X) ^ EatMeat(X) → Carnivore(X)
6. Mammal(X) ^ HasPointedTeeth(X) ^ HasClaws(X) ^ HasForwardEyes(X) → Carnivore(X)
7. Mammal(X) ^ EatGrass(X) → Herbivore(X)
8. Mammal(X) ^ HasHooves(X) → Herbivore(X)
9. Carnivore(X) ^ HasColorTawny(X) ^ HasDarkSpots(X) → Cheetah(X)
10. Carnivore(X) ^ HasColorTawny(X) ^ HasDarkStripes(X) → Tiger(X)
11. Herbivore(X) ^ HasColorTawny(X) ^ HasDarkSpots(X) ^ HasLongNeck(X) → Giraffe(X)
12. Herbivore(X) ^ HasColorBlackWhite(X) → Zebra(X)
13. Bird(X) ^ CanWalk(X) ^ HasColorBlackWhite(X) ^ HasLongNeck(X) → Ostrich(X)
14. Bird(X) ^ CanSwim(X) ^ HasColorBlackWhite(X) → Penguin(X)
15. Bird(X) ^ CanFly(X) ^ HasColorBlackWhite(X) → Albatross(X)

[Exercise] Deductive Reasoning by Logical Inference

- **Knowledge Base (KB)**
  - Rule sets plus below Facts:
  - HasHair(X)
  - HasClaws(X)
  - HasPointedTeeth(X)
  - HasForwardEyes(X)
  - HasColorTawny(X)
  - HasDarkSpots(X)
- **Clause form conversion : $p \rightarrow q \equiv \neg p \vee q$**
- **Hypothesis to prove is : α = Cheetah(X)**
- **Refutation of hypothesis is : $\neg\alpha = \neg$Cheetah(X)**

3.2 Reasoning under Uncertainty

3.2.1 Analogical Reasoning (Similarity)

3.2.2 Fuzzy Logic Reasoning

3.2.3 Abductive Reasoning (Probability)

3.2.4 Exercise

3.2 Reasoning under Uncertainty

3.2.1 Analogical Reasoning (Similarity)

3.2.2 Fuzzy Logic Reasoning

3.2.3 Abductive Reasoning (Probability)

3.2.4 Exercise

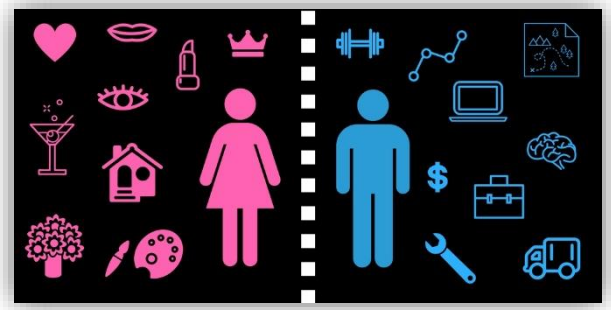
Analogical Reasoning (Similarity based method)

- Analogical reasoning compare similarity based on subject's observed **features/properties** with other subjects. It relates to nearest neighbor approaches, in that it is data-based rather than abstraction-based. Example method like k-nearest neighbors algorithm (k-NN): finding the most similar K subjects (in feature space), then use **majority voting** to make decision (classification/prediction) for current subject of interest.

Common Forms of Reasoning

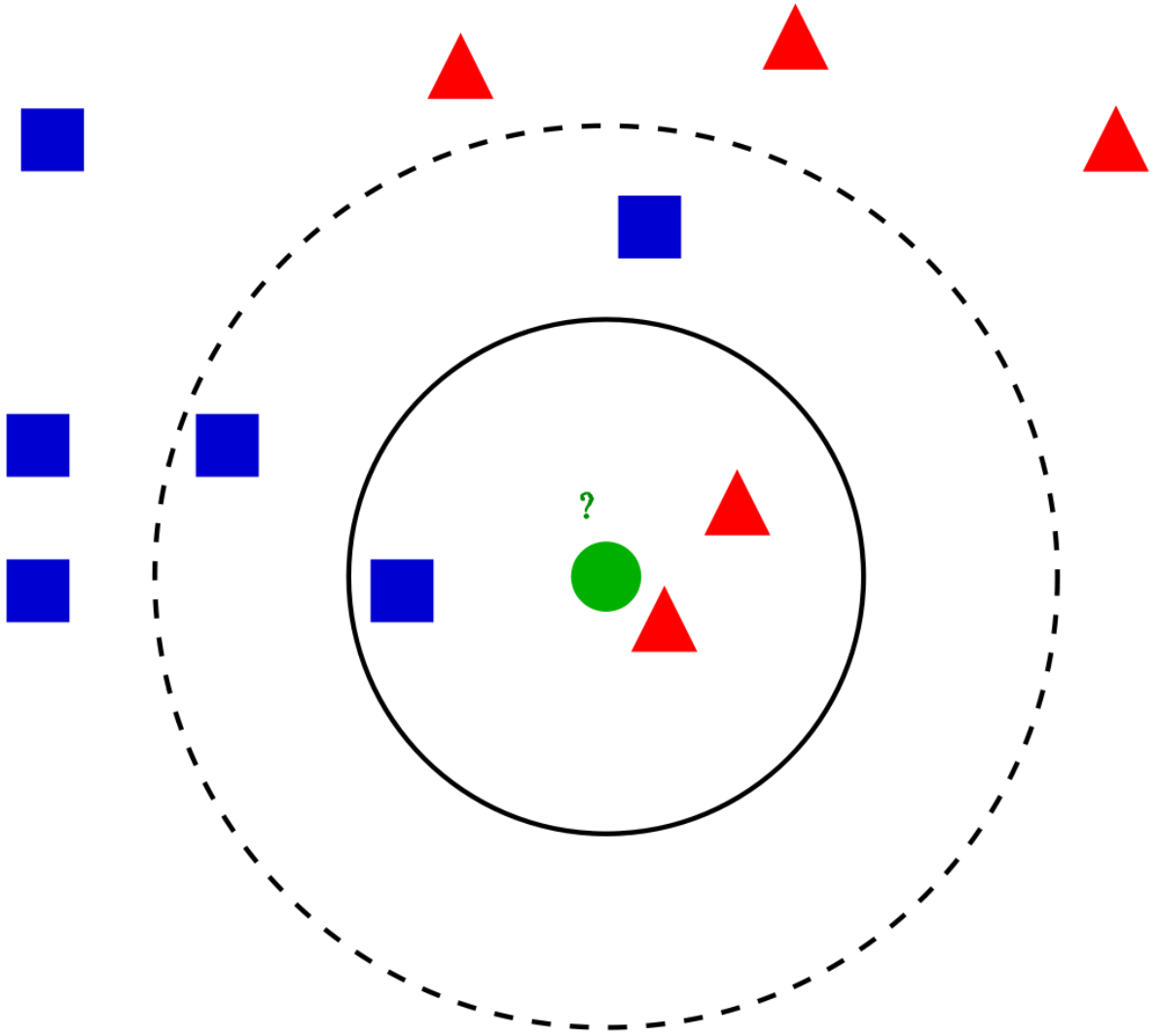
Analogical Reasoning

- Similarity based reasoning; Case based; K nearest neighbour; (including customer profiling for recommendation; even stereotyping)



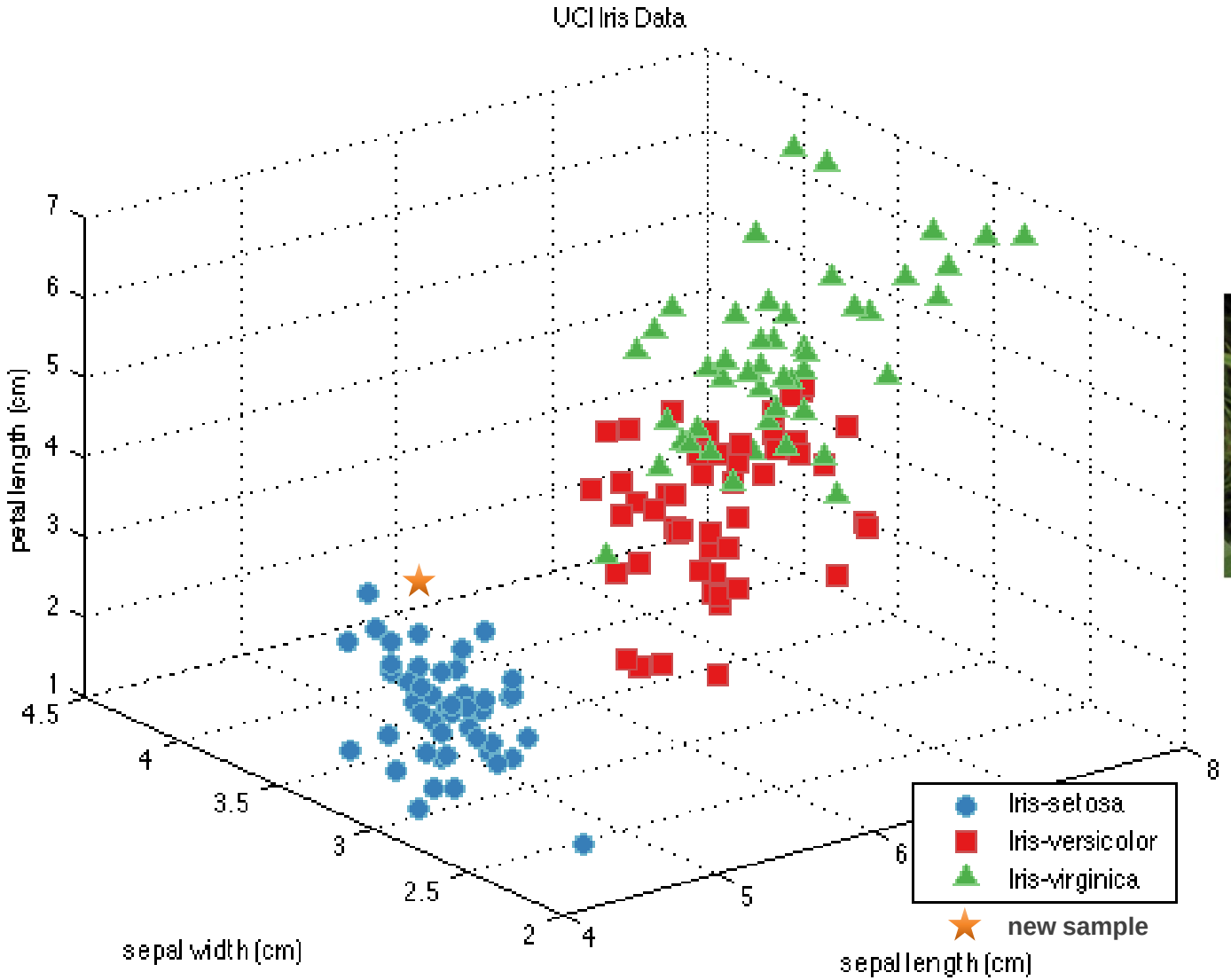
Analogical Reasoning (Similarity)

Best dish of Singapore: Chicken rice or Duck noodle?



Source https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#/media/File:KnnClassification.svg
Source https://cdn.cnn.com/cnnnext/dam/assets/170227111355-3-singapore-michelin-hawker-stall.jpg
Source https://www.healthxchange.sg/sites/hexassets/Assets/food-nutrition/chinese-hawker-foods-updated-pics/best-worst-singapore-hawker-chinese-food-duck-rice-fishball-noodle-b.jpg

Analogical Reasoning (Similarity)



Iris Versicolor **Iris Setosa** **Iris Virginica**

Source https://prasanth-ntu.github.io/html/ML-Course/Intro-to-ML-with-Python/images/iris-machinelearning.png

3.2 Reasoning under Uncertainty

3.2.1 Analogical Reasoning (Similarity)

3.2.2 Fuzzy Logic Reasoning

3.2.3 Abductive Reasoning (Probability)

3.2.4 Exercise

Fuzzy Logic (FL)

- Fuzzy Logic are **measures of inclination (degree of truth)** towards a **linguistic** concept/word, which lacks a **rigorous definition**.

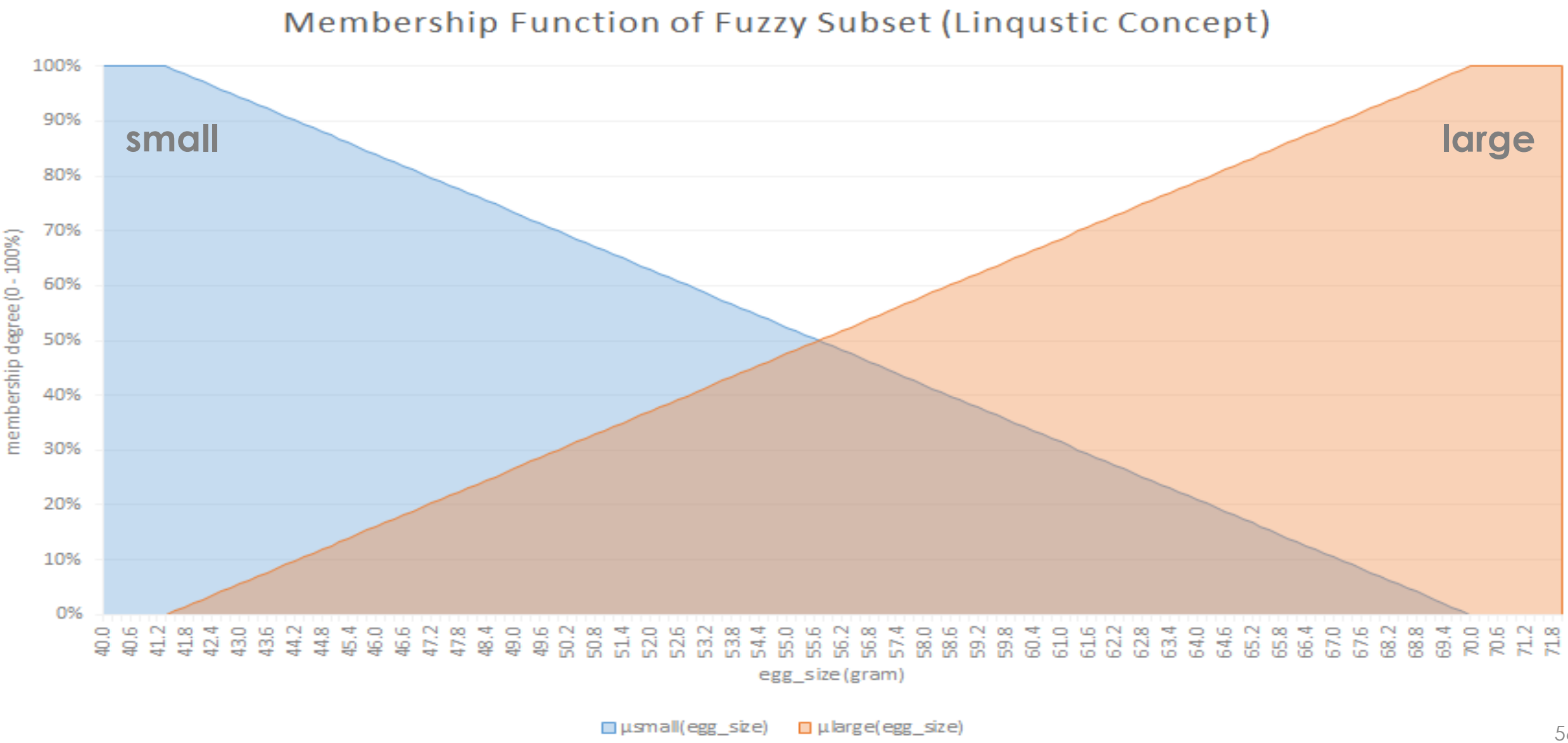
Source: https://www.youtube.com/watch?v=J_Q5X0nTmrA

Fuzzy Logic
Fuzzy Rules

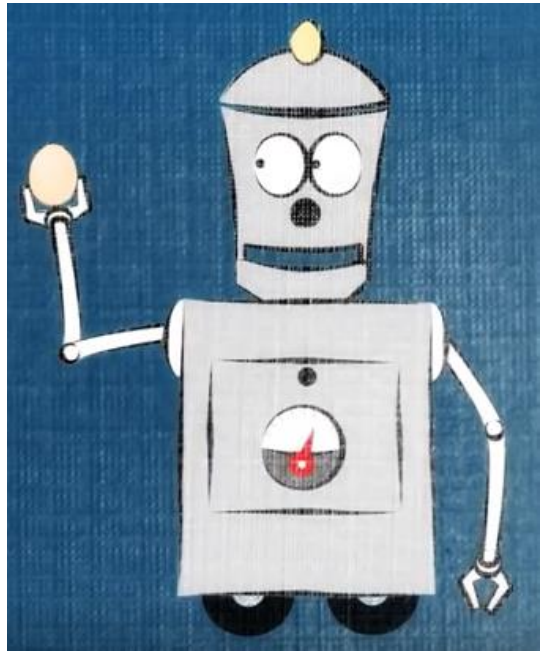
- **Egg-boiling Fuzzy Ruleset**
  - **IF** egg size is **small** **THEN** boil **less than 5 minutes**
  - **IF** egg size is **large** **THEN** boil **more than 5 minutes**
- **3 Steps of Fuzzy Reasoning**
  - Fuzzification
  - Inference
  - Defuzzificaiton

Fuzzy Logic

Fuzzy Rules : Fuzzification



Fuzzy Logic
Fuzzy Rules : Inference

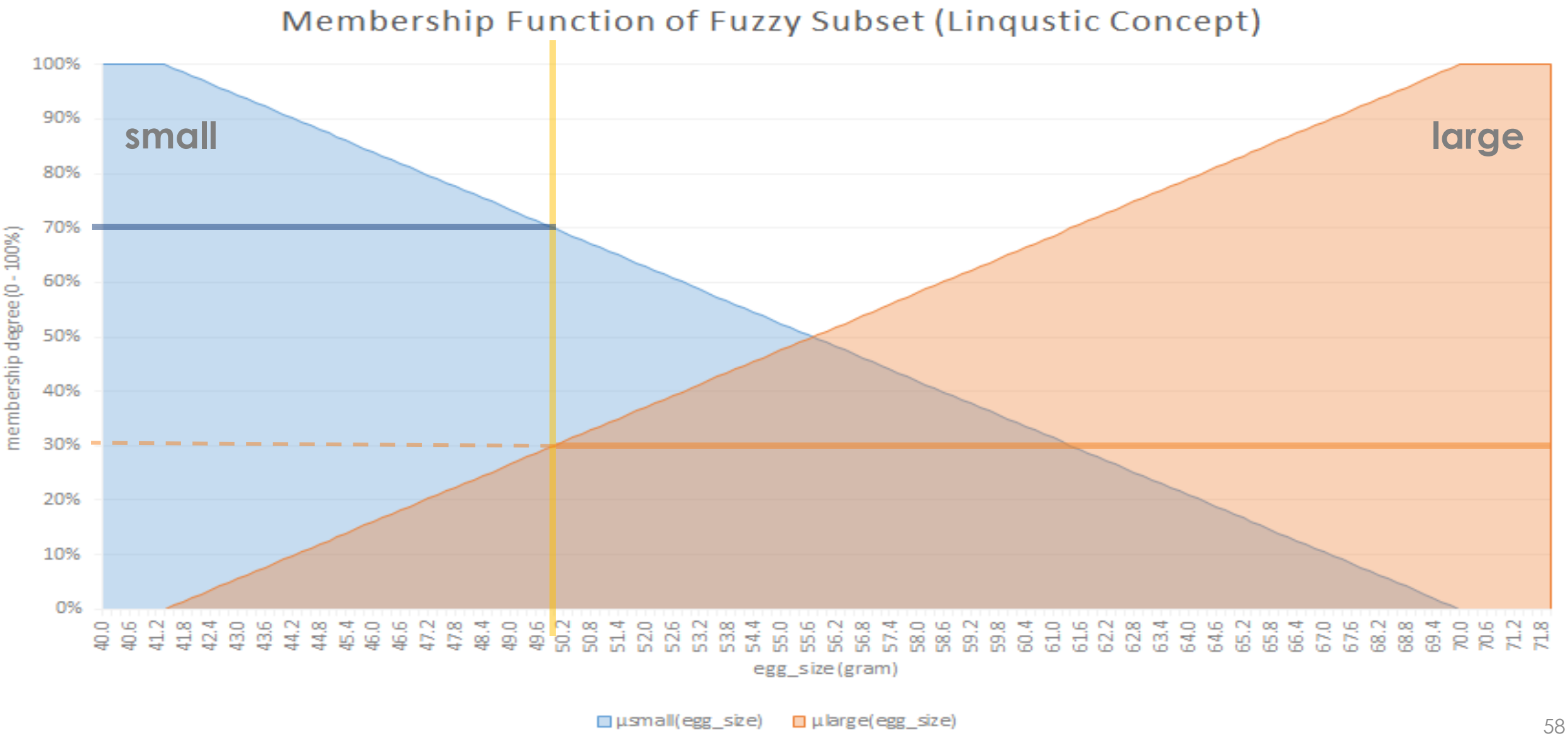


Link https://www.youtube.com/watch?v=J_Q5X0nTmrA

- **New fact: The egg is 50 grams**
- **Egg-boiling Fuzzy Ruleset**
  - **IF** egg size is **small (%) THEN** boil **less than 5 minutes (%)**
  - **IF** egg size is **large (%) THEN** boil **more than 5 minutes (%)**

Fuzzy Logic

Fuzzy Rules : Inference



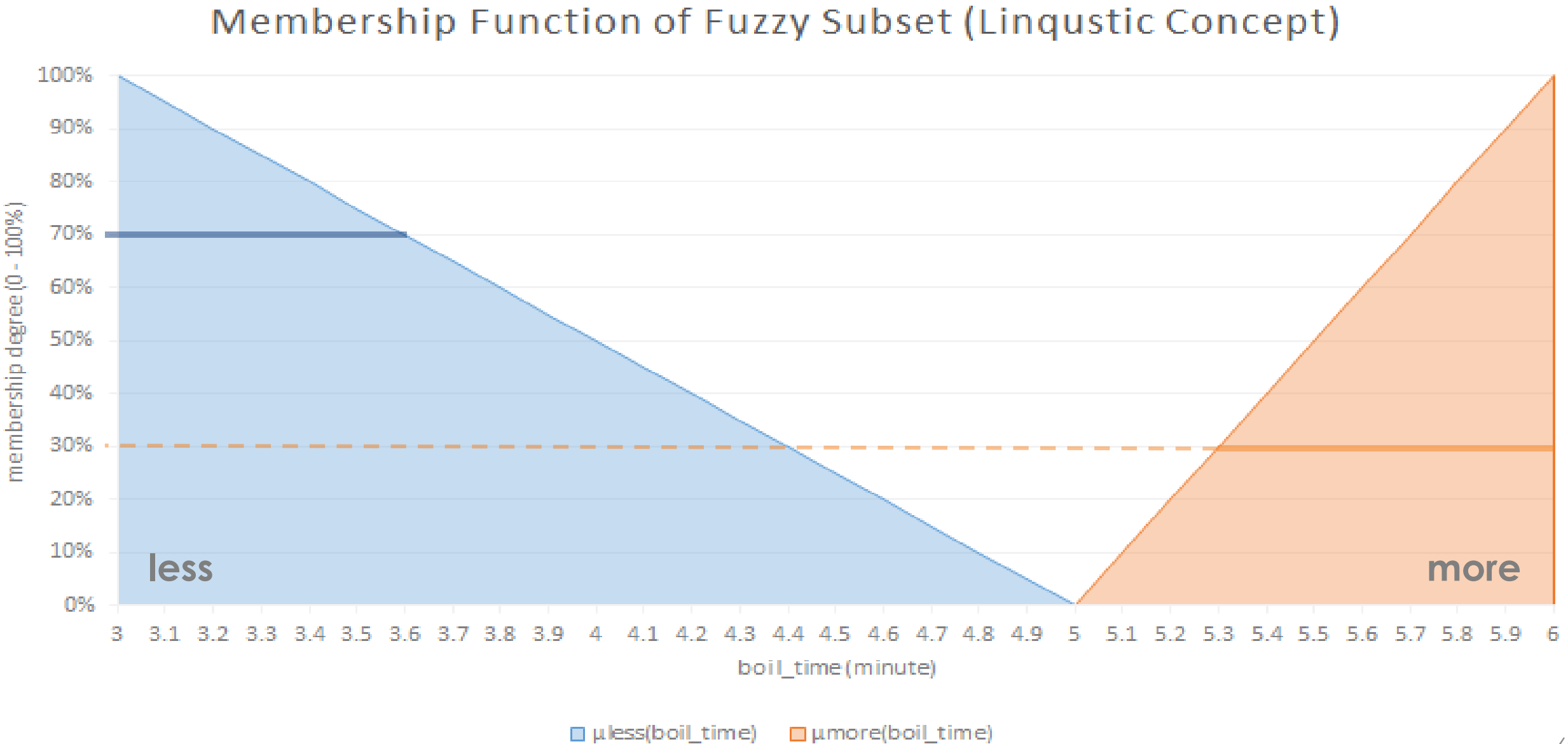
Fuzzy Logic

Fuzzy Rules : Inference

- **Egg-boiling Fuzzy Ruleset**
  - **IF** egg size is **small (70%)** **THEN** boil **less than 5 minutes (70%)**
  - **IF** egg size is **large (30%)** **THEN** boil **more than 5 minutes (30%)**

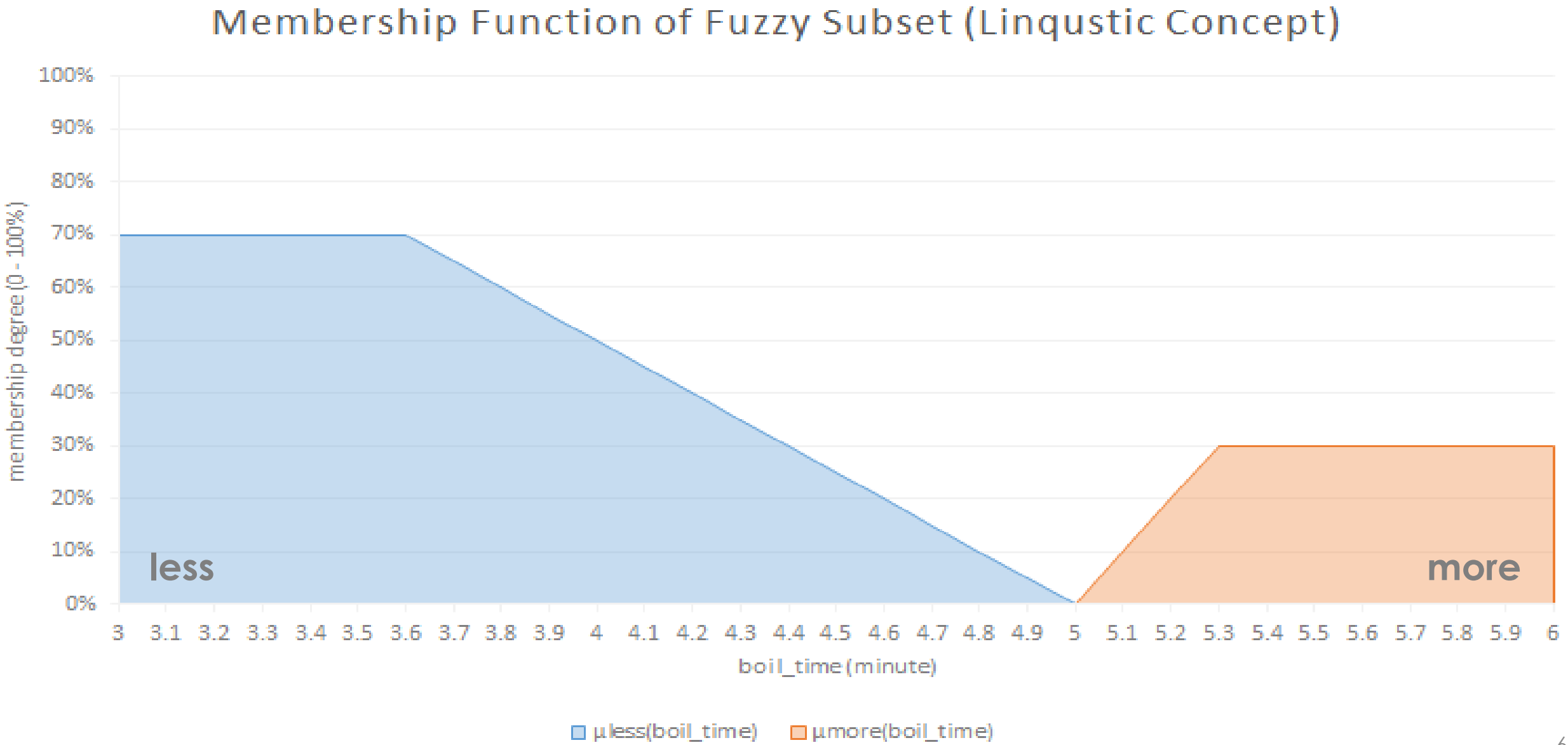
Fuzzy Logic

Fuzzy Rules : Inference



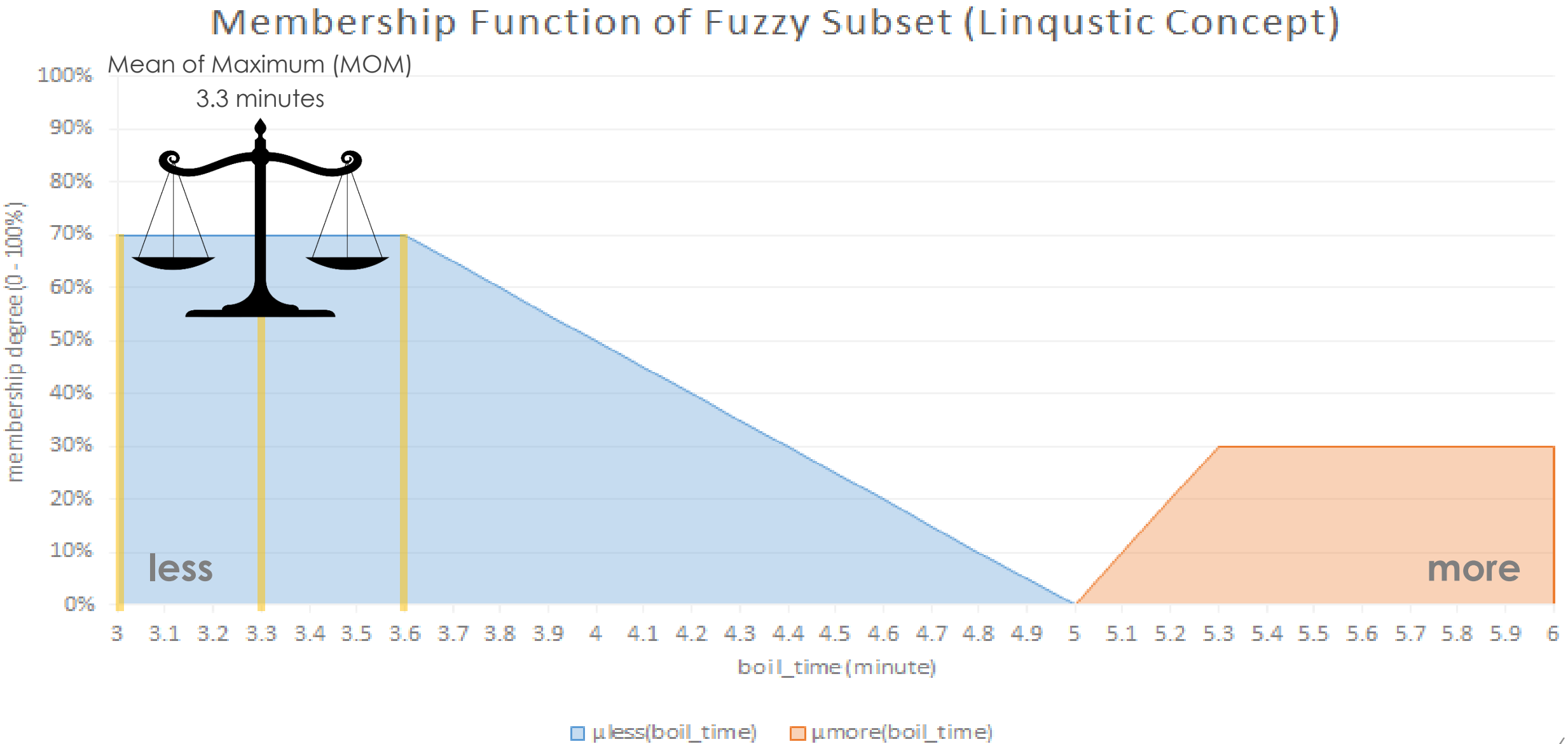
Fuzzy Logic

Fuzzy Rules : Inference



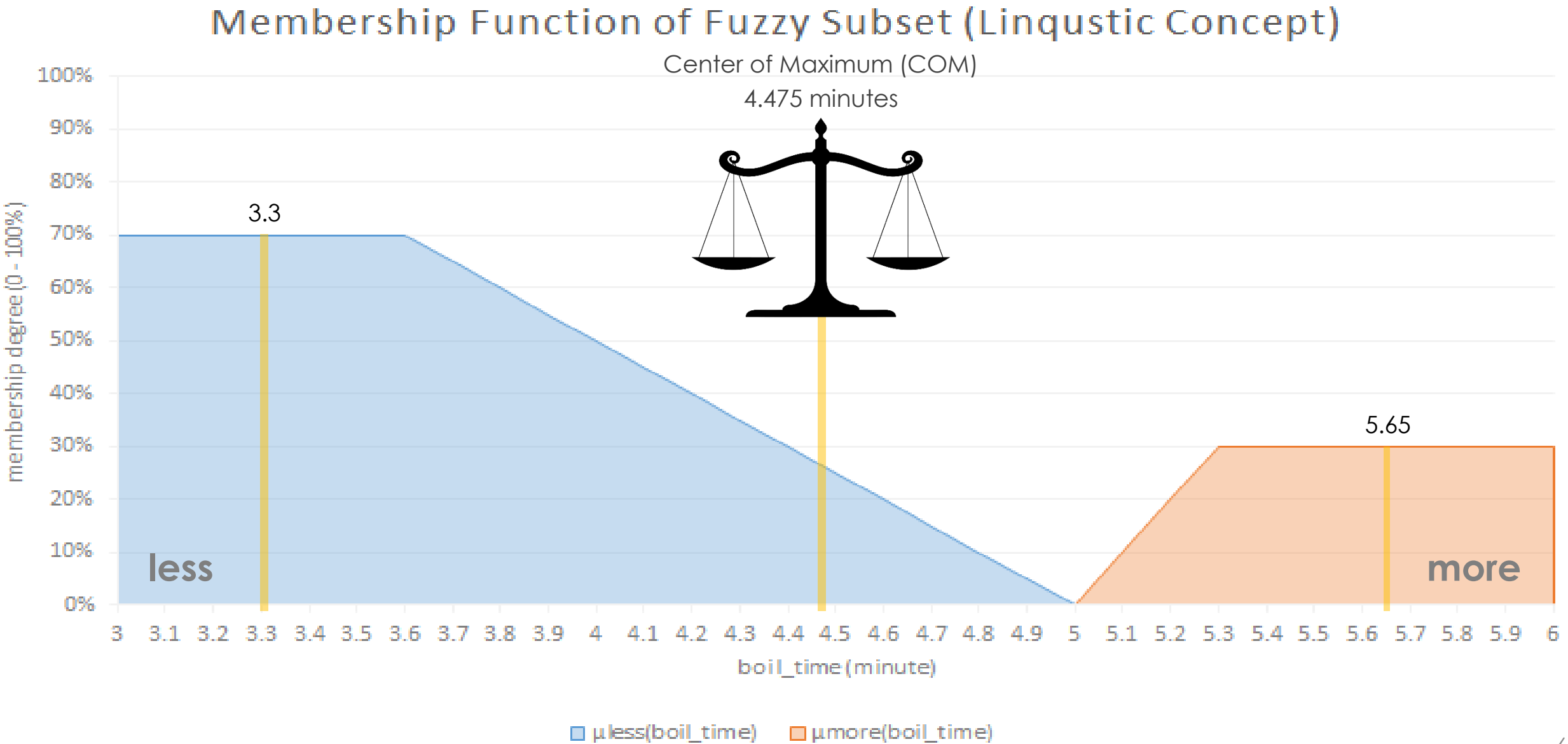
Fuzzy Logic

Fuzzy Rules : Defuzzification



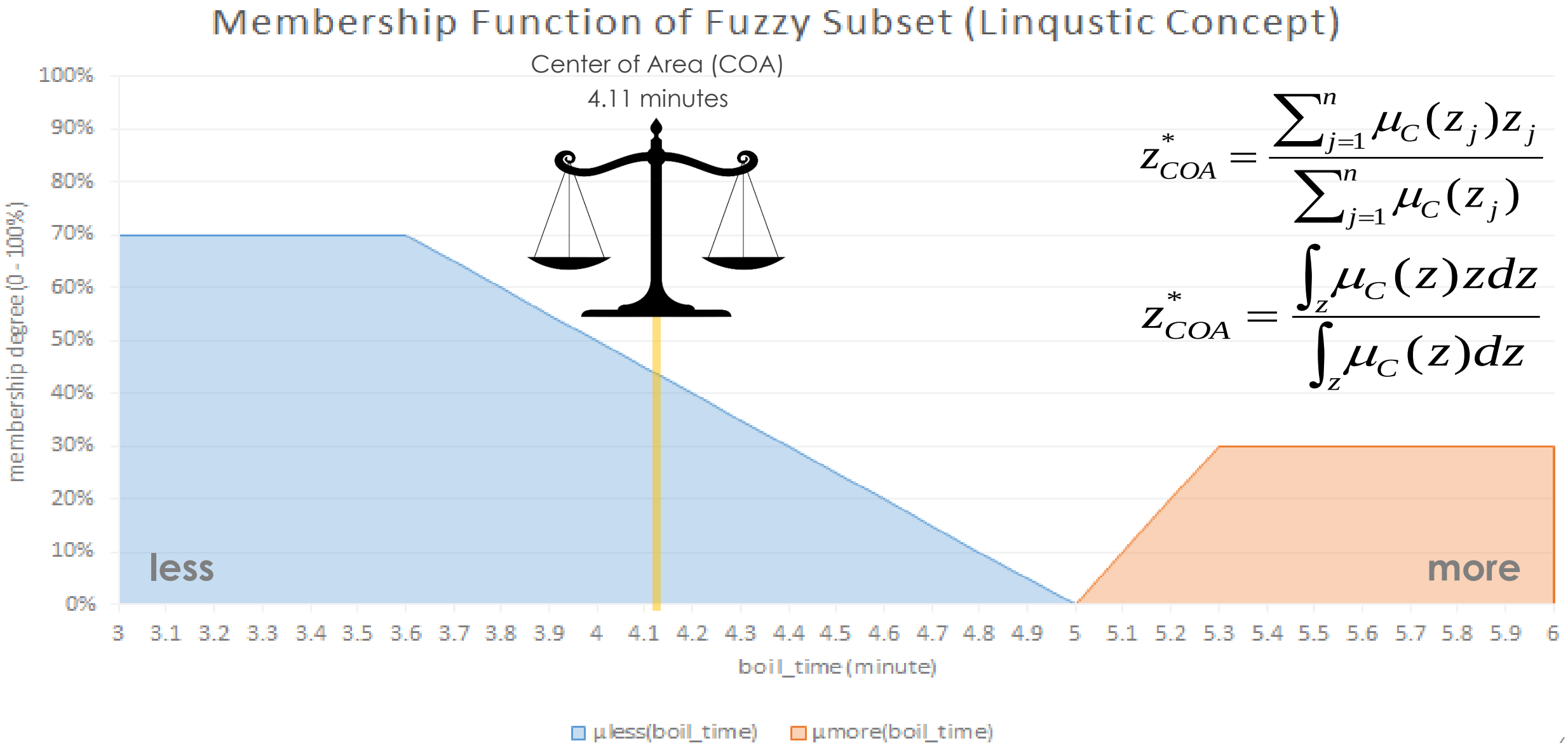
Fuzzy Logic

Fuzzy Rules : Defuzzification



Fuzzy Logic

Fuzzy Rules : Defuzzification



$$z^*_{COA} = \frac{\sum_{j=1}^{n} \mu_C(z_j) z_j}{\sum_{j=1}^{n} \mu_C(z_j)}$$

$$z^*_{COA} = \frac{\int_z \mu_C(z) z dz}{\int_z \mu_C(z) dz}$$

Fuzzy Logic

Fuzzy Rules : Extension

- **More than one conditions**
  - **IF** egg size is **small (70%)** **OR** **very** hungry **(90%)** **THEN** boil **less than 5 minutes (%)**
  - **IF** egg size is **large (45%)** **AND** **slightly** hungry **(75%)** **THEN** boil **more than 5 minutes (%)**
- **Composition operation**
  - **AND Min()**
  - **OR Max()**

Fuzzy Logic

Fuzzy Logic Applications

Automatic Washing Machine

- **Using fuzzy rules in the form of:**

  IF **few** clothes and they are soft THEN gentle flow and **short** washing time

  (where **few**, **soft** , … are based on measure (fuzzy values) from sensor, **gentle**, **short**, … are fuzzy concepts for control)

Fuzzy Vacuum Cleaner

- **Fuzzy control of absorbing power based on the material & the dirty degree of the floor**

  If the sucking power is too strong, the nozzle will stuck on floor (difficult to operate); if too weak, the corner dust cannot be absorbed well.

Fuzzy Logic

Fuzzy Logic Summary

The theory of fuzziness is to build models for entities which lack a rigorous definition.

- **The concept of "graded membership" belongs to a class which could be subjective in different business context.**

- **It is not compatible with a concept suitable for the lack of information, which is with probability.**

Fuzzy Logic

Fuzzy Logic vs. Probability

Fuzziness and randomness deal with different types of uncertainty in our life

- **Is it a raining day now?**
  - To describe some existing situation
  - It is more subjective (different people may have different ideas)
  - Uncertainty of classification
- **Is it going to rain tomorrow?**
  - The event may or may not happen
  - It is objective (determined by natural law)
  - Uncertainty of occurrence

https://us.123rf.com/450wm/spawn83/spawn831809/spawn83180900051/108908180-rain-outside-the-window-raindrops-on-the-windowpane-on-a-cloudy-day.jpg?ver=6

3.2 Reasoning under Uncertainty

3.2.1 Analogical Reasoning (Similarity)

3.2.2 Fuzzy Logic Reasoning

3.2.3 Abductive Reasoning (Probability)

3.2.4 Exercise

Abductive Reasoning (Probability)

Probabilistic Reasoning (Bayesian inference)

- Bayesian inference is a method of statistical inference in which Bayes' theorem is used to update the probability for a **hypothesis** as more **evidence** or information becomes available.

Common Forms of Reasoning

Abductive Reasoning

- Probabilistic calculation; Prior/Conditional/Joint probability; Bayesian network; (Hypothesis ~ Evidence)

Male or Female?

Probabilistic Reasoning

Bayesian Inference

P(man with long hair) = P(long hair) * P(man | long hair)

P(long hair and man) = P(man) * P(long hair | man)

Because P(man and long hair) = P(long hair and man)

P(long hair) * P(man | long hair) = P(man) * P(long hair | man)

P(man | long hair) = P(man) * P(long hair | man) / P(long hair)

P(A | B) = P(B | A) * P(A) / P(B)

Bayesian inference

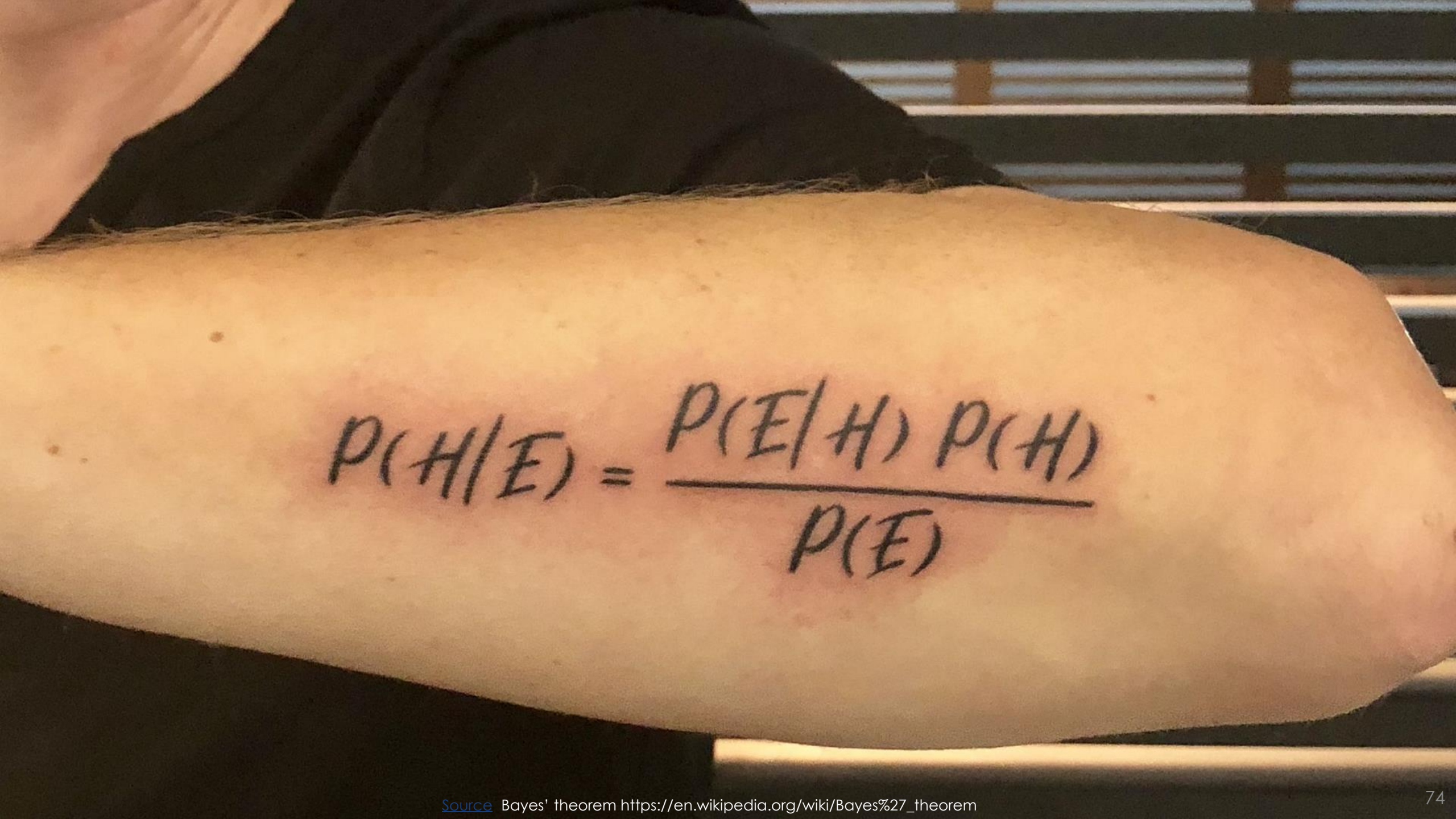
https://en.wikipedia.org/wiki/Bayesian_inference

How Bayesian inference works

https://brohrer.github.io/how_bayesian_inference_works.html

P(man | long hair) = [P(man) * P(long hair | man)] / [P(woman with_and long hair) + P(man with_and long hair)]

= (0.5 * 0.04) / (0.25 + 0.02) = 0.02 / 0.27 = 0.07 (7%)



Source Bayes' theorem https://en.wikipedia.org/wiki/Bayes%27_theorem

3.2 Reasoning under Uncertainty

3.2.1 Analogical Reasoning (Similarity)

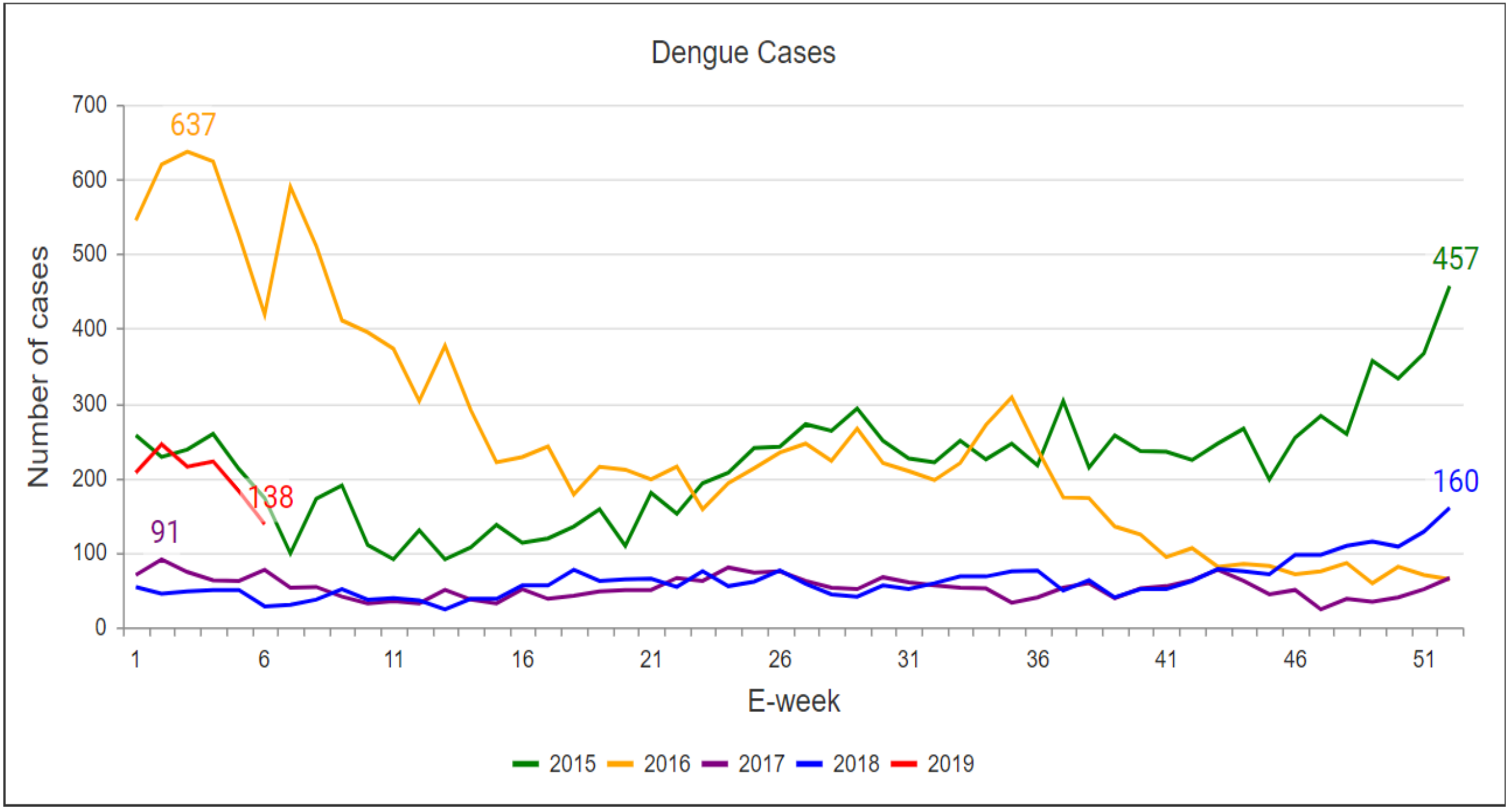
3.2.2 Fuzzy Logic Reasoning

3.2.3 Abductive Reasoning (Probability)

3.2.4 Exercise

Danger? Dengue!

February 2019
Singapore National Environment Agency NEA



Reasoning under Uncertainty

[Exercise] Sam's test report retuned as positive!

Table 7 **Overall accuracy of physician's dengue diagnosis with NS1 rapid test result.**

Physician's diagnosis (Medical test result: positive/negative?)	Confirmed diagnosis (Virus in blood?)		
	Dengue (virus in blood)	Non-dengue (no-virus in blood)	Total
Dengue (Medical test: positive)	137	44	181
Non-dengue (Medical test: negative)	46	170	216
Total	183	214	397

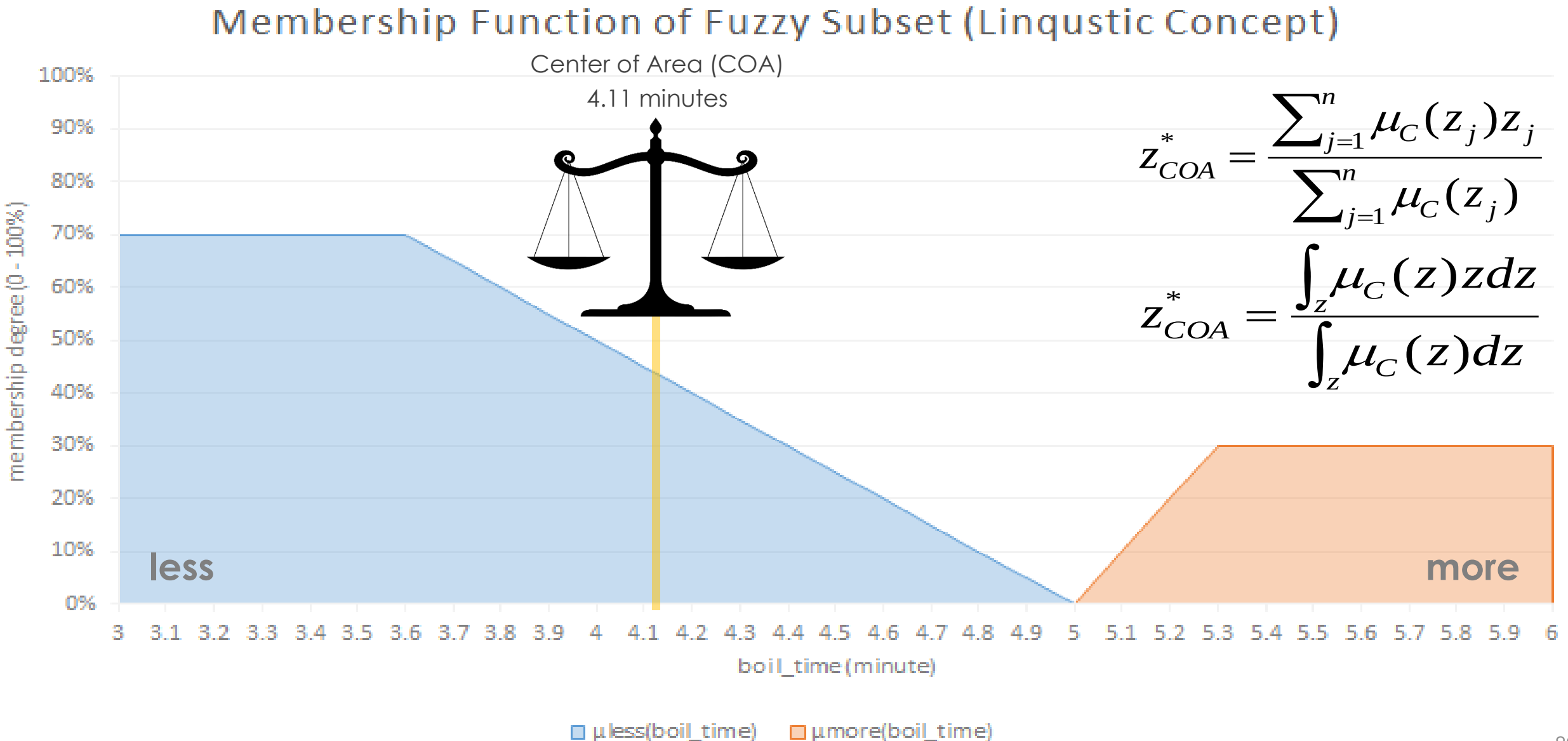
Sensitivity 75%; Specificity 79%

https://doi.org/10.1371/journal.pntd.0006573.t007

Based on above known past knowledge/data; quantify the chance (calculate probability) that there is real dengue virus in Sam's blood.

Reasoning under Uncertainty

[Exercise] Fuzzy Logic : COA defuzzification calculation



$$z^*_{COA} = \frac{\sum_{j=1}^{n} \mu_C(z_j) z_j}{\sum_{j=1}^{n} \mu_C(z_j)}$$

$$z^*_{COA} = \frac{\int_z \mu_C(z) z dz}{\int_z \mu_C(z) dz}$$

Reasoning under Uncertainty

[Exercise] Fuzzy Logic : COA defuzzification calculation

boil_time (minute)	$\mu_{less(boil\_time)}$	$\mu_{more(boil\_time)}$	$\mu_{(boil\_time)} * boil\_time$ (minute)	$\sum_{j=1}^{n} \mu_C(z_j) z_j$	$\sum_{j=1}^{n} \mu_C(z_j)$	Z_{COA}
3	0.7	0				
3.1	0.7	0				
3.2	0.7	0				
3.3	0.7	0				
3.4	0.7	0				
3.5	0.7	0				
3.6	0.7	0				
3.7	0.65	0				
3.8	0.6	0				
3.9	0.55	0				
4	0.5	0				
4.1	0.45	0				
4.2	0.4	0				
4.3	0.35	0				
4.4	0.3	0				
4.5	0.25	0				
4.6	0.2	0				
4.7	0.15	0				
4.8	0.1	0				
4.9	0.05	0				
5	0	0				
5.1	0	0.1				
5.2	0	0.2				
5.3	0	0.3				
5.4	0	0.3				
5.5	0	0.3				
5.6	0	0.3				
5.7	0	0.3				
5.8	0	0.3				
5.9	0	0.3				
6	0	0.3				

3.3 Deductive Reasoning (under Uncertainty) [Workshop]

3.3.1 Analogical Reasoning (Similarity)

Special thanks to Yan Wei Quan (A0215498U) for his contribution.

3.3.2 Fuzzy Logic Reasoning

Special thanks to Yan Wei Quan (A0215498U) for his contribution.

3.3.3 Abductive Reasoning (Probability)

Special thanks to Yan Wei Quan (A0215498U) for his contribution.

3.3.4 Workshop Submission

3.3 Deductive Reasoning (under Uncertainty) [Workshop]

3.3.1 Analogical Reasoning (Similarity)

Special thanks to Yan Wei Quan (A0215498U) for his contribution.

3.3.2 Fuzzy Logic Reasoning

Special thanks to Yan Wei Quan (A0215498U) for his contribution.

3.3.3 Abductive Reasoning (Probability)

Special thanks to Yan Wei Quan (A0215498U) for his contribution.

3.3.4 Workshop Submission

Analogical Reasoning (Similarity)

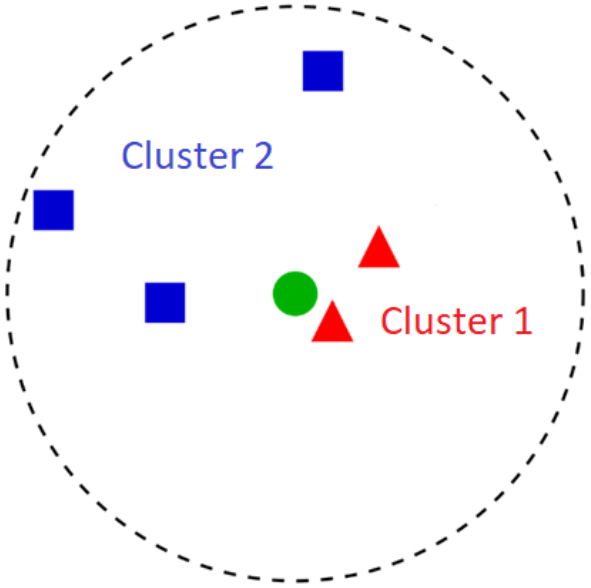
- **Learning Objectives**
  - Understand how the kNN: k nearest neighbors algorithm works
  - Learn how to apply kNN algorithm into recommendation system
  - Modify the parameters to see the difference among the models
- **Package to use,**
  - orange-canvas
  - Scikit-learn 0.23.1 in Python
- **Technique,**
  - Similarity-based Machine Reasoning (kNN algorithm)
- **Case/Scenario,**
  - Autistic Spectrum Disorder classification and result visualization
- **Requirement,**
  - Familiar with Python

What is k nearest neighbors algorithm

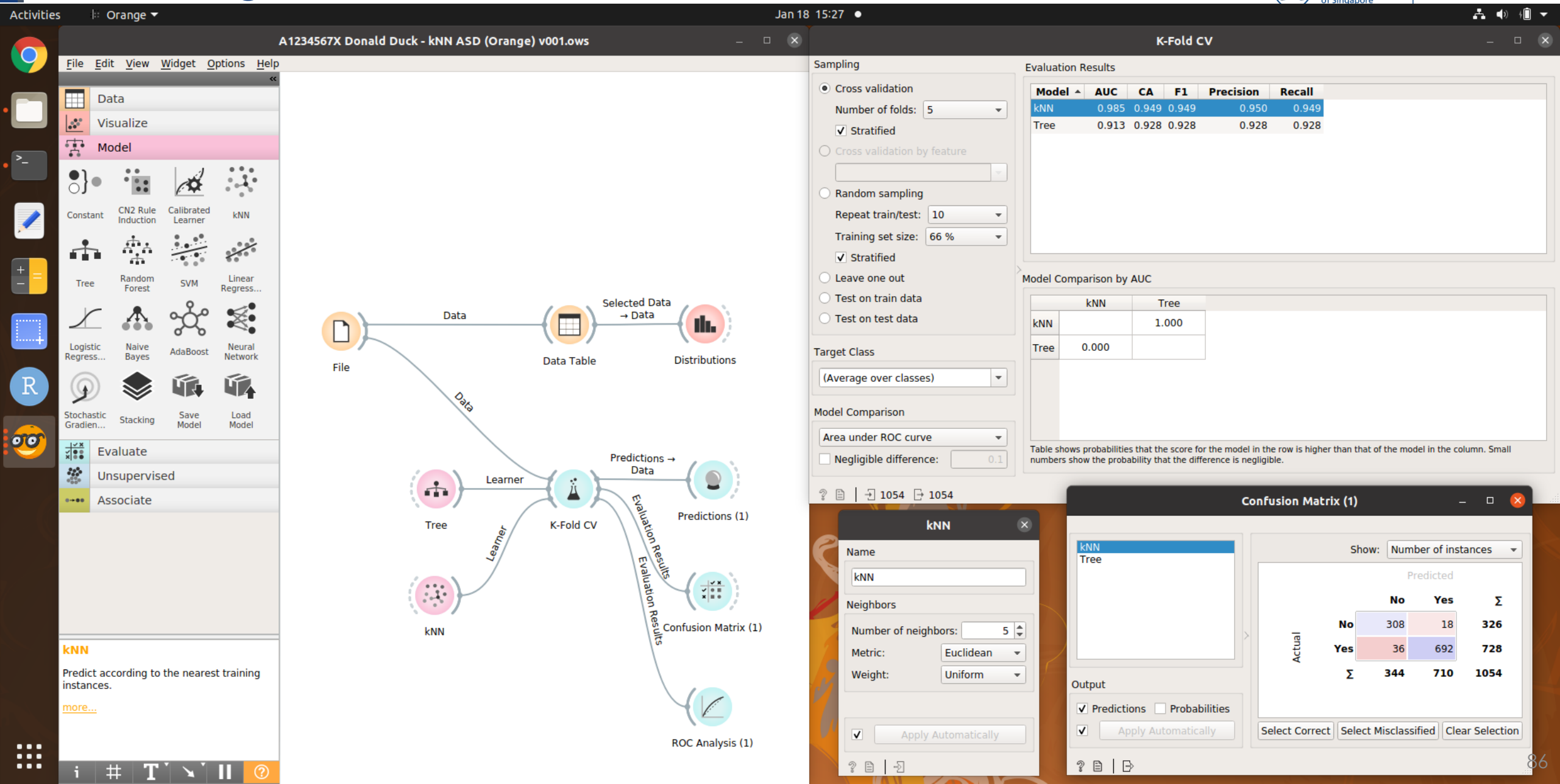
- **In kNN algorithm, which cluster a new case (don't know the label) belongs to is decided by it's k nearest neighbors.**
- **WHEN:**

1. **k is five;**
2. **A new case: F, its five nearest neighbors are A, B, C, D, E;**
3. **A and B belong to cluster 1;**
4. **C, D and E belong to cluster 2;**

- **THEN:**
- **F is classified to cluster 2, via majority voting: 3 vs 2.**



Orange's kNN


Activities
Orange
Jan 18 15:27
A1234567X Donald Duck - kNN ASD (Orange) v001.ows
File Edit View Widget Options Help
Data
Visualize
Model
Constant
CN2 Rule Induction
Calibrated Learner
kNN
Tree
Random Forest
SVM
Linear Regress...
Logistic Regress...
Naive Bayes
AdaBoost
Neural Network
Stochastic Gradien...
Stacking
Save Model
Load Model
Evaluate
Unsupervised
Associate
kNN
Predict according to the nearest training instances.
more...
File
Data
Data Table
Selected Data → Data
Distributions
Data
Tree
Learner
K-Fold CV
Predictions → Data
Predictions (1)
Learner
kNN
Evaluation Results
Confusion Matrix (1)
Evaluation Results
ROC Analysis (1)
K-Fold CV
Sampling
Cross validation
Number of folds: 5
Stratified
Cross validation by feature
Random sampling
Repeat train/test: 10
Training set size: 66 %
Stratified
Leave one out
Test on train data
Test on test data
Target Class
(Average over classes)
Model Comparison
Area under ROC curve
Negligible difference: 0.1
Evaluation Results
Model AUC CA F1 Precision Recall
kNN 0.985 0.949 0.949 0.950 0.949
Tree 0.913 0.928 0.928 0.928 0.928
Model Comparison by AUC
kNN Tree
kNN 1.000
Tree 0.000
Table shows probabilities that the score for the model in the row is higher than that of the model in the column. Small numbers show the probability that the difference is negligible.
1054 1054
kNN
Name
kNN
Neighbors
Number of neighbors: 5
Metric: Euclidean
Weight: Uniform
Apply Automatically
Confusion Matrix (1)
kNN
Tree
Output
Predictions Probabilities
Apply Automatically
Show: Number of instances
Predicted
No Yes Σ
Actual
No 308 18 326
Yes 36 692 728
Σ 344 710 1054
Select Correct Select Misclassified Clear Selection

Scikit's kNN

- **Installation & Tutorial:**

1. **Type in pip install scikit-learn in your anaconda environment**
2. **Documentations of kNN in https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier**

Workshop

- Open the A1234567X Donald Duck – kNN python notebook file and complete the workshop.
- Run the workshop and see the result. Can you find something wrong in this workshop? (Hint, before training the kNN, what should we do?)

Workshop

- **Reflection: kNN is very sensitive to the distance between two samples. Therefore, we have to carefully preprocess our training samples. In this workshop. Scaling part is missing.**
- **Besides, for multi-values categorical variable, we should use one-hot vector instead which is also missing.**
- **You should solve the problem by pre-processing data frame before fitting a model to improve the kNN classification accuracy.**

Workshop Submission

- **Naming convention: StudentID YourFullName Workshop Name, e.g. A1234567X Donald Duck – sln – kNN.ipynb/zip**
- **Use zip to a single file, then rename, if you plan to submit multiple files.**

References

1. User guide of scikit learn: https://scikit-learn.org/stable/user_guide.html

2. kNN model in Wikipedia: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

3.3 Deductive Reasoning (under Uncertainty) [Workshop]

3.3.1 Analogical Reasoning (Similarity)

Special thanks to Yan Wei Quan (A0215498U) for his contribution.

3.3.2 Fuzzy Logic Reasoning

Special thanks to Yan Wei Quan (A0215498U) for his contribution.

3.3.3 Abductive Reasoning (Probability)

Special thanks to Yan Wei Quan (A0215498U) for his contribution.

3.3.4 Workshop Submission

Fuzzy Logic Reasoning

- **Learning Objectives**
  - **Learn the complete procedure of fuzzy machine reasoning**
- **Package to use,**
  - **scikit-fuzzy 0.4.2**
- **Technique,**
  - **Fuzzy logic**
- **Case/Scenario,**
  - **Fuzzy washing machine**
- **Requirement,**
  - **Familiar with python**

Before the workshop

- **Installation**

1. **Type in pip install scikit-fuzzy in your anaconda environment**

Intelligent Fuzzy Washing Machine

- **This machine has two sensors (linguistic/fuzzy variable)**
  - **Height sensor**
  - **Weight sensor**
- **According to the data given by the sensors, we set three categories (fuzzy subsets) for each sensor.**
  - **Height: Low, Medium, High (Range 0 – 100 cm)**
  - **Weight: Light, Medium, Heavy (Range 0 – 10 kg)**
- **Goal is to obtain value of washing time fuzzy variable**
  - **Washing time: Short, Medium, Long (Range 15 – 50 minutes)**
- **Cloth loads (sensor data) are randomly generated here to simulate the real washing situation.**

Fuzzy rules

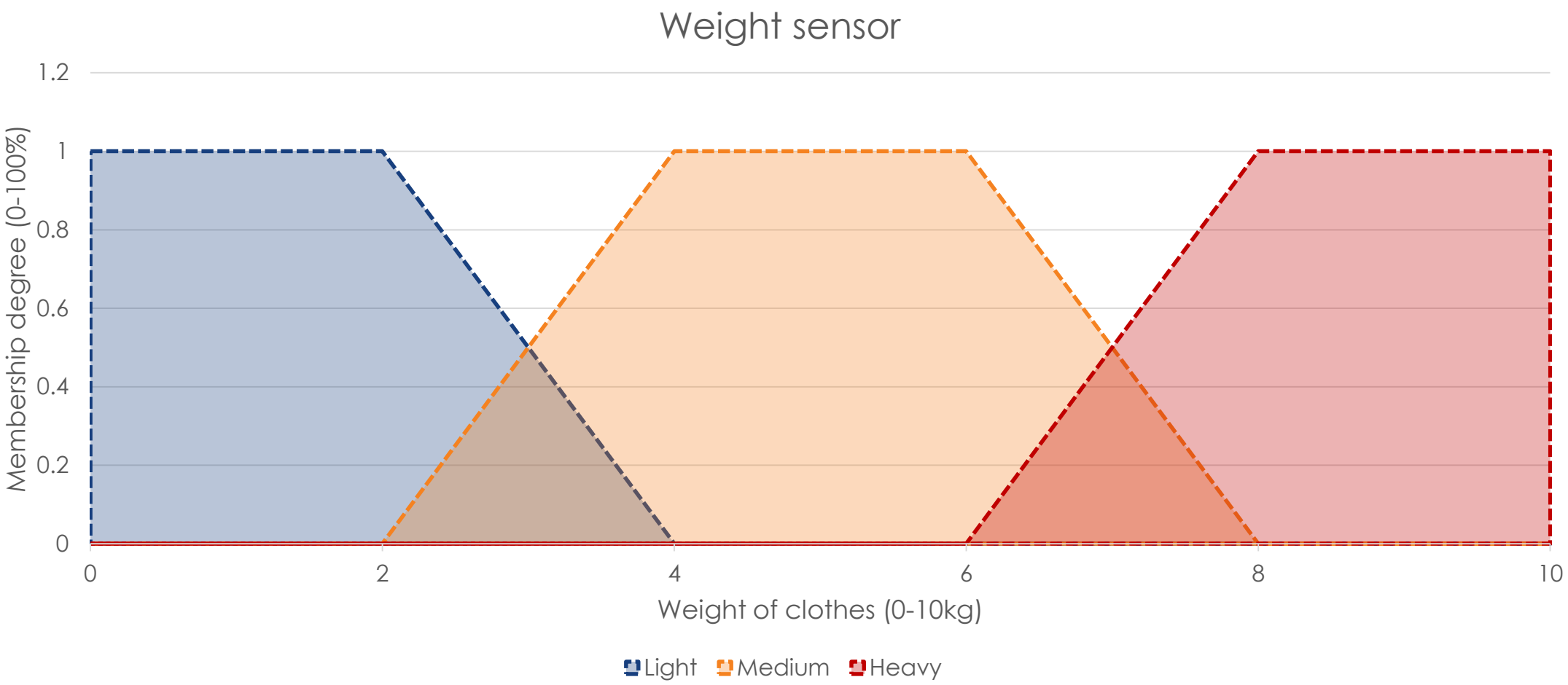
Height \ Weight	Light	Medium	Heavy
Low	Short R1	Short R2	Medium R4
Medium	Short R3	Medium R5	Long R7
High	Medium R6	Long R8	Long R9

Fuzzy rule sets: WHEN Height = ? AND Weight = ? THEN Time = ?

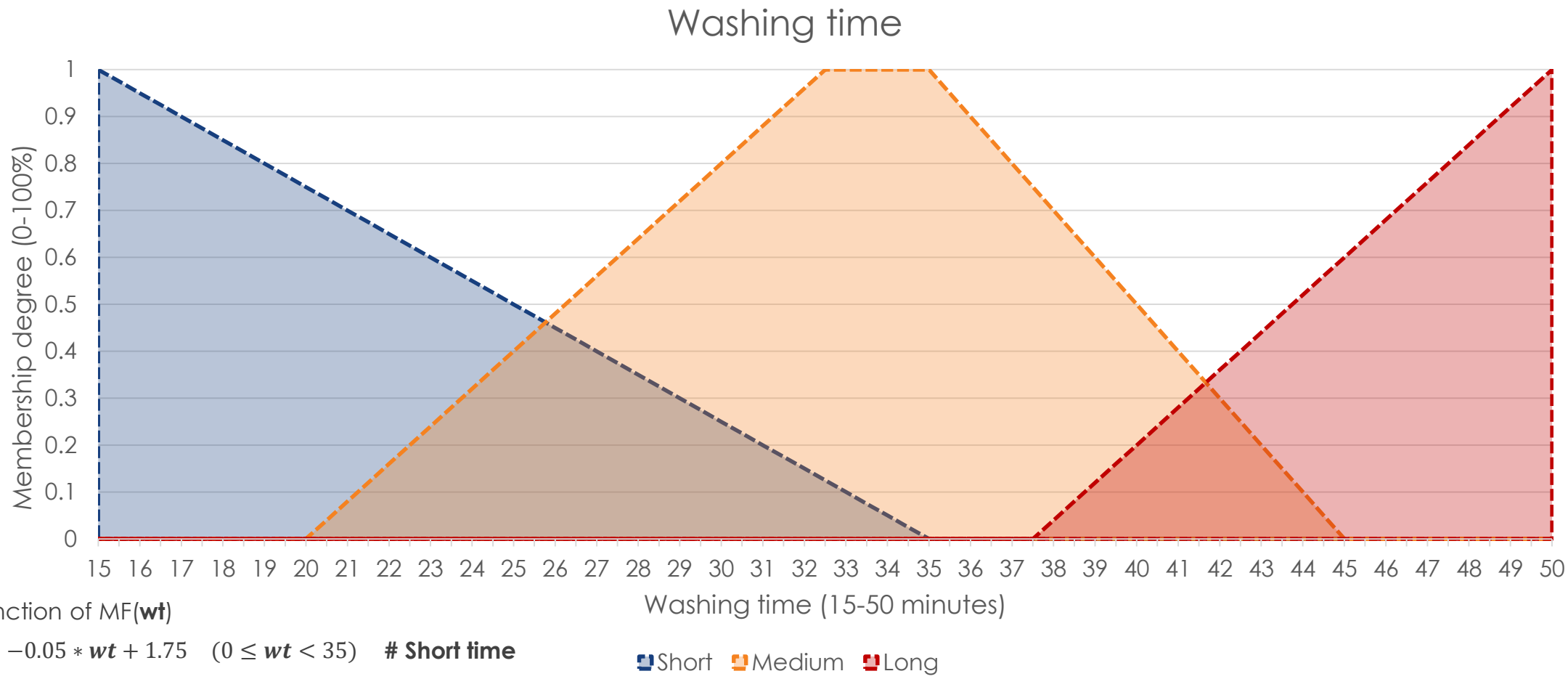
For example, for the orange highlighted cell, when weight is light & height is high, then we should wash for medium time.

Membership function

Membership function



Membership function



Function of MF(**wt**)

$\boldsymbol{\mu} = -0.05 * \boldsymbol{wt} + 1.75 \quad (0 \leq \boldsymbol{wt} < 35)$ **# Short time**

$$\boldsymbol{\mu} = \begin{cases} 0.08 * \boldsymbol{wt} - 1.6 & (20 \leq \boldsymbol{wt} < 32.5) \\ 1 & (32.5 \leq \boldsymbol{wt} < 35) \\ -0.1 * \boldsymbol{wt} + 4.5 & (35 \leq \boldsymbol{wt} < 45) \end{cases}$$ **# Medium time**

$\boldsymbol{\mu} = 0.08 * \boldsymbol{wt} - 3 \quad (37.5 \leq \boldsymbol{wt} < 50)$ **# Long time**

μ value range (0, 100%)

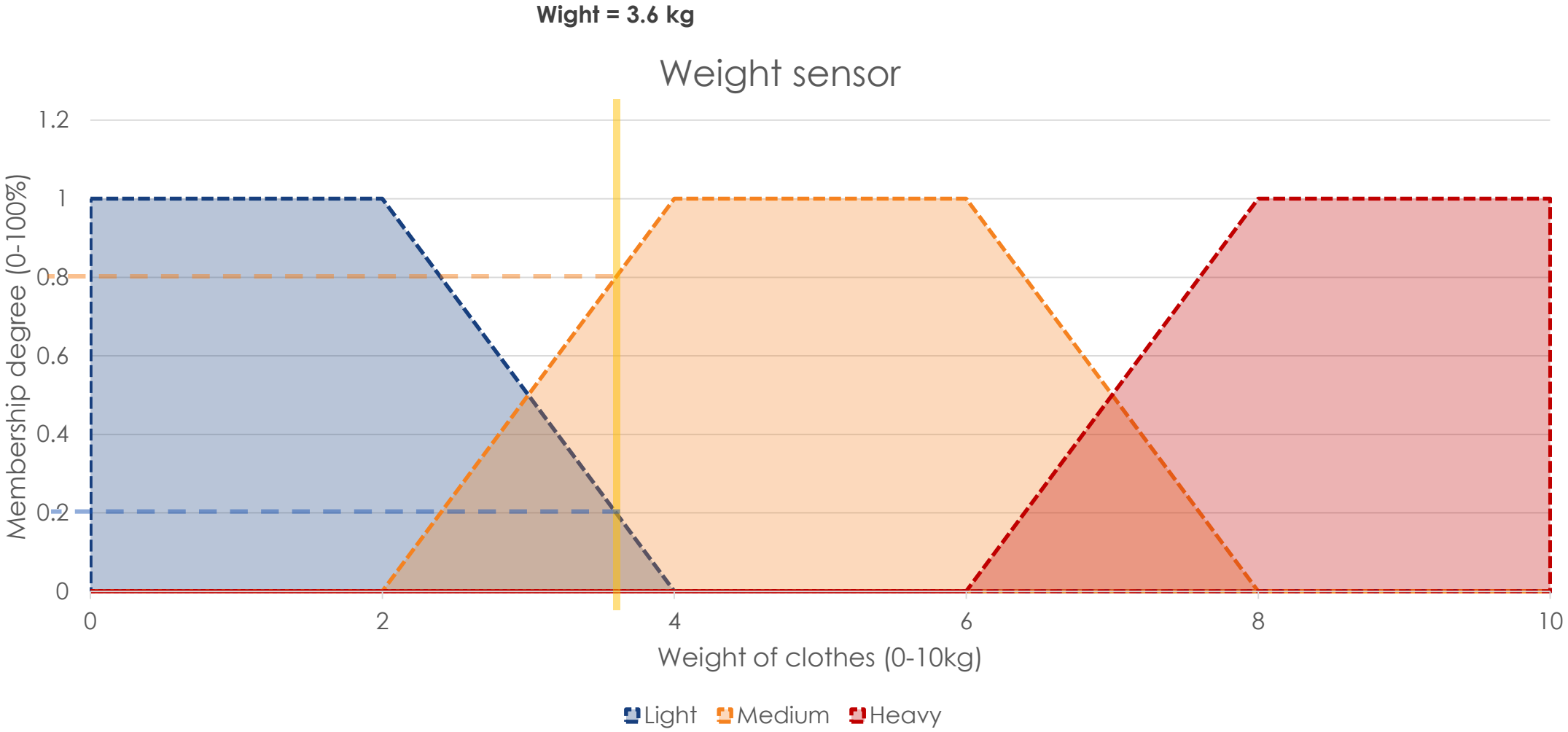
Customer initiates a new washing need/load

Each time the wash machine initializes:

- **Measure weight height of clothes**
  - E.g. ***Height*** = ***65*** cm ***Weight*** = ***3.6*** kg

Fuzzification (visualization)

Fuzzification (visualization)



Fuzzification (calculation)

Each time the wash machine initializes:

- **Measure weight height of clothes**
  - E.g. Height = 65 cm Weight = 3.6 kg
- **Fuzzification: Calculate the membership degree (fuzzy values) of weight and height (%).**
  - E.g. MF_{Low} (Height=65) = 0; MF_{Medium}(Height=65) = 0.75; MF_{High}(Height=65) = 0.25
    The MFs of height are [0, 0.75, 0.25] (%) [low, medium, high]
  - E.g. MF_{Light}(Weight=3.6) = 0.2; MF_{Medium}(Weight=3.6) = 0.8; MF_{Heavy}(Weight=3.6) = 0
    The MFs of weight are [0.2, 0.8, 0] (%) [light, medium, heavy]

Weight / Height	Light	Medium	Heavy
Low	Short R1	Short R2	Medium R4
Medium	Short R3	Medium R5	Long R7
High	Medium R6	Long R8	Long R9

Inference (calculation)

Each time the wash machine initializes:

- **Measure weight height of clothes**
  - E.g. Height = 65 cm Weight = 3.6 kg
- **Fuzzification: Calculate the membership degree (fuzzy values) of weight and height (%).**
  - E.g. MF_{Low} (Height=65) = 0; MF_{Medium}(Height=65) = 0.75; MF_{High}(Height=65) = 0.25
    The MFs of height are [0, 0.75, 0.25] (%) [low, medium, high]
  - E.g. MF_{Light}(Weight=3.6) = 0.2; MF_{Medium}(Weight=3.6) = 0.8; MF_{Heavy}(Weight=3.6) = 0
    The MFs of weight are [0.2, 0.8, 0] (%) [light, medium, heavy]

Height \ Weight	Light	Medium	Heavy
Low	Short R1	Short R2	Medium R4
Medium	Short R3	Medium R5	Long R7
High	Medium R6	Long R8	Long R9

- **Inference: Use min-max composition/strategy to determine the MF degree (fuzzy values) for three classes (Short/Medium/Long) of washing time. E.g. Fire fuzzy rules (total 9 rules)**

& → min(MF(Height), MF(Weight))

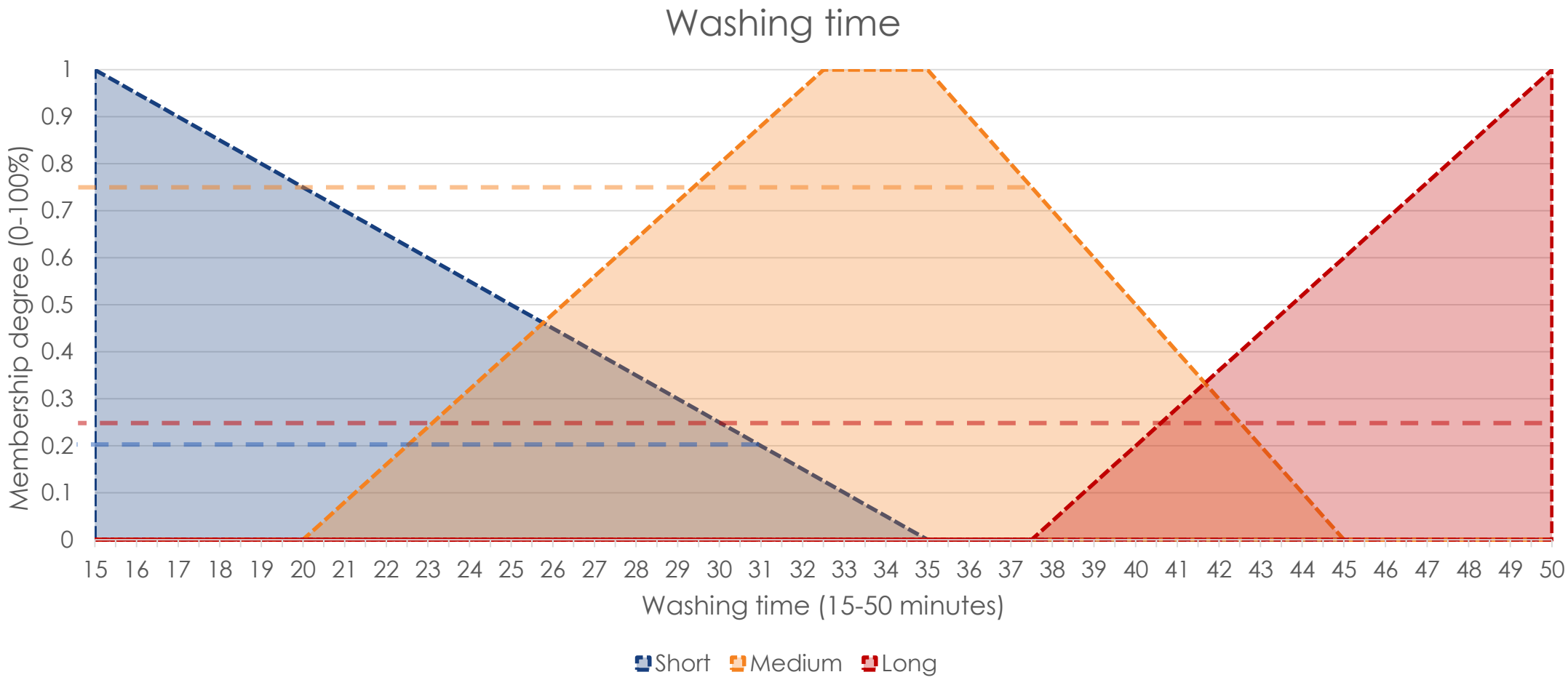
1. WHEN Height = Low (0) & Weight = Light (0.2) THEN Time = Short (0)
2. WHEN Height = Low (0) & Weight = Medium (0.8) THEN Time = Short (0)
3. WHEN Height = Medium (0.75) & Weight = Light (0.2) THEN Time = Short (0.2)
4. WHEN Height = Low (0) & Weight = Heavy (0) THEN Time = Medium (0)
5. WHEN Height = Medium (0.75) & Weight = Medium (0.8) THEN Time = Medium (0.75)
6. WHEN Height = High (0.25) & Weight = Light (0.2) THEN Time = Medium (0.2)
7. WHEN Height = Medium (0.75) & Weight = Heavy (0) THEN Time = Long (0)
8. WHEN Height = High (0.25) & Weight = Medium (0.8) THEN Time = Long (0.25)
9. WHEN Height = High (0.25) & Weight = Heavy (0) THEN Time = Long (0)

MF_{Short}(Time) = max() = 0.2

MF_{Medium}(Time) = max() = 0.75

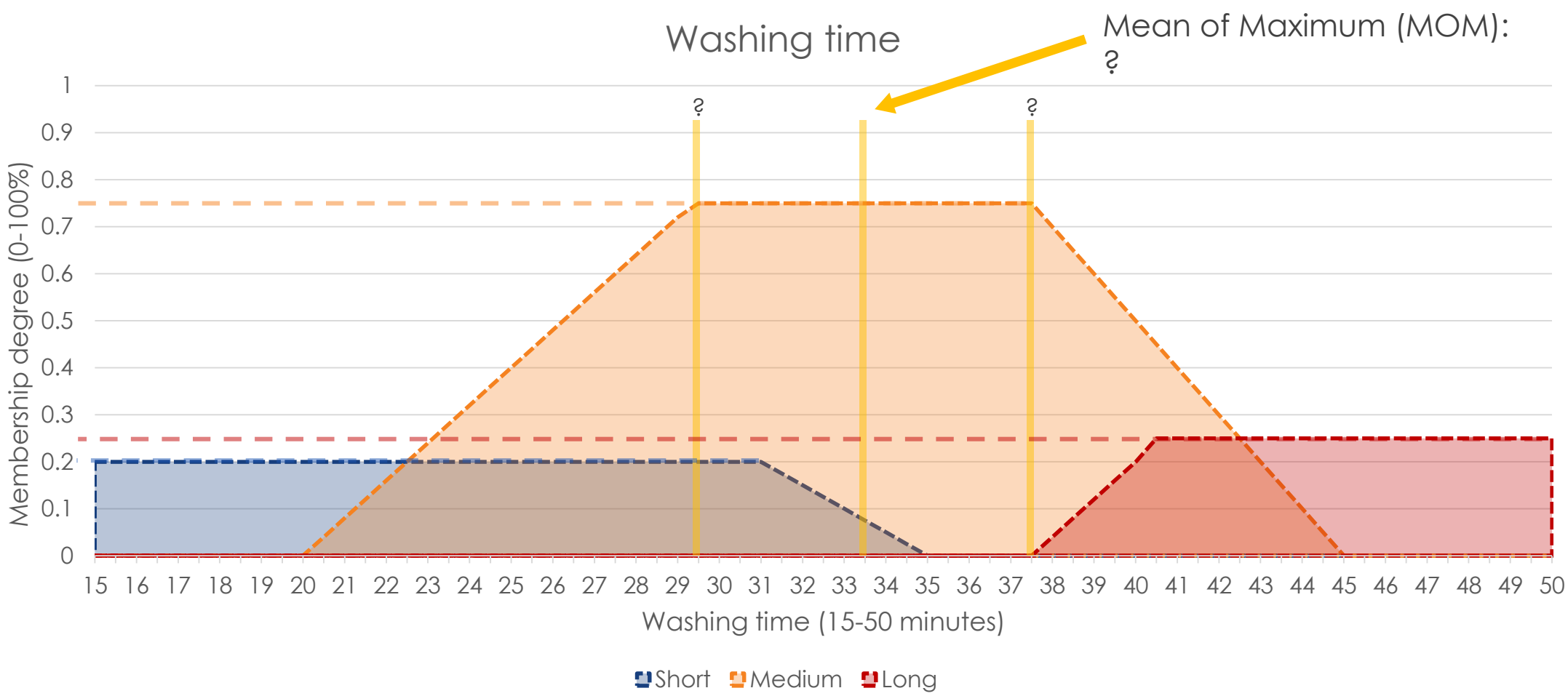
MF_{Long}(Time) = max() = 0.25

Inference (visualization)



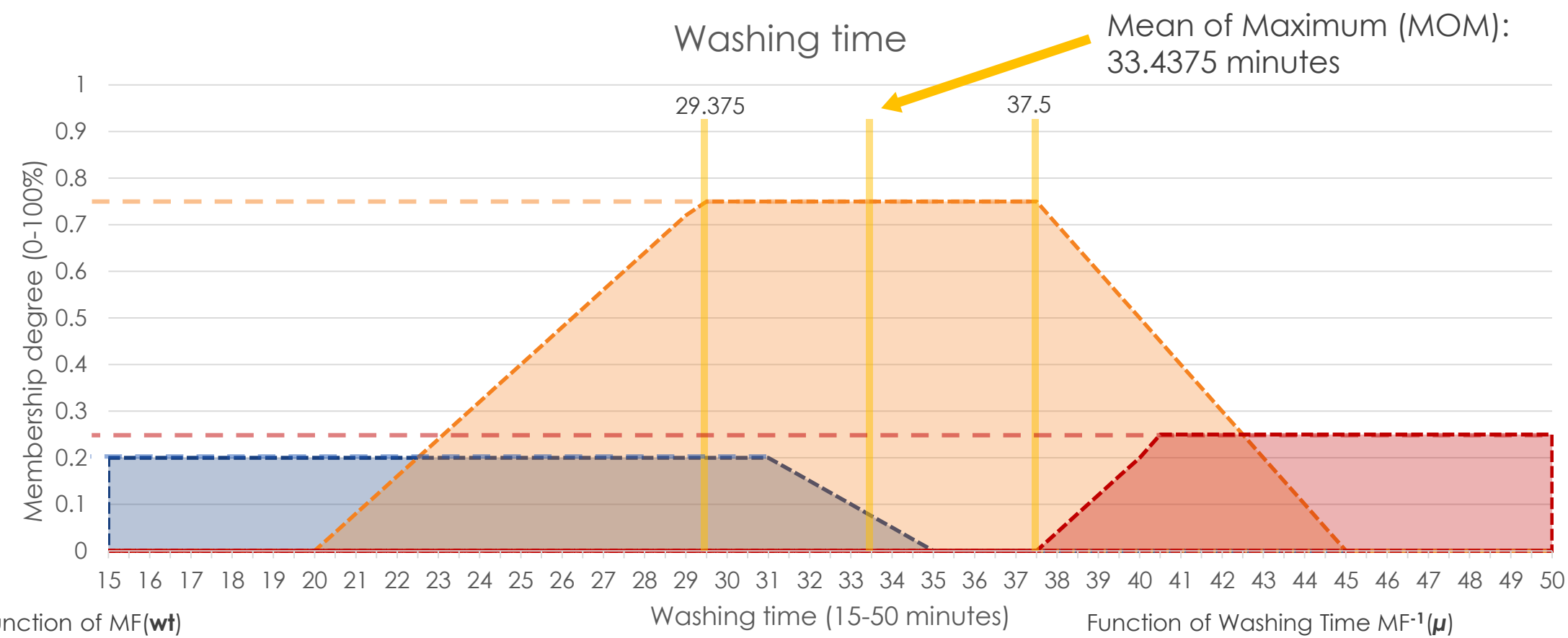
$MF_{Short}(Time) = 0.2$	$MF_{Medium}(Time) = 0.75$	$MF_{Long}(Time) = 0.25$

Defuzzification (visualization)



$MF_{Short}(Time) = 0.2$	$MF_{Medium}(Time) = 0.75$	$MF_{Long}(Time) = 0.25$

Defuzzification (calculation)



Function of MF(**wt**)

$\mu = -0.05 * wt + 1.75 \quad (0 \le wt < 35)$ **# Short time**

$$\mu = \begin{cases} 0.08 * wt - 1.6 & (20 \le wt < 32.5) \\ 1 & (32.5 \le wt < 35) \\ -0.1 * wt + 4.5 & (35 \le wt < 45) \end{cases}$$ **# Medium time**

$\mu = 0.08 * wt - 3 \quad (37.5 \le wt < 50)$ **# Long time**

μ value range (0, 100%)

$MF_{Medium}(Time) = 0.75$

Function of Washing Time $MF^{-1}(\mu)$
 # Medium time

$wt_{left\ slop} = (1.6 + \mu)/0.08$ =(1.6+0.75)/0.08 = **29.375**

$wt_{right\ slop} = (4.5 - \mu)/0.1$ =(4.5-0.75)/0.1 = **37.5**

Average(29.375, 37.5) = **33.4375** minutes

Defuzzification (calculation)

Each time the wash machine initializes:

- **Measure weight height of clothes**
  - E.g. Height = 65 cm Weight = 3.6 kg
- **Fuzzification: Calculate the membership degree (fuzzy values) of weight and height (%).**
  - E.g. MF_{Low} (Height=65) = 0; MF_{Medium}(Height=65) = 0.75; MF_{High}(Height=65) = 0.25
    The MFs of height are [0, 0.75, 0.25] (%) [low, medium, high]
  - E.g. MF_{Light}(Weight=3.6) = 0.2; MF_{Medium}(Weight=3.6) = 0.8; MF_{Heavy}(Weight=3.6) = 0
    The MFs of weight are [0.2, 0.8, 0] (%) [light, medium, heavy]

Height \ Weight	Light	Medium	Heavy
Low	Short R1	Short R2	Medium R4
Medium	Short R3	Medium R5	Long R7
High	Medium R6	Long R8	Long R9

- **Inference: Use min-max composition/strategy to determine the MF degree (fuzzy values) for three classes (Short/Medium/Long) of washing time. E.g. Fire fuzzy rules (total 9 rules)**

& → min(MF(Height), MF(Weight))

1. WHEN Height = Low (0) & Weight = Light (0.2) THEN Time = Short (0)
2. WHEN Height = Low (0) & Weight = Medium (0.8) THEN Time = Short (0)
3. WHEN Height = Medium (0.75) & Weight = Light (0.2) THEN Time = Short (0.2)
4. WHEN Height = Low (0) & Weight = Heavy (0) THEN Time = Medium (0)
5. WHEN Height = Medium (0.75) & Weight = Medium (0.8) THEN Time = Medium (0.75)
6. WHEN Height = High (0.25) & Weight = Light (0.2) THEN Time = Medium (0.2)
7. WHEN Height = Medium (0.75) & Weight = Heavy (0) THEN Time = Long (0)
8. WHEN Height = High (0.25) & Weight = Medium (0.8) THEN Time = Long (0.25)
9. WHEN Height = High (0.25) & Weight = Heavy (0) THEN Time = Long (0)

MF_{Short}(Time) = max() = 0.2

MF_{Medium}(Time) = max() = 0.75

MF_{Long}(Time) = max() = 0.25

- **Defuzzification: Calculate the washing time in minutes using defuzzification strategies like MOM, COM, COA, etc.**
  - E.g. Use **MOM** strategy: $\begin{cases} wt_{left\ slop} = (1.6 + \mu)/0.08 & =(1.6+0.75)/0.08 = \mathbf{29.375} \\ wt_{right\ slop} = (4.5 - \mu)/0.1 & =(4.5-0.75)/0.1 = \mathbf{37.5} \end{cases}$ → **MOM(Time) = Average(29.375, 37.5) = 33.4375 minutes**

Workshop

- Open the A1234567X Donald Duck – Fuzzy Washing Machine python notebook file and complete the workshop.
- Run the workshop and see the result.
- Enhance the system based on requirement in workshop.

Workshop Submission

- **Naming convention: StudentID YourFullName Workshop Name, e.g. A234567X Donald Duck – sln – Fuzzy Washing Machine.ipynb/zip**
- **Use zip to a single file, then rename, if you plan to submit multiple files.**

References

1. **Fuzzy logic package “scikit-fuzzy 0.4.2”:**
   https://pythonhosted.org/scikit-fuzzy/

2. **More example <The Tipping Problem>:**
   https://pythonhosted.org/scikit-fuzzy/auto_examples/plot_tipping_problem.html#example-plot-tipping-problem-py

3.3 Deductive Reasoning (under Uncertainty) [Workshop]

3.3.1 Analogical Reasoning (Similarity)

Special thanks to Yan Wei Quan (A0215498U) for his contribution.

3.3.2 Fuzzy Logic Reasoning

Special thanks to Yan Wei Quan (A0215498U) for his contribution.

3.3.3 Abductive Reasoning (Probability)

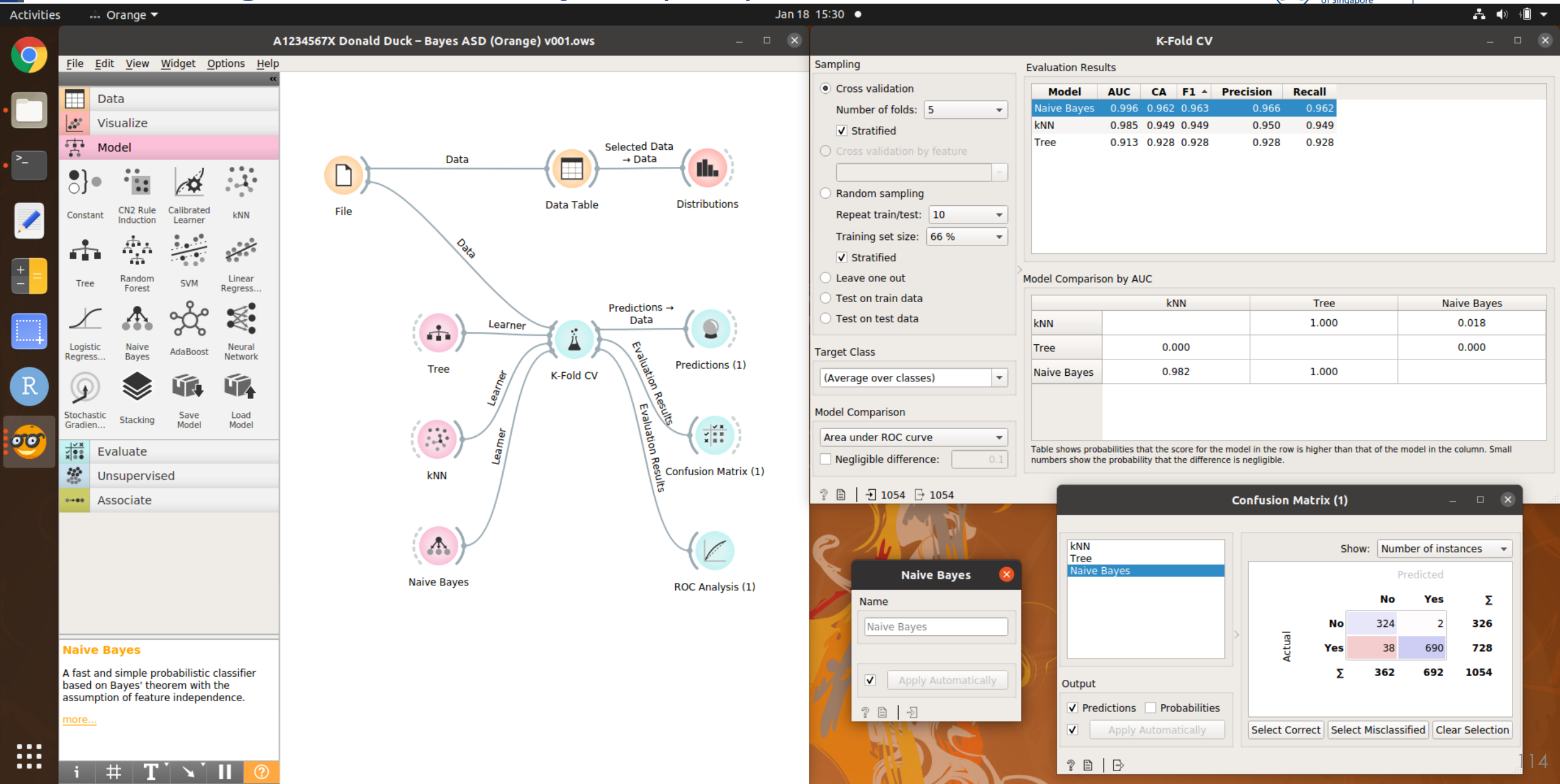
Special thanks to Yan Wei Quan (A0215498U) for his contribution.

3.3.4 Workshop Submission

Probabilistic Reasoning

- **Learning Objectives**
  - Understand how the Naïve Bayes algorithm works
  - Learn how to apply Naïve Bayes algorithm to learn and diagnose ASD cases for new children
  - Compare and analyze the results between decision tree model and Bayes model
- **Package to use,**
  - orange-canvas
  - Scikit-learn 0.20.3 in Python
- **Technique,**
  - Probability-based Machine Reasoning (Naïve Bayes algorithm)
- **Case/Scenario,**
  - Autistic Spectrum Disorder classification
- **Requirement,**
  - Familiar with Python

Orange's Naïve Bayes (NB)



Scikit's Naïve Bayes (NB)

- **Installation & Tutorial:**

1. **Type in pip install scikit-learn in your anaconda environment**
2. **Read documentations about scikit-learn in https://scikit-learn.org/stable/**
3. **Read documentations about NB in https://scikit-learn.org/stable/modules/classes.html?highlight=naive%20bayes#module-sklearn.naive_bayes**

Workshop

- Open the A1234567X Donald Duck – Bayes ASD python notebook file and complete the workshop.
- Run the workshop and see the result.
- [Optionally] compare the performance of NB with previous model we have learnt (e.g. Decision Tree) and write down what you finding in the comparison, e.g. which model's performance is better?

Workshop Submission

- **Naming convention: StudentID YourFullName Workshop Name, e.g. A234567X Donald Duck – sln – Bayes ASD.ipynb/zip**
- **Use zip to a single file, then rename, if you plan to submit multiple files.**

References

1. User guide of scikit learn NB model: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.naive_bayes

3.3 Deductive Reasoning (under Uncertainty) [Workshop]

3.3.1 Analogical Reasoning (Similarity)

Special thanks to Yan Wei Quan (A0215498U) for his contribution.

3.3.2 Fuzzy Logic Reasoning

Special thanks to Yan Wei Quan (A0215498U) for his contribution.

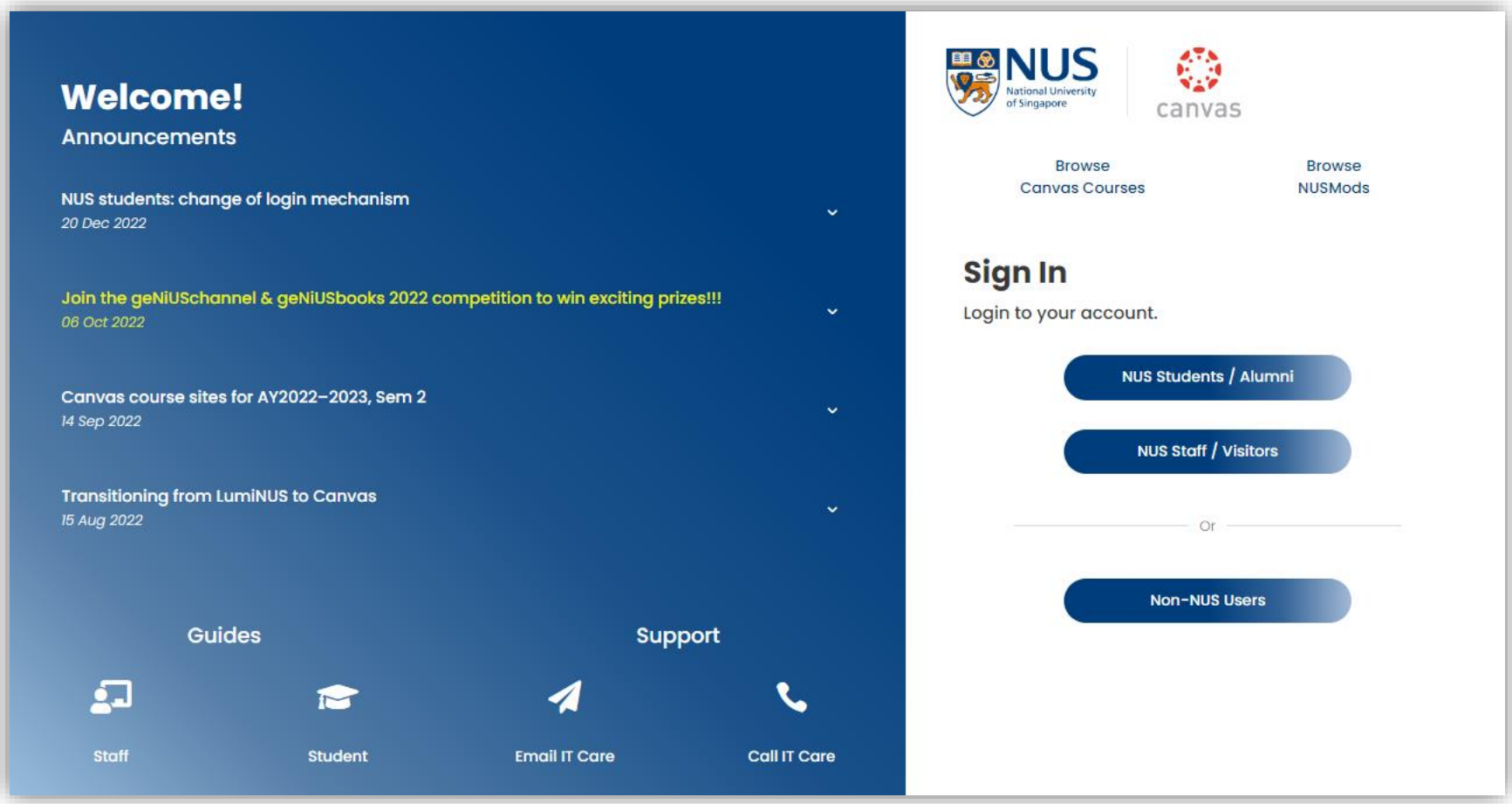
3.3.3 Abductive Reasoning (Probability)

Special thanks to Yan Wei Quan (A0215498U) for his contribution.

3.3.4 Workshop Submission

Workshop Submission

- **Naming convention: StudentID YourFullName**
- **Use zip to a single file, then rename, if you plan to submit multiple files.**



END OF NOTES

APPENDICES

[Optional] Deductive Reasoning by Logical Inference

Hello Kitty Example

Rules & Logical Inference

[Example] You are investigating a case:

- **Given below known intelligence (rules & facts):**
  1. Person "***Sam***" loves animals.
  2. Any person who loves animals does not kill an/that animal.
  3. All cats are animals.
  4. "***HelloKitty***" is a cat.
  5. Either one person "***Sam***" or "***Curiosity***" killed "***HelloKitty***".
- **Question: Who killed HelloKitty?**
- **Use the following predicates:**
  - Person(x) x is person.
  - Animal(x) x is animal.
  - Cat(x) x is cat.
  - Loves(x, y) x loves y.
  - Kills(x, y) x kills y.

Source
https://www.pinterest.com/pin/355925176775527097/

Rules & Logical Inference

Logical inference/calculation steps:

1. **Express the knowledge/sentences (rules & facts) in first order logic forms**
2. **Convert first order logic expressions to clauses (clausal form)**
3. **Conduct resolution (logical inference) using unification and variable renaming**

Rules & Logical Inference

Formal Logic – First Order Logic

- **Sentences in Knowledge Base (KB)**
  1. $\forall x$ Animal(x) → Loves(Sam, x)
  2. $\forall x \forall y$ Loves(x, y) $\wedge$ Person(x) $\wedge$ Animal(y) → $\neg$Kills(x, y)
  3. $\forall x$ Cat(x) → Animal(x)
  4. Cat(HelloKitty)
  5. Kills(Sam, HelloKitty) $\vee$ Kills(Curiosity, HelloKitty)
  6. Person(Sam)
  7. Person(Curiosity)
- **Clausal form conversion: p → q ≡ $\neg$p $\vee$ q**
  1. $\forall x$ Animal(x) → Loves(Sam, x) ≡ $\forall x$ $\neg$Animal(x) $\vee$ Loves(Sam, x)
  2. $\forall x \forall y$ Loves(x, y) $\wedge$ Person(x) $\wedge$ Animal(y) → $\neg$Kills(x, y) ≡ $\forall x \forall y \neg$[Loves(x, y) $\wedge$ Person(x) $\wedge$ Animal(y)] $\vee$ $\neg$Kills(x, y)
  3. $\forall x$ Cat(x) → Animal(x) ≡ $\forall x$ $\neg$Cat(x) $\vee$ Animal(x)

Rules & Logical Inference

- **Hypothesis/Clause to prove is : Curiosity killed HelloKitty.**

  α = Kills(Curiosity, HelloKitty)

- **Available reference of truth: Knowledge Base**

Rules & Logical Inference

Inference strategy to prove: KB ⊨ α (KB entails α) by:

- **Forward chaining: "KB → α"** is valid

  Validity is connected to inference via the Deduction Theorem:

  KB ⊨ α if and only if "KB → α" is valid, meaning sentence **α** is TRUE (can be deduced) in all KB models/variable-instantiations.

- **Backward chaining: "KB ∧ ¬α"** is unsatisfiable: = {empty set}

  Satisfiability is connected to inference via the following:

  KB ⊨ α if and only if "KB ∧ ¬α" is unsatisfiable, meaning sentence **¬α** is FALSE in all KB models/instantiations. (There is no model/variable-instantiation **{}**, for which KB and **¬α** are both TRUE, at the same time/model.)

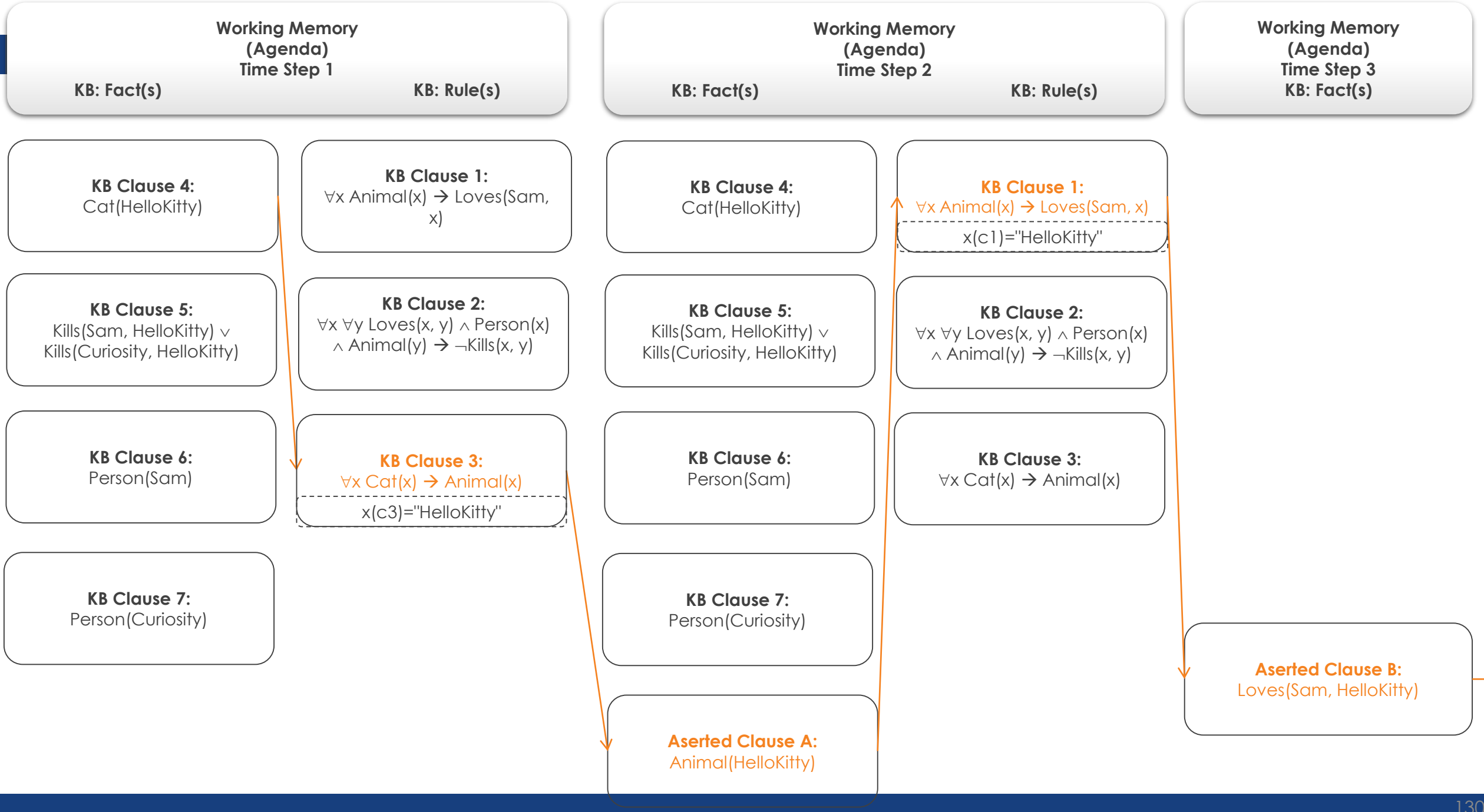
Rules & Logical Inference

Inference using Forward chaining

- **One conclusion to seek is HelloKitty.** **: Curiosity killed**

  α = Kills(Curiosity, HelloKitty)

- **Forward chaining: "KB → α"** is valid


Working Memory
(Agenda)
Time Step 1
KB: Fact(s)
KB: Rule(s)
Working Memory
(Agenda)
Time Step 2
KB: Fact(s)
KB: Rule(s)
Working Memory
(Agenda)
Time Step 3
KB: Fact(s)
KB Clause 4:
Cat(HelloKitty)
KB Clause 1:
∀x Animal(x) → Loves(Sam, x)
KB Clause 5:
Kills(Sam, HelloKitty) ∨ Kills(Curiosity, HelloKitty)
KB Clause 2:
∀x ∀y Loves(x, y) ∧ Person(x) ∧ Animal(y) → ¬Kills(x, y)
KB Clause 6:
Person(Sam)
KB Clause 3:
∀x Cat(x) → Animal(x)
x(c3)="HelloKitty"
KB Clause 7:
Person(Curiosity)
KB Clause 4:
Cat(HelloKitty)
KB Clause 1:
∀x Animal(x) → Loves(Sam, x)
x(c1)="HelloKitty"
KB Clause 5:
Kills(Sam, HelloKitty) ∨ Kills(Curiosity, HelloKitty)
KB Clause 2:
∀x ∀y Loves(x, y) ∧ Person(x) ∧ Animal(y) → ¬Kills(x, y)
KB Clause 6:
Person(Sam)
KB Clause 3:
∀x Cat(x) → Animal(x)
KB Clause 7:
Person(Curiosity)
Aserted Clause A:
Animal(HelloKitty)
Aserted Clause B:
Loves(Sam, HelloKitty)

Working Memory (Agenda) Time Step 2
KB: Rule(s)

KB Clause 1:
∀x Animal(x) → Loves(Sam, x)
x(c1)="HelloKitty"

KB Clause 2:
∀x ∀y Loves(x, y) ∧ Person(x) ∧ Animal(y) → ¬Kills(x, y)

KB Clause 3:
∀x Cat(x) → Animal(x)

Working Memory (Agenda) Time Step 3
KB: Fact(s) **KB: Rule(s)**

KB Clause 4:
Cat(HelloKitty)

KB Clause 5:
Kills(Sam, HelloKitty) ∨ Kills(Curiosity, HelloKitty)

KB Clause 6:
Person(Sam)

KB Clause 7:
Person(Curiosity)

Aserted Clause A:
Animal(HelloKitty)

Aserted Clause B:
Loves(Sam, HelloKitty)

KB Clause 1:
∀x Animal(x) → Loves(Sam, x)

KB Clause 2:
∀x ∀y Loves(x, y) ∧ Person(x) ∧ Animal(y) → ¬Kills(x, y)
x(c2)="Sam" y(c2)="HelloKitty"

KB Clause 3:
∀x Cat(x) → Animal(x)

Working Memory (Agenda) Time Step 4
KB: Fact(s) **KB: Rule(s)**

KB Clause 4:
Cat(HelloKitty)

KB Clause 5:
Kills(Sam, HelloKitty) ∨ Kills(Curiosity, HelloKitty)

KB Clause 6:
Person(Sam)

KB Clause 7:
Person(Curiosity)

Aserted Clause A:
Animal(HelloKitty)

Aserted Clause B:
Loves(Sam, HelloKitty)

Aserted Clause C:
¬Kills(Sam, HelloKitty)

Aserted Clause D:
Kills(Curiosity, HelloKitty)

KB Clause 1:
∀x Animal(x) → Loves(Sam, x)

KB Clause 2:
∀x ∀y Loves(x, y) ∧ Person(x) ∧ Animal(y) → ¬Kills(x, y)

KB Clause 3:
∀x Cat(x) → Animal(x)

"KB → a" is valid


Working Memory
(Agenda)
Time Step 1
KB: Fact(s)
KB: Rule(s)
KB Clause 4:
Cat(HelloKitty)
KB Clause 1:
∀x Animal(x) → Loves(Sam, x)
KB Clause 5:
Kills(Sam, HelloKitty) ∨
Kills(Curiosity, HelloKitty)
KB Clause 2:
∀x ∀y Loves(x, y) ∧ Person(x) ∧
Animal(y) → ¬Kills(x, y)
KB Clause 6:
Person(Sam)
KB Clause 3:
∀x Cat(x) → Animal(x)
x(c3)="HelloKitty"
KB Clause 7:
Person(Curiosity)
Working Memory
(Agenda)
Time Step 2
KB: Fact(s)
KB: Rule(s)
KB Clause 4:
Cat(HelloKitty)
KB Clause 1:
∀x Animal(x) → Loves(Sam, x)
x(c1)="HelloKitty"
KB Clause 5:
Kills(Sam, HelloKitty) ∨
Kills(Curiosity, HelloKitty)
KB Clause 2:
∀x ∀y Loves(x, y) ∧ Person(x) ∧
Animal(y) → ¬Kills(x, y)
KB Clause 6:
Person(Sam)
KB Clause 3:
∀x Cat(x) → Animal(x)
KB Clause 7:
Person(Curiosity)
Aserted Clause A:
Animal(HelloKitty)
Working Memory
(Agenda)
Time Step 3
KB: Fact(s)
KB: Rule(s)
KB Clause 4:
Cat(HelloKitty)
KB Clause 1:
∀x Animal(x) → Loves(Sam, x)
KB Clause 5:
Kills(Sam, HelloKitty) ∨
Kills(Curiosity, HelloKitty)
KB Clause 2:
∀x ∀y Loves(x, y) ∧ Person(x) ∧
Animal(y) → ¬Kills(x, y)
x(c2)="Sam" y(c2)="HelloKitty"
KB Clause 6:
Person(Sam)
KB Clause 3:
∀x Cat(x) → Animal(x)
KB Clause 7:
Person(Curiosity)
Aserted Clause A:
Animal(HelloKitty)
Aserted Clause B:
Loves(Sam, HelloKitty)
Working Memory
(Agenda)
Time Step 4
KB: Fact(s)
KB: Rule(s)
KB Clause 4:
Cat(HelloKitty)
KB Clause 1:
∀x Animal(x) → Loves(Sam, x)
KB Clause 5:
Kills(Sam, HelloKitty) ∨
Kills(Curiosity, HelloKitty)
KB Clause 2:
∀x ∀y Loves(x, y) ∧ Person(x) ∧
Animal(y) → ¬Kills(x, y)
KB Clause 6:
Person(Sam)
KB Clause 3:
∀x Cat(x) → Animal(x)
KB Clause 7:
Person(Curiosity)
Aserted Clause A:
Animal(HelloKitty)
Aserted Clause B:
Loves(Sam, HelloKitty)
Aserted Clause C:
¬Kills(Sam, HelloKitty)
Aserted Clause D:
Kills(Curiosity, HelloKitty)

Rules & Logical Inference

Inference using Backward chaining

- **Hypothesis to prove is : Curiosity killed HelloKitty.**

  α = Kills(Curiosity, HelloKitty)

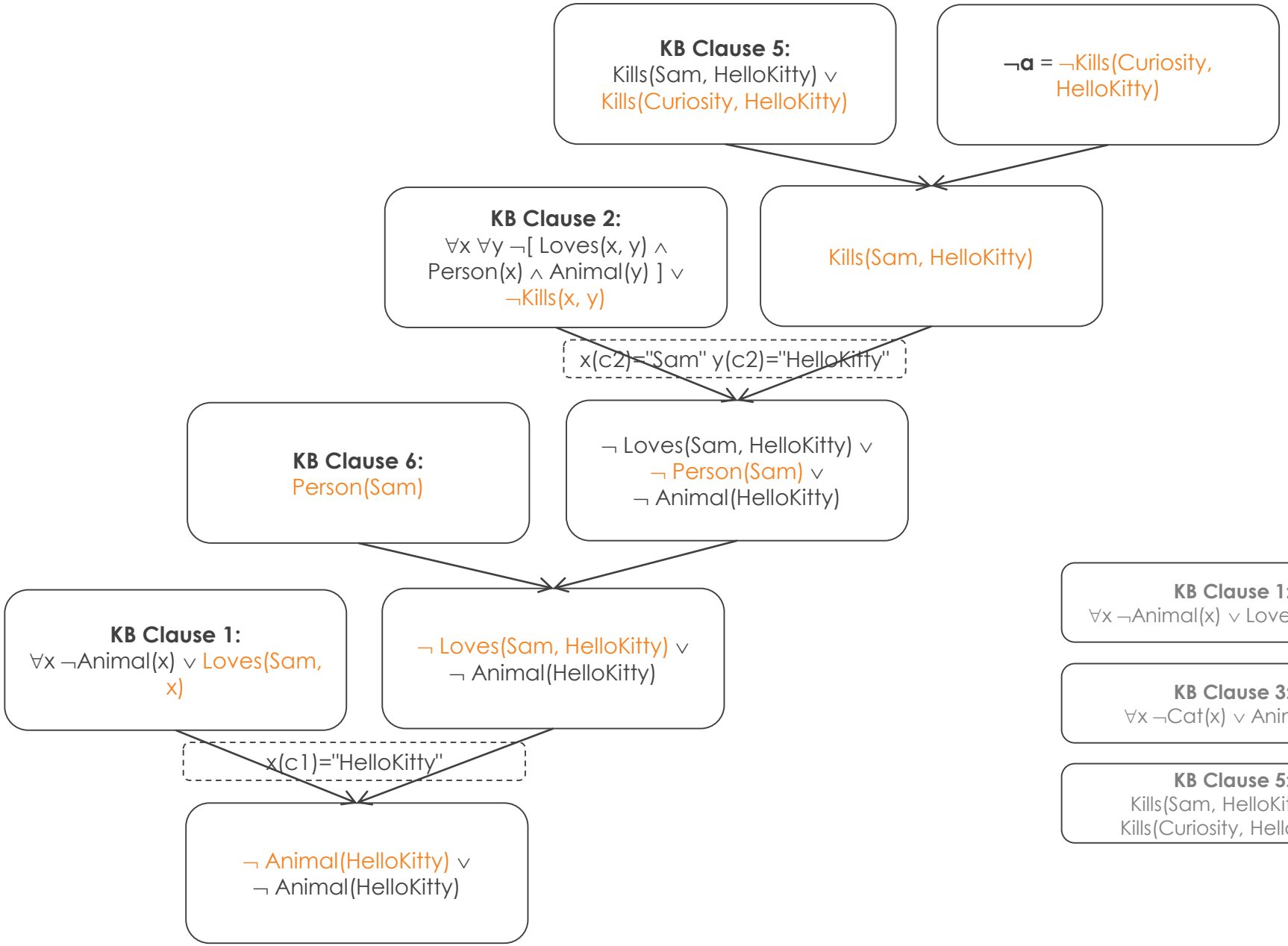
- **Refutation of hypothesis/Clause is : Curiosity didn't kill HelloKitty.**

  ¬α = ¬Kills(Curiosity, HelloKitty)

- **Backward chaining: "KB ∧ ¬ α"** is unsatisfiable: = {empty set}

  proof by refutation

KB ∧ ¬ α



KB:

KB Clause 1:
∀x ¬Animal(x) ∨ Loves(Sam, x)

KB Clause 2:
∀x ∀y ¬[Loves(x, y) ∧ Person(x) ∧ Animal(y)] ∨ ¬Kills(x, y)

KB Clause 3:
∀x ¬Cat(x) ∨ Animal(x)

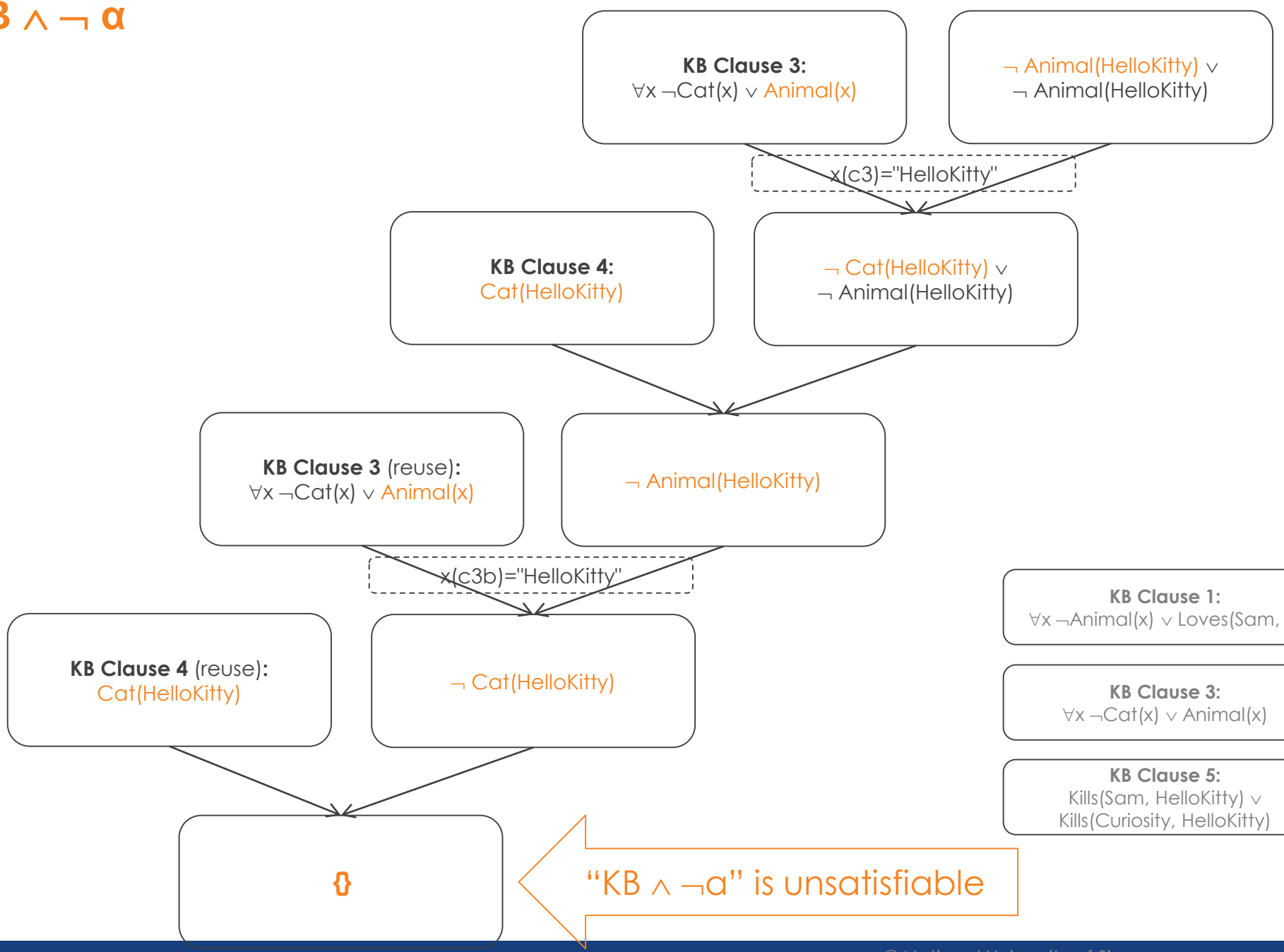
KB Clause 4:
Cat(HelloKitty)

KB Clause 5:
Kills(Sam, HelloKitty) ∨
Kills(Curiosity, HelloKitty)

KB Clause 6:
Person(Sam)

KB Clause 7:
Person(Curiosity)

KB ∧ ¬ α



KB:

KB Clause 1:
∀x ¬Animal(x) ∨ Loves(Sam, x)

KB Clause 2:
∀x ∀y ¬[Loves(x, y) ∧ Person(x) ∧ Animal(y)] ∨ ¬Kills(x, y)

KB Clause 3:
∀x ¬Cat(x) ∨ Animal(x)

KB Clause 4:
Cat(HelloKitty)

KB Clause 5:
Kills(Sam, HelloKitty) ∨
Kills(Curiosity, HelloKitty)

KB Clause 6:
Person(Sam)

KB Clause 7:
Person(Curiosity)

Rules & Logical Inference

- **Proof by refutation: KB ∧ ¬α = {empty set}**
- **Conclusion:**

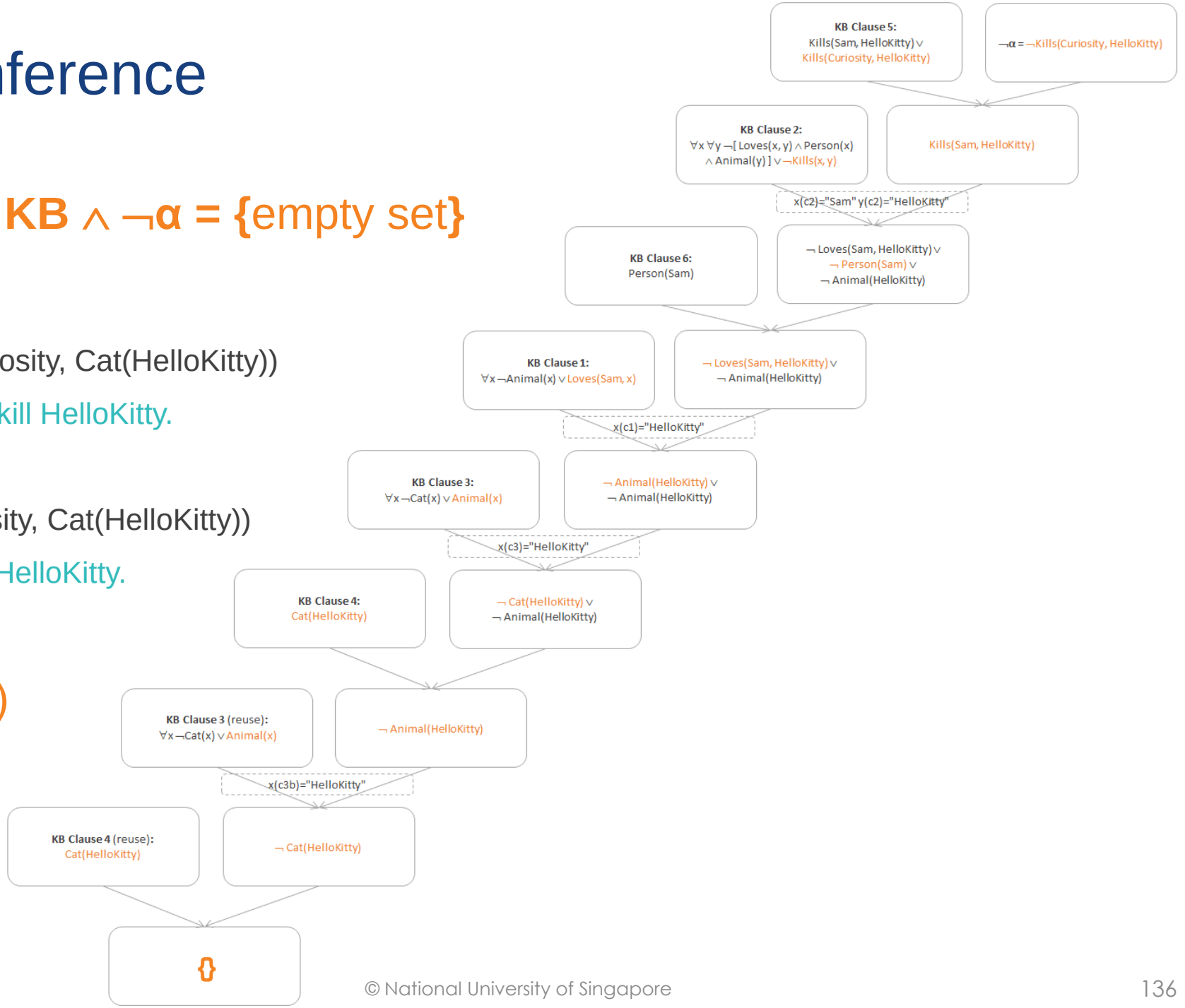
We reject: ¬α = ¬Kills(Curiosity, Cat(HelloKitty))

Curiosity didn't kill HelloKitty.

But accept: α = Kills(Curiosity, Cat(HelloKitty))

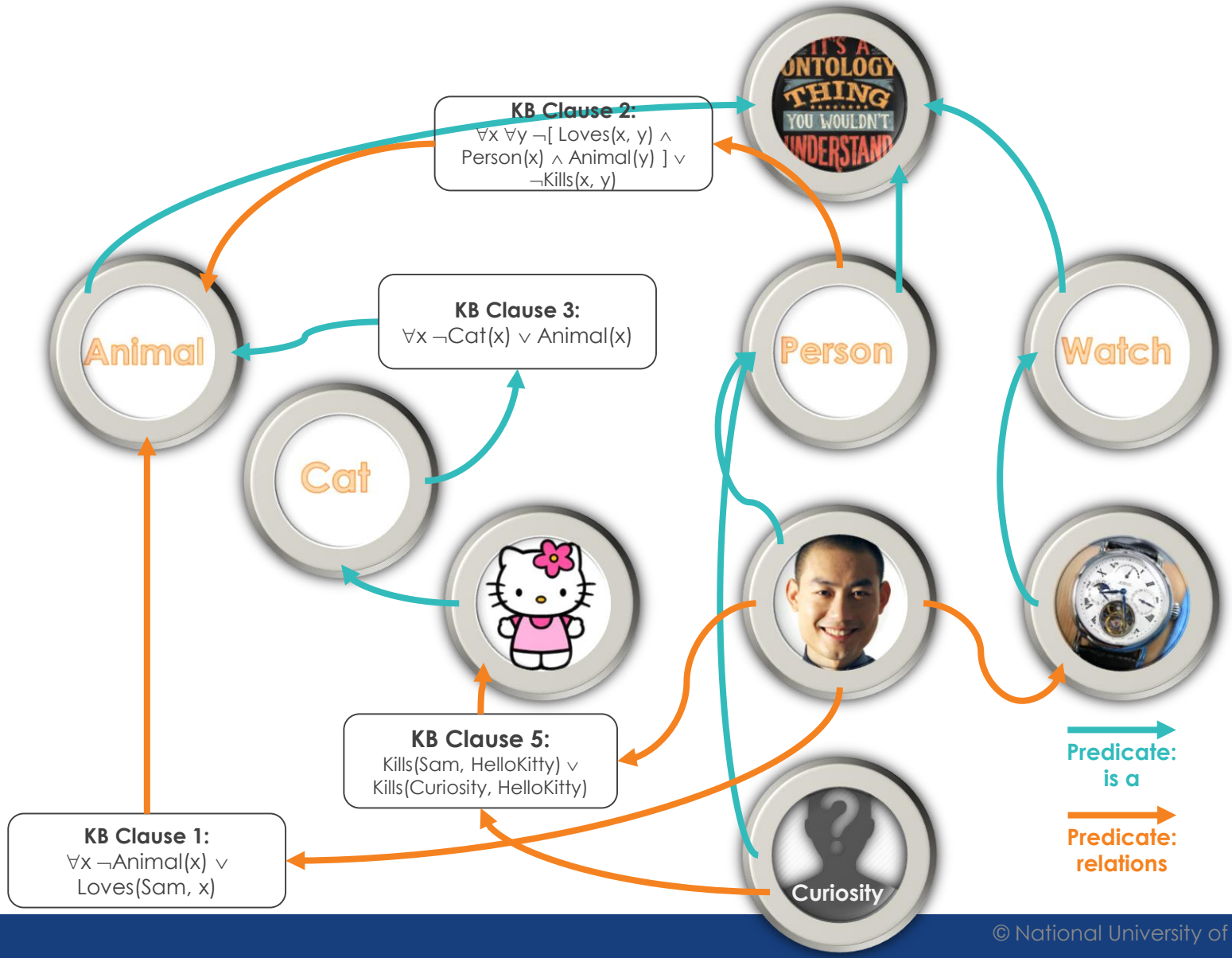
Curiosity killed HelloKitty.

- **KB ⊨ α** (KB entails α)



Rules & Logical Inference

Ontology



KB:

KB Clause 4:
Cat(HelloKitty)

KB Clause 1:
∀x ¬Animal(x) ∨ Loves(Sam, x)

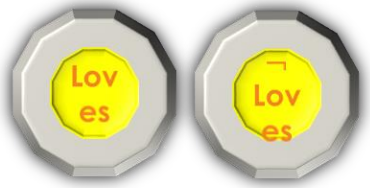
KB Clause 5:
Kills(Sam, HelloKitty) ∨ Kills(Curiosity, HelloKitty)

KB Clause 2:
∀x ∀y ¬[Loves(x, y) ∧ Person(x) ∧ Animal(y)] ∨ ¬Kills(x, y)

KB Clause 6:
Person(Sam)

KB Clause 3:
∀x ¬Cat(x) ∨ Animal(x)

KB Clause 7:
Person(Curiosity)



Aserted Clause A:
Animal(HelloKitty)

Aserted Clause B:
Loves(Sam, HelloKitty)

Aserted Clause C:
¬Kills(Sam, HelloKitty)

Aserted Clause D:
Kills(Curiosity, HelloKitty)

Hyper Edges

[Optional] Deductive Reasoning by Logical Inference [Workshop]

<Hello Kitty>

<Guess the Animal>

Rule-based Reasoning System

- **Learning Objectives**
  - Understand rule-based system in machine reasoning
  - Learn how to represent formal logic – first order logic and do forward chaining & backward chaining in Pyke
  - Build your own reasoning system in WhatTheAnimalIs scenario.
- **Package to use,**
  - Pyke 1.1.1 in Python
- **Technique,**
  - Rule-based inference (forward chaining & backward chaining)
- **Case/Scenario,**
  - Who kill the Hellokitty
- **Requirement,**
  - Familiar with Python

Before the workshop

- **Installation:**

1. **go to https://sourceforge.net/projects/pyke/files/pyke/1.1.1/**
   - - download pyke3-1.1.1.zip if your Python version is 3.x
   - - download pyke-1.1.1.zip if your Python version is 2.x

2. **After unzipping the file, go into the directory and run:**
   - python setup.py build
   - python setup.py install

- **Read documentations about Pyke in http://pyke.sourceforge.net/**

.kfb file – knowledge fact base

Each .krb file is a knowledge base containing some facts.

The syntax of fact is

fact_name(argument 1, argument 2, …., argument n)

Example:

For statement Sam loves Hellokitty, the rule can be set as:

Love(Sam, Hellokitty, True)

Here, Pyke don't know what the rule means, it will only check if the syntax of rule is correct. However, the structure

here is fixed implicitly by yourself.

Love(Subject, Object, Flag)

If you have another statement like Curiosity does not love Donaldduck, the rule will be

Love(Curiosity, Donaldduck, False)

.krb file – knowledge rule base

- **Each .krb file is a knowledge base containing some rules.**

- **Importance note: The syntax of forward chaining rule and backward chaining rule are different.**

How does PyKe inference engine work?

For rules, Pyke engine will automatically define python functions (under compiled_krb folder) according to what you write down in .krb file.

For example, when you define new engine, `knowledge_engine.engine(__file__)`

it recodes the rules in .krb file like  **into** 

All the rules are reformatted into python functions so when it tries to do the inference, it just calls the python functions.

The facts  **will be reformatted into .fbc**  **file for engine to read them quickly.**

Forward Chaining Rule

The syntax of forward chaining rule is,

rule_name_fc

foreach

kb.statement 1

kb.statement 2

kb.statement 3

assert

kb.statement 4

kb.statement 5

It means,

for rule_name_fc, if statement 1 & statement 2 & statement 3, then statement 4 & statement 5. kb. means statement comes from kb.

Backward Chaining Rule

The syntax of backward chaining rule is,

```
rule_name_bc
    use
        statement 1 (only one statement)
    when
        kb.statement 2
        kb.statement 3
```

It means,

for rule_name_bc, statement 1 will be true if statement 2 & statement 3. kb. means statement comes from kb.

Generalize the rule

Sometimes, you want to increase the generalizability of the rule. Like Sam loves all the animal. In the Pyke, '$' is used to represent anything.

For example,

Statement: Sam loves all animal

Can be converted into,

```
SamLoveAnimal                          # rule name
    foreach
        isAnimal($creature)            # fact or statement
    assert
        Love(Sam, $creature, True)     # fact or statement
```

Extra function

Sometimes, you will want to call Python function in the rules to implement more complex functions.

You can define extra function in fc.krb like,

```
fc_extras #This is keyword in Pyke, you cannot change the name
    def your_function(arguments):
        #do something
         return something
```

and similar in bc.krb,

```
bc_extras #This is keyword in Pyke, you cannot change the name
    def your_function(arguments):
        #do something
         return something
```

Extra function (2)

How to use the function in your rule?

Let's say either Sam or Curiosity kills the Hello Kitty. We can use extra function to implement the OR expression.

```
fc_extras
    def either(creature1, flag1):
        dict_person = {'Sam': 'Curiosity', 'Curiosity': 'Sam'}
        creature2 = dict_person[creature1]
        if flag1:
            flag2 = False
        else:
            flag2 = True
        return creature2, flag2
```

Then use the extra function in the rule

```
EitherSamorCuriosityKillHelloKitty_fc
    foreach
        hellokitty_facts.Kills($creature1, Hellokitty, $flag1)
        ($creature2, $flag2) = either($creature1, $flag1)
    assert
        hellokitty_facts.Kills($creature2, Hellokitty, $flag2)
```

Example: Who kill Hello Kitty?

- **Sentences in Knowledge Base (KB)**
  1. $\forall x$ Animal(x) → Loves(Sam, x)
  2. $\forall x \forall y$ Loves(x, y) $\wedge$ Person(x) $\wedge$ Animal(y) → $\neg$Kills(x, y)
  3. $\forall x$ Cat(x) → Animal(x)
  4. Cat(HelloKitty)
  5. Kills(Sam, HelloKitty) $\vee$ Kills(Curiosity, HelloKitty)
  6. Person(Sam)
  7. Person(Curiosity)
- **Clausal form conversion: p → q ≡ $\neg$p $\vee$ q**
  1. $\forall x$ Animal(x) → Loves(Sam, x) ≡ $\forall x$ $\neg$Animal(x) $\vee$ Loves(Sam, x)
  2. $\forall x \forall y$ Loves(x, y) $\wedge$ Person(x) $\wedge$ Animal(y) → $\neg$Kills(x, y) ≡ $\forall x \forall y \neg$[Loves(x, y) $\wedge$ Person(x) $\wedge$ Animal(y)] $\vee$ $\neg$Kills(x, y)
  3. $\forall x$ Cat(x) → Animal(x) ≡ $\forall x$ $\neg$Cat(x) $\vee$ Animal(x)

We have seen the Hello Kitty example.

Let's use Pyke logical inference engine to find out who kill the Hello Kitty.

Write down the facts into .kfb file

- For example,

  Fact 1: Hello Kitty is cat

isCat(Hellokitty, True)

See the hellokitty_facts.kfb file for all the facts

Write down the rules into .krb file

- **For example,**

  Rule 1: All the cats are animal

```
Cat_is_Animal_fc
    foreach
        hellokitty_facts.isCat($creature)
    assert

hellokitty_facts.isAnimal($creature)
```

See the hellokitty_fc.krb & hellokitty_bc.krb file for all the rules

[Optional] Workshop

- **Open the A1234567X Donald Duck – Hello Kitty python notebook file and complete the workshop.**
- **Run the workshop and see the result.**

[Optional] Workshop Submission

- **Naming convention: StudentID YourFullName Workshop Name, e.g. A234567X Donald Duck – sln – Hello Kitty.ipynb/zip**
- **Use zip to a single file, then rename, if you plan to submit multiple files.**

References

1. Pyke, python knowledge engine: http://pyke.sourceforge.net/

[Optional] Deductive Reasoning by Logical Inference [Workshop]

<Hello Kitty>

<Guess the Animal>

3.1 MACHINE INFERENCE

Exercise – A possible solution: propositional logic

1. WHEN BodyCoverType = HasHair THEN AnimalType = Mammal
2. WHEN FeedType = FeedMilk THEN AnimalType = Mammal
3. WHEN BodyCoverType = HasFeather THEN AnimalType = Bird
4. WHEN MoveType = CanFly AND ReproduceType = LayEgg THEN AnimalType = Bird
5. WHEN AnimalType = Mammal AND FeedType = EatMeat THEN AnimalType = Carnivore
6. WHEN AnimalType = Mammal AND ToothType = HasPointedTeeth AND FootType = HasClaws AND EyeType = HasForwardEyes THEN AnimalType = Carnivore
7. WHEN AnimalType = Mammal AND FeedType = EatGrass THEN AnimalType = Herbivore
8. WHEN AnimalType = Mammal AND FootType = HasHooves THEN AnimalType = Herbivore
9. WHEN AnimalType = Carnivore AND ColorType = HasColorTawny AND MarkingType = HasDarkSpots THEN AnimalName = Cheetah
10. WHEN AnimalType = Carnivore AND ColorType = HasColorTawny AND MarkingType = HasDarkStripes THEN AnimalName = Tiger
11. WHEN AnimalType = Herbivore AND ColorType = HasColorTawny AND MarkingType = HasDarkSpots AND NeckType = HasLongNeck THEN AnimalName = Giraffe
12. WHEN AnimalType = Herbivore AND ColorType = HasColorBlackWhite THEN AnimalName = Zebra
13. WHEN AnimalType = Bird AND MoveType = CanWalk AND ColorType = HasColorBlackWhite AND NeckType = HasLongNeck THEN AnimalName = Ostrich
14. WHEN AnimalType = Bird AND MoveType = CanWwim AND ColorType = HasColorBlackWhite THEN AnimalName = Penguin
15. WHEN AnimalType = Bird AND MoveType = CanFly AND ColorType = HasColorBlackWhite THEN AnimalName = Albatross

3.1 MACHINE INFERENCE

Exercise – A possible solution: first order logic

X is an (instance of) animal (class).

1. HasHair(X) → Mammal(X)
2. FeedMilk(X) → Mammal(X)
3. HasFeather(X) → Bird(X)
4. CanFly(X) ^ LayEgg(X) → Bird(X)
5. Mammal(X) ^ EatMeat(X) → Carnivore(X)
6. Mammal(X) ^ HasPointedTeeth(X) ^ HasClaws(X) ^ HasForwardEyes(X) → Carnivore(X)
7. Mammal(X) ^ EatGrass(X) → Herbivore(X)
8. Mammal(X) ^ HasHooves(X) → Herbivore(X)
9. Carnivore(X) ^ HasColorTawny(X) ^ HasDarkSpots(X) → Cheetah(X)
10. Carnivore(X) ^ HasColorTawny(X) ^ HasDarkStripes(X) → Tiger(X)
11. Herbivore(X) ^ HasColorTawny(X) ^ HasDarkSpots(X) ^ HasLongNeck(X) → Giraffe(X)
12. Herbivore(X) ^ HasColorBlackWhite(X) → Zebra(X)
13. Bird(X) ^ CanWalk(X) ^ HasColorBlackWhite(X) ^ HasLongNeck(X) → Ostrich(X)
14. Bird(X) ^ CanSwim(X) ^ HasColorBlackWhite(X) → Penguin(X)
15. Bird(X) ^ CanFly(X) ^ HasColorBlackWhite(X) → Albatross(X)

3.1 MACHINE INFERENCE

Exercise – A possible solution: first order logic

- **Knowledge Base (KB)**
  - Rule sets plus below Facts:
  - HasHair(X)
  - HasClaws(X)
  - HasPointedTeeth(X)
  - HasForwardEyes(X)
  - HasColorTawny(X)
  - HasDarkSpots(X)
- **Clause form conversion : $p \rightarrow q \equiv \neg p \vee q$**
- **Hypothesis to prove is : α = Cheetah(X)**
- **Refutation of hypothesis is : $\neg\alpha = \neg$Cheetah(X)**

3.1 MACHINE INFERENCE

Exercise – A possible solution: first order logic

1. HasHair(X) → Mammal(X)
2. FeedMilk(X) → Mammal(X)
3. HasFeather(X) → Bird(X)
4. CanFly(X) ^ LayEgg(X) → Bird(X)
5. Mammal(X) ^ EatMeat(X) → Carnivore(X)
6. Mammal(X) ^ HasPointedTeeth(X) ^ HasClaws(X) ^ HasForwardEyes(X) → Carnivore(X)
7. Mammal(X) ^ EatGrass(X) → Herbivore(X)
8. Mammal(X) ^ HasHooves(X) → Herbivore(X)
9. Carnivore(X) ^ HasColorTawny(X) ^ HasDarkSpots(X) → Cheetah(X)
10. Carnivore(X) ^ HasColorTawny(X) ^ HasDarkStripes(X) → Tiger(X)
11. Herbivore(X) ^ HasColorTawny(X) ^ HasDarkSpots(X) ^ HasLongNeck(X) → Giraffe(X)
12. Herbivore(X) ^ HasColorBlackWhite(X) → Zebra(X)
13. Bird(X) ^ CanWalk(X) ^ HasColorBlackWhite(X) ^ HasLongNeck(X) → Ostrich(X)
14. Bird(X) ^ CanSwim(X) ^ HasColorBlackWhite(X) → Penguin(X)
15. Bird(X) ^ CanFly(X) ^ HasColorBlackWhite(X) → Albatross(X)

Facts:
HasHair(X)
HasClaws(X)
HasPointedTeeth(X)
HasForwardEyes(X)
HasColorTawny(X)
HasDarkSpots(X)

[Optional] Workshop

According to the example – WhoKillHellokitty, try to build a rule-based system in this scenario.

Given the facts and rules, try to find out which animal the creature is.

Forward chaining and backward chaining are both required.

Facts:
HasHair($creature)
HasClaws(($creature)
HasPointedTeeth(($creature)
HasForwardEyes(($creature)
HasColorTawny(($creature)
HasDarkSpots(($creature)

Rules:

HasHair(X) → Mammal(X)

FeedMilk(X) → Mammal(X)

HasFeather(X) → Bird(X)

CanFly(X) ^ LayEgg(X) → Bird(X)

Mammal(X) ^ EatMeat(X) → Carnivore(X)

Mammal(X) ^ HasPointedTeeth(X) ^ HasClaws(X) ^ HasForwardEyes(X) → Carnivore(X)

Mammal(X) ^ EatGrass(X) → Herbivore(X)

Mammal(X) ^ HasHooves(X) → Herbivore(X)

Carnivore(X) ^ HasColorTawny(X) ^ HasDarkSpots(X) → Cheetah(X)

Carnivore(X) ^ HasColorTawny(X) ^ HasDarkStripes(X) → Tiger(X)

Herbivore(X) ^ HasColorTawny(X) ^ HasDarkSpots(X) ^ HasLongNeck(X) → Giraffe(X)

Herbivore(X) ^ HasColorBlackWhite(X) → Zebra(X)

Bird(X) ^ CanWalk(X) ^ HasColorBlackWhite(X) ^ HasLongNeck(X) → Ostrich(X)

Bird(X) ^ CanSwim(X) ^ HasColorBlackWhite(X) → Penguin(X)

Bird(X) ^ CanFly(X) ^ HasColorBlackWhite(X) → Albatross(X)

[Optional] Reasoning under Uncertainty

Certainty Factor

Certainty Factor

Certainty Factor (CF)

- **It allows experts to fairly easily express their personal "probability" and, it also allows analyst to easily incorporate them in machine reasoning systems**
- **Certainty Factors are measures of belief or how much confidence we have in the data/information**
- **Certainty Factors can be incorporated into Rules and Facts.**
- **Typically CF range: -1.0 ≤ CF ≤ +1.0**
  - CF = +1.0 The rule/fact is certainly true.
  - CF = 0.0 We don't know whether it is true or not.
  - CF = -1.0 The rule/fact is certainly false.

Certainty Factor

Certainty Factors in Rules

- **CF in a rule represents the expert's confidence or belief in that chunk of knowledge.**
- **Rules with CF has the following structure:**

```
IF good_earnings THEN share_up {cf 0.7}
IF win_contract THEN share_up {cf 0.9}
```

- **If the condition is true then the conclusion is known to be true (proportional to the strength of the CF).**
- **CF can be elicited by "How confident are you that good earnings will cause the share price to go up?".**

Certainty Factor

Certainty Factors in Facts

- **CF in facts represents the expert's or user's belief in that piece of information:**

```
good_earnings {cf -0.7}
win_contract {cf 0.8}
```

- **Facts can consist of evidence, observations, intuition, therefore fact CF can be subjective.**
- **It can also be based on probability or obtained through statistical analysis and surveys.**
- **CF can be elicited by "What is the chance of the company winning the contract?".**

Certainty Factor

Uncertain Terms Interpretation

Definitely NOT	-1.0
Almost Certainly NOT	-0.8
Probably NOT	-0.6
Maybe NOT	-0.4
UNKNOWN	-0.2 to +0.2
Maybe	+0.4
Probably	+0.6
Almost Certainly	+0.8
Definitely	+1.0

Certainty Factor

Reasoning with Certainty Factors

- **Certainty factors are propagated (calculated) through the reasoning chain when rules are fired.**
- **The following is a typical sequence of CF propagation:**
  1. User inputs a fact with a certainty value.
  2. All applicable rules are activated, ready to fire.
  3. When a rule is fired, the net rule certainty is calculated.
  4. When many rules are fired, their combined net certainty value is calculated.
  5. Final rule conclusion is then given with a merged single certainty value.

Certainty Factor

Finding the Net Certainty of a Rule

- **When a rule is fired, the net certainty of the rule conclusion is calculated as follows:**

$$cf(H,E) = cf(E) * cf(R)$$

cf(H,E) – Net Certainty of the rule conclusion
cf(E) – Certainty of the fact (rule input)
cf(R) – Certainty of the rule

For example:

*IF earnings=good THEN shares=up {cf **0.7**}*

and the current certainty of earnings=good is **0.8**, then

***cf** (H,E) = 0.8 x 0.7 = 0.56*

This result can be interpreted as "shares will probably go up".

Certainty Factor

Conjunctive Evidences

- **For rules with conjunctive evidences the certainty of the hypothesis *H* is calculated as follows:**

$$cf\,(H, E_1 \cap E_2 \cap \ldots \cap E_n) = min\,[cf\,(E_1), cf\,(E_2), \ldots, cf\,(E_n)] \times cf$$

For example:

IF earnings=good ***AND*** *contract=big THEN shares=up {cf* ***0.9****}*

current certainty of earnings=good is **0.8**, and contract=big is **0.1** then

$$cf\,(H, E_1 \cap E_2) = min[0.8, 0.1] \times 0.9 = 0.1 \times 0.9 = 0.09$$

This result can be interpreted as "**it is unknown if shares will go up**"

Certainty Factor

Disjunctive Evidences

- **For rules with disjunctive evidences the certainty of the hypothesis *H* is calculated as follows:**

$$cf\,(H, E1 \cup E2 \cup \ldots \cup E_n) = max\,[cf\,(E_1), cf\,(E_2), \ldots, cf\,(E_n)] \times cf$$

For example:

IF earnings=good **OR** *contract=big THEN shares=up {cf* ***0.9****}*

current certainty of earnings=good is **0.8**, and contract=big is **0.1** then

$$cf\,(H, E_1 \cup E_2) = max[0.8, 0.1] \times 0.9 = 0.8 \times 0.9 = 0.72$$

This result can be interpreted as "**shares will most probably go up**"

Certainty Factor

Combining Multiple Conclusions

- When rules are fired, they **insert/assert** their respective *cf(H,E)* into working memory
- When the same Hypothesis *H* is asserted by two or more rules, e.g. $cf(H,E_1)$ ... $cf(H,E_n)$, all *cfs* are combined to a single *cf(H)*

For example:

*IF earnings=good THEN shares=up {cf **0.7**}*

*IF contract=big THEN shares=up {cf **0.9**}*

and earnings=good is **0.8**, and contract=big is **0.1** then

$cf(H,E_1) = 0.56$ $cf(H,E_2) = 0.09$

- What will be the advice? "**share will probably go up**" or "**it is unknown**"

Certainty Factor

Combining Multiple Conclusions

- **When several rules are fired that lead to the same conclusion, we combine them as follows:**

$$cf(cf_1, cf_2) = \begin{cases} cf_1 + cf_2 \times (1 - cf_1) & \text{if } cf_1 > 0 \text{ and } cf_2 > 0 \\ \dfrac{cf_1 + cf_2}{1 - min\left[|cf_1|, |cf_2|\right]} & \text{if } cf_1 < 0 \text{ or } cf_2 < 0 \\ cf_1 + cf_2 \times (1 + cf_1) & \text{if } cf_1 < 0 \text{ and } cf_2 < 0 \end{cases}$$

$cf_1 = cf(H, E_1)$ **is the net certainty of rule 1 conclusion**

$cf_2 = cf(H, E_2)$ **is the net certainty of rule 2 conclusion**

Certainty Factor

Certainty Factor Exercise

R1: **IF** dividends=yes **AND** mgnt=good **AND** earnings=positive
THEN **buy=yes (0.6)**

R2: **IF** contract=large
THEN **buy=yes (1.0)**

R3: **IF** stock=penny
THEN **buy=yes (-0.7)**

Inputs: dividends=yes (cf 0.9)
mgnt=good (cf 0.7)
earnings=positive (cf 0.5)
contract=large (cf 0.8)
stock=penny (cf 1.0)

- `Fire R1: buy=yes = min (0.9, 0.7, 0.5) * 0.6 = 0.3`
- `Fire R2: buy=yes = 0.8 * 1.0 = 0.8`
- `Fire RX: buy=yes = 0.3 + 0.8 * (1.0 - 0.3) = 0.86`
- `Fire R3: buy=yes = 1.0 * -0.7 = -0.7`
- `Fire RX: buy=yes = (-0.7 + 0.86) / (1.0 - 0.7) = 0.53`
- `Therefore, final recommendation: buy=yes (0.53)`

Certainty Factor

Certainty Factor Summary

- **Certainty factors theory provides a practical alternative to probability calculation.**

- **Certainty Factor approach mimics the thinking process of a human expert.**

- **Certainty Factor approach provides better intuitive explanations to users.**

END OF APPENDICES