Computer Assignment 2

CPE 261456 (Introduction to Computational Intelligence)

โดย

นายพีรณัฐ ธารทะเลทอง

รหัสนักศึกษา 550610530

เสนอ

ผศ.ดร. ศันสนีย์ เอื้อพันธ์วิริยะกุล

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่

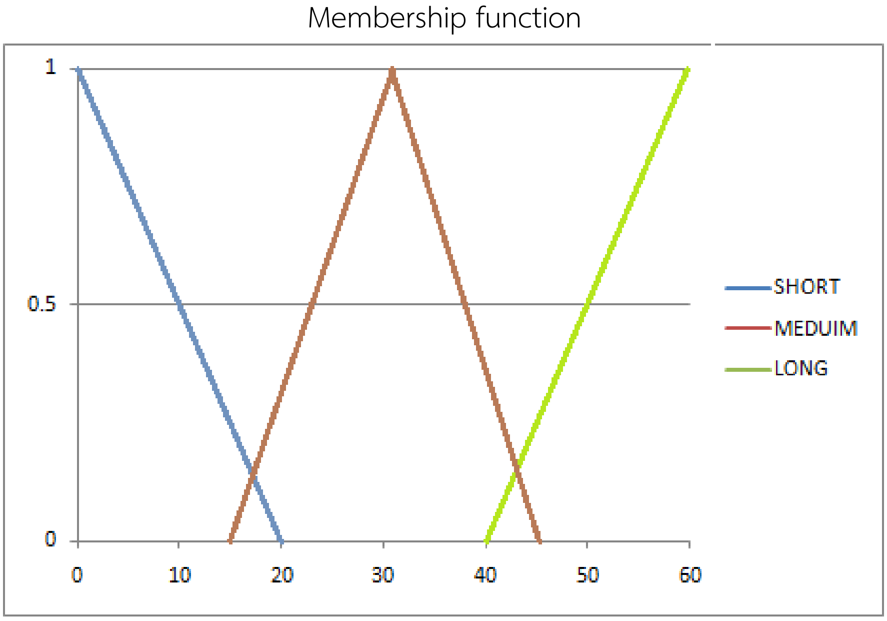
Steak Fuzzy Logic Simulator

เป็นระบบควบคุมความแรงของไฟในการย่างสเต็ก

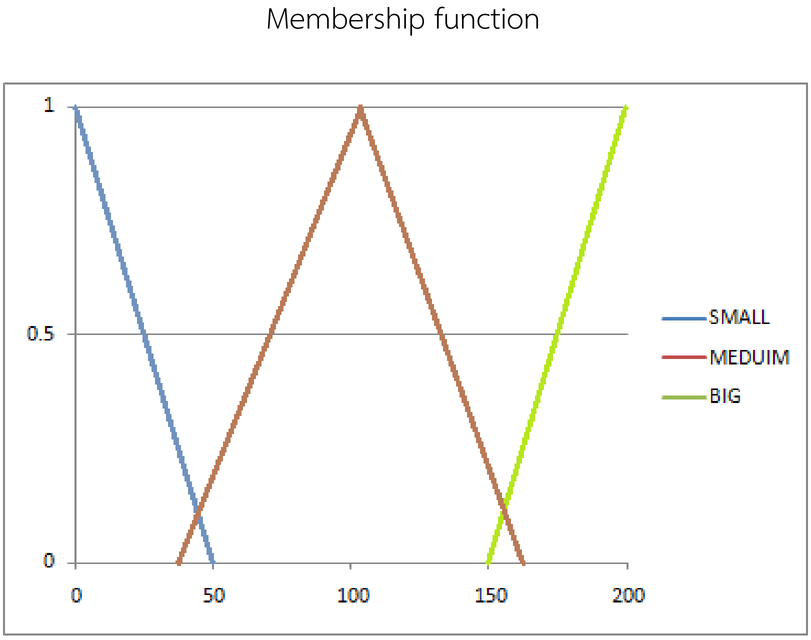
ลักษณะการทำงานของระบบ รวมถึง rules ที่ใช้

Input

1. เวลา (Time) มี 3 ระดับคือ SHORT MEDIUM LONG

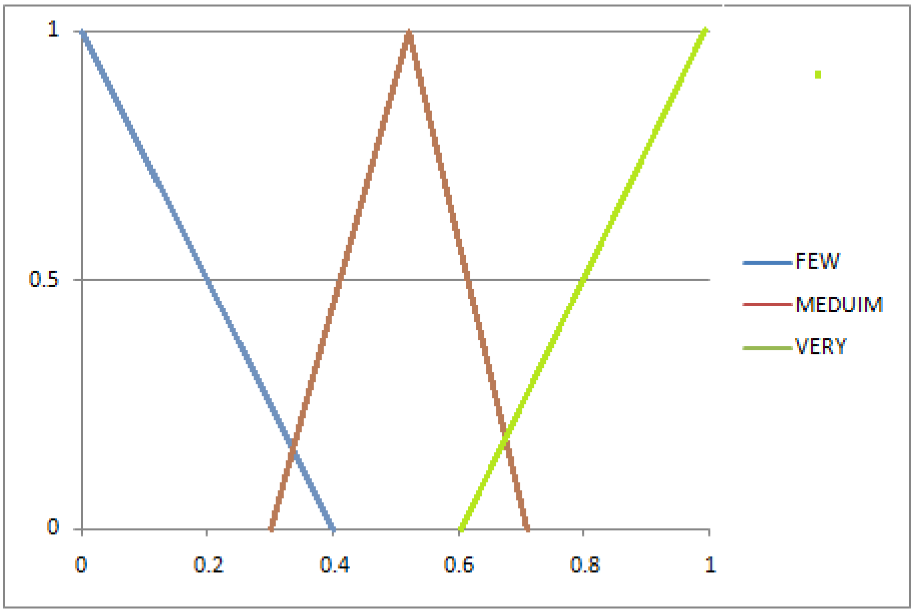


2. ขนาด (Size) มี 3 ระดับคือ SMALL MEDIUM BIG



3. ความแข็งของเนื้อ (Hardness) มี 3 ระดับคือFEW MEDIUM VERY

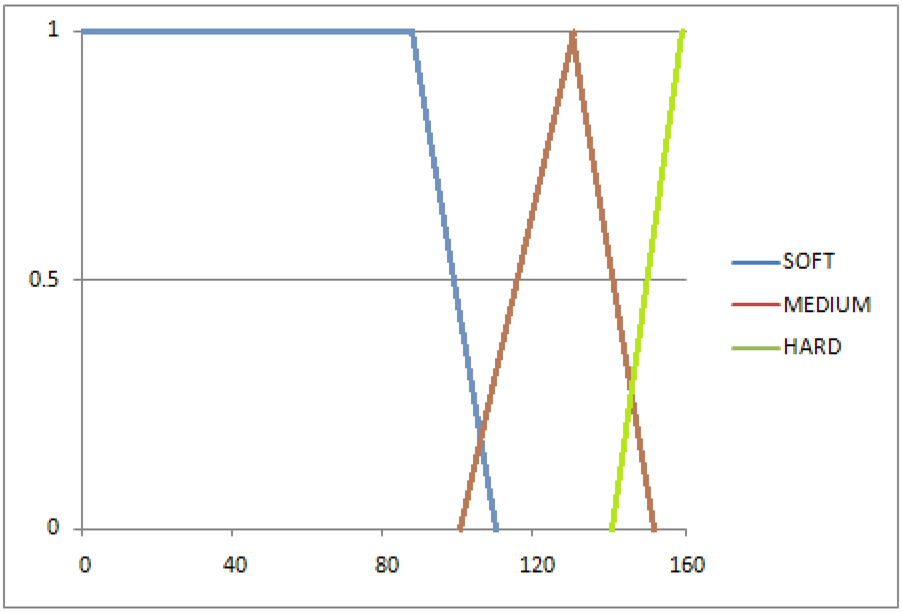
Membership function



Output

ความแรงของไฟที่จะใช้ย่างสเต็ก (Fire) มี 3 ระดับคือ SOFT MEDIUM HARD

Membership function



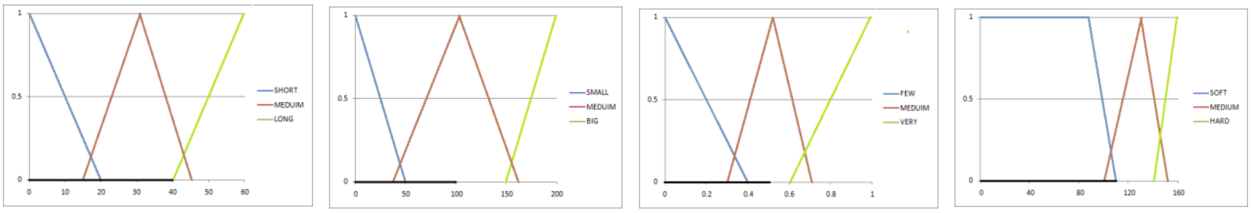
Rules

1. If Time is **SHORT** and Size is **SMALL** and Hardness is **FEW** then Fire is **SOFT**
2. If Time is **SHORT** and Size is **SMALL** and Hardness is **MEDIUM** then Fire is **SOFT**
3. If Time is **SHORT** and Size is **SMALL** and Hardness is **BIG** then Fire is **SOFT**
4. If Time is **SHORT** and Size is **MEDIUM** and Hardness is **FEW** then Fire is **HARD**
5. If Time is **SHORT** and Size is **MEDIUM** and Hardness is **MEDIUM** then Fire is **HARD**
6. If Time is **SHORT** and Size is **MEDIUM** and Hardness is **BIG** then Fire is **HARD**
7. If Time is **SHORT** and Size is **BIG** and Hardness is **FEW** then Fire is **HARD**
8. If Time is **SHORT** and Size is **BIG** and Hardness is **MEDIUM** then Fire is **HARD**
9. If Time is **SHORT** and Size is **BIG** and Hardness is **BIG** then Fire is **HARD**
10. If Time is **MEDIUM** and Size is **SMALL** and Hardness is **FEW** then Fire is **SOFT**
11. If Time is **MEDIUM** and Size is **SMALL** and Hardness is **MEDIUM** then Fire is **SOFT**
12. If Time is **MEDIUM** and Size is **SMALL** and Hardness is **BIG** then Fire is **SOFT**
13. If Time is **MEDIUM** and Size is **MEDIUM** and Hardness is **FEW** then Fire is **MEDIUM**
14. If Time is **MEDIUM** and Size is **MEDIUM** and Hardness is **MEDIUM** then Fire is **MEDIUM**
15. If Time is **MEDIUM** and Size is **MEDIUM** and Hardness is **BIG** then Fire is **MEDIUM**
16. If Time is **MEDIUM** and Size is **BIG** and Hardness is **FEW** then Fire is **HARD**
17. If Time is **MEDIUM** and Size is **BIG** and Hardness is **MEDIUM** then Fire is **HARD**
18. If Time is **MEDIUM** and Size is **BIG** and Hardness is **BIG** then Fire is **HARD**
19. If Time is **LONG** and Size is **SMALL** and Hardness is **FEW** then Fire is **SOFT**
20. If Time is **LONG** and Size is **SMALL** and Hardness is **MEDIUM** then Fire is **SOFT**
21. If Time is **LONG** and Size is **SMALL** and Hardness is **BIG** then Fire is **SOFT**
22. If Time is **LONG** and Size is **MEDIUM** and Hardness is **FEW** then Fire is **SOFT**
23. If Time is **LONG** and Size is **MEDIUM** and Hardness is **MEDIUM** then Fire is **SOFT**
24. If Time is **LONG** and Size is **MEDIUM** and Hardness is **BIG** then Fire is **SOFT**
25. If Time is **LONG** and Size is **BIG** and Hardness is **FEW** then Fire is **HARD**
26. If Time is **LONG** and Size is **BIG** and Hardness is **MEDIUM** then Fire is **HARD**
27. If Time is **LONG** and Size is **BIG** and Hardness is **BIG** then Fire is **HARD**

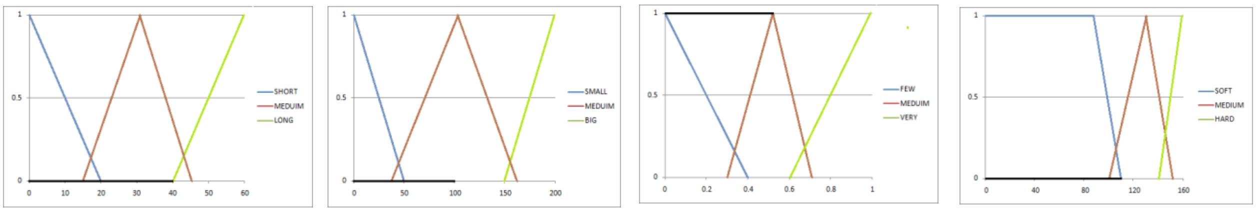
Simulation ของระบบ

สมมติให้ Input Time = 40, Size = 100, Hardness = 0.5

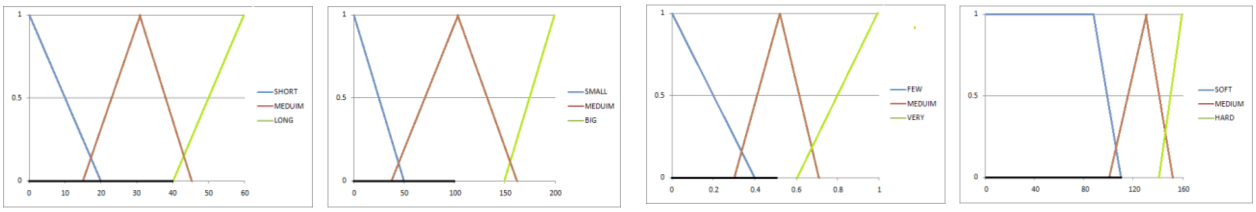
1. If Time is **SHORT** and Size is **SMALL** and Hardness is **FEW** then Fire is **SOFT**



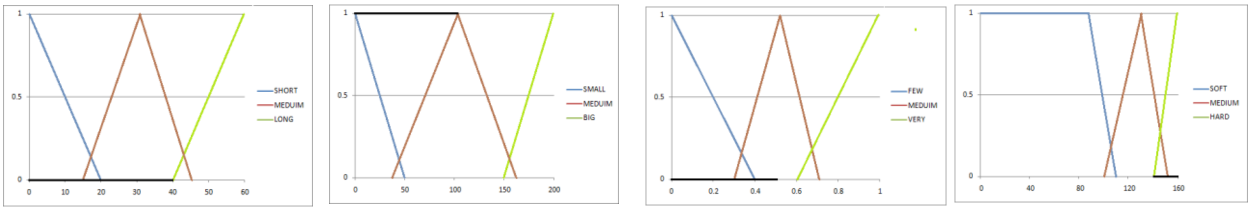
2. If Time is **SHORT** and Size is **SMALL** and Hardness is **MEDIUM** then Fire is **SOFT**



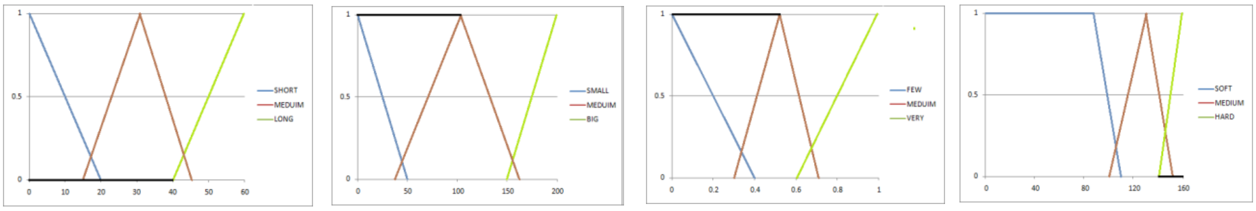
3. If Time is **SHORT** and Size is **SMALL** and Hardness is **BIG** then Fire is **SOFT**



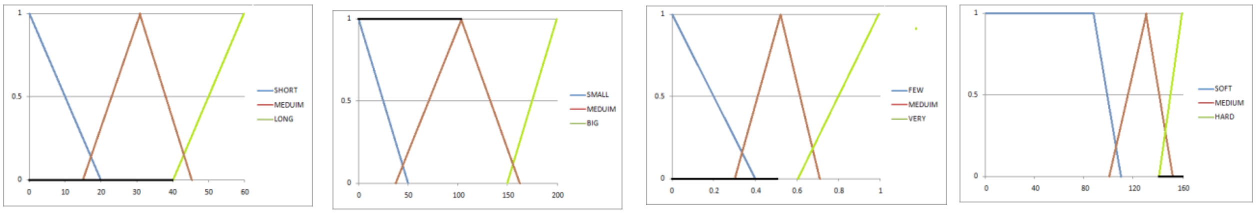
4. If Time is **SHORT** and Size is **MEDIUM** and Hardness is **FEW** then Fire is **HARD**



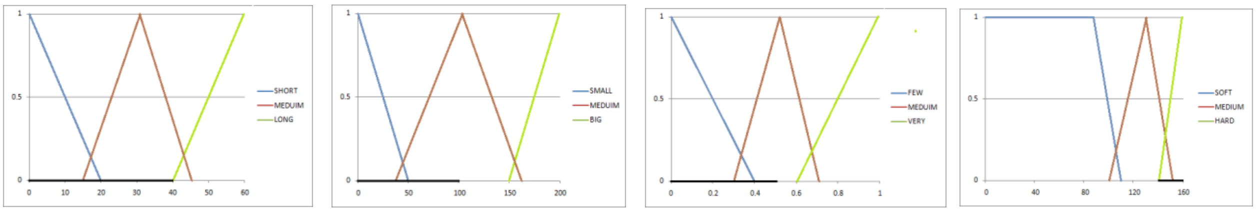
5. If Time is **SHORT** and Size is **MEDIUM** and Hardness is **MEDIUM** then Fire is **HARD**



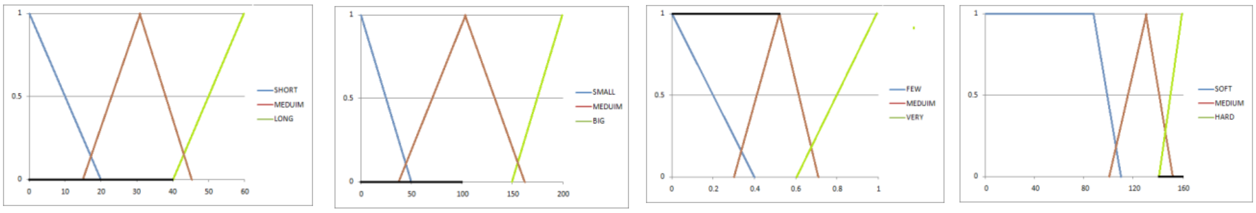
6. If Time is **SHORT** and Size is **MEDIUM** and Hardness is **BIG** then Fire is **HARD**



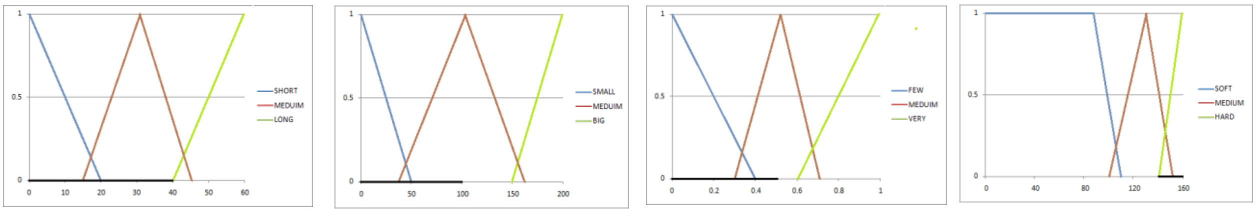
7. If Time is **SHORT** and Size is **BIG** and Hardness is **FEW** then Fire is **HARD**



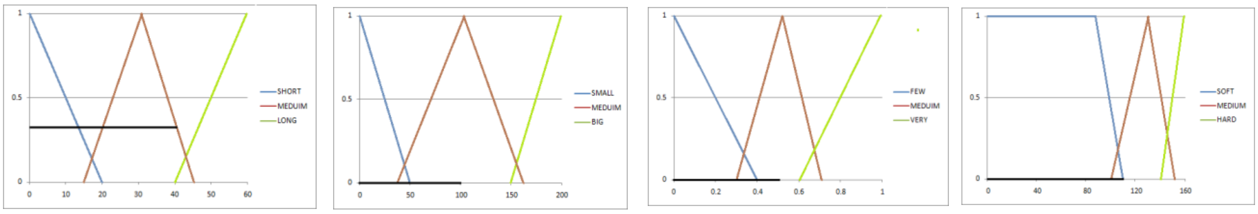
8. If Time is **SHORT** and Size is **BIG** and Hardness is **MEDIUM** then Fire is **HARD**



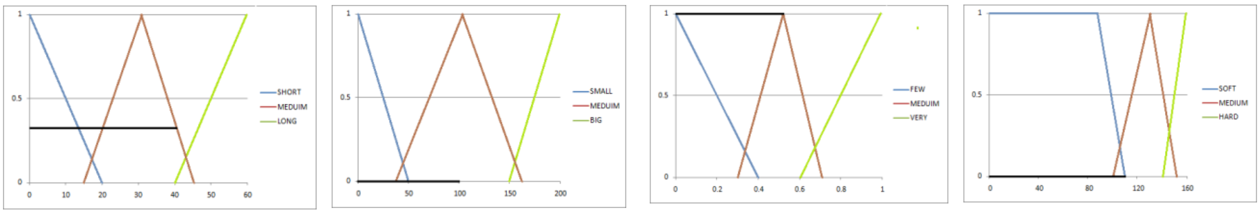
9. If Time is **SHORT** and Size is **BIG** and Hardness is **BIG** then Fire is **HARD**



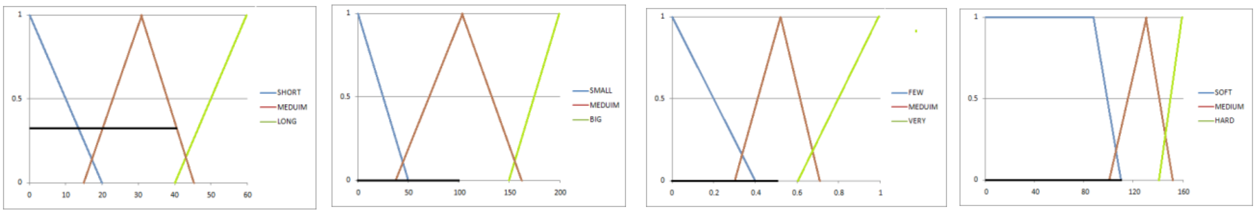
10. If Time is **MEDIUM** and Size is **SMALL** and Hardness is **FEW** then Fire is **SOFT**



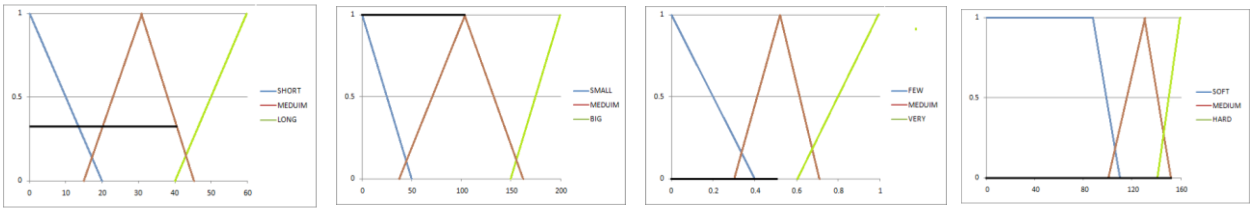
11. If Time is **MEDIUM** and Size is **SMALL** and Hardness is **MEDIUM** then Fire is **SOFT**



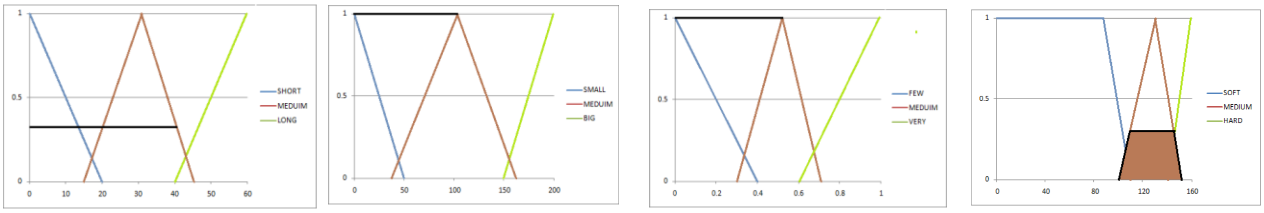
12. If Time is **MEDIUM** and Size is **SMALL** and Hardness is **BIG** then Fire is **SOFT**



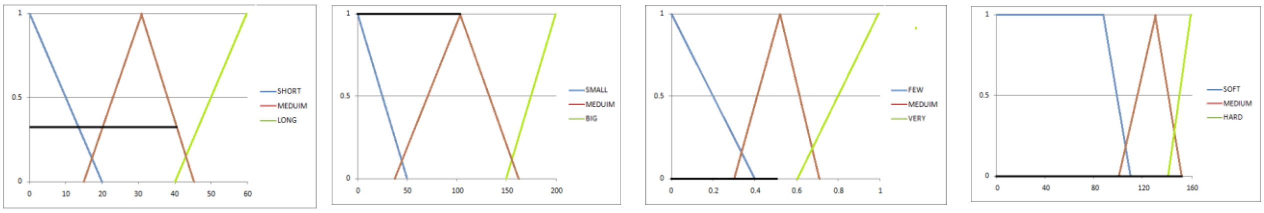
13. If Time is **MEDIUM** and Size is **MEDIUM** and Hardness is **FEW** then Fire is **MEDIUM**



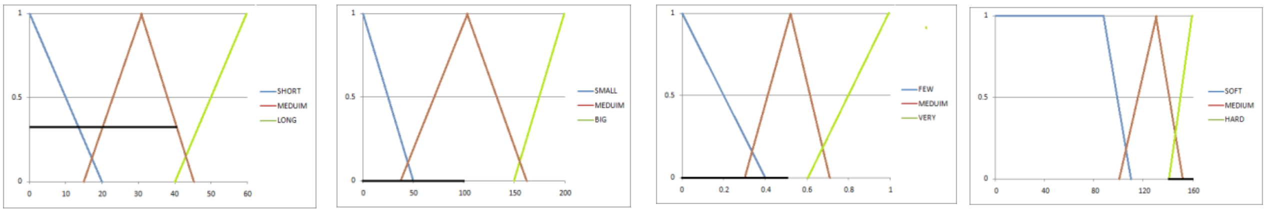
14. If Time is **MEDIUM** and Size is **MEDIUM** and Hardness is **MEDIUM** then Fire is **MEDIUM**



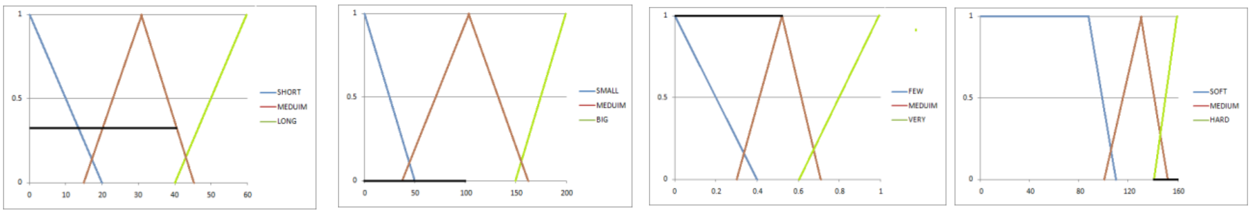
15. If Time is **MEDIUM** and Size is **MEDIUM** and Hardness is **BIG** then Fire is **MEDIUM**



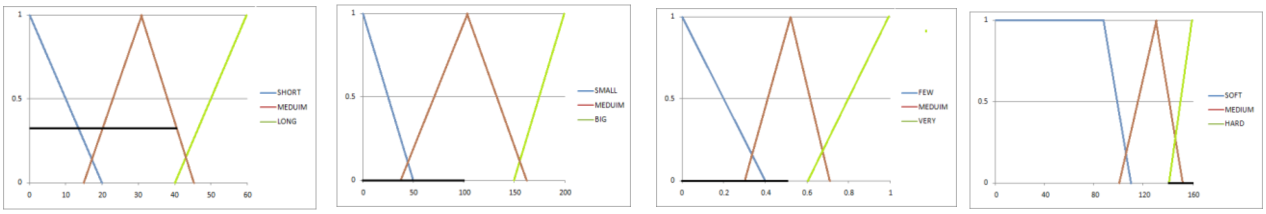
16. If Time is **MEDIUM** and Size is **BIG** and Hardness is **FEW** then Fire is **HARD**



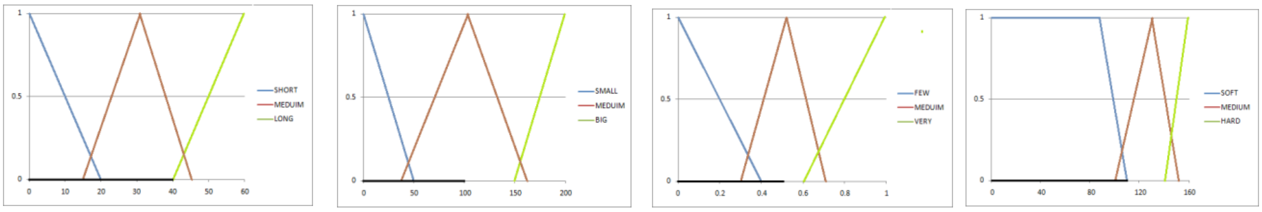
17. If Time is **MEDIUM** and Size is **BIG** and Hardness is **MEDIUM** then Fire is **HARD**



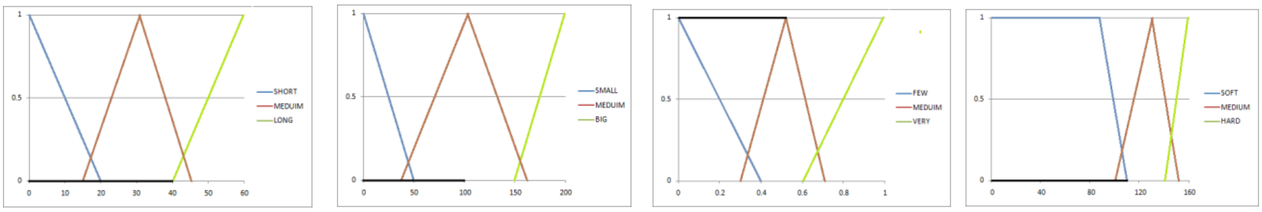
18. If Time is **MEDIUM** and Size is **BIG** and Hardness is **BIG** then Fire is **HARD**



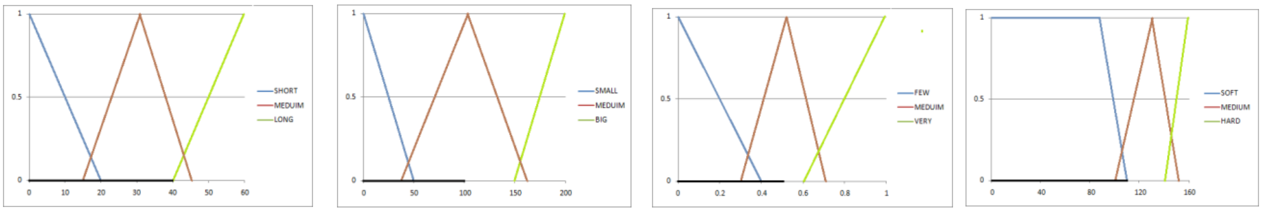
19. If Time is **LONG** and Size is **SMALL** and Hardness is **FEW** then Fire is **SOFT**



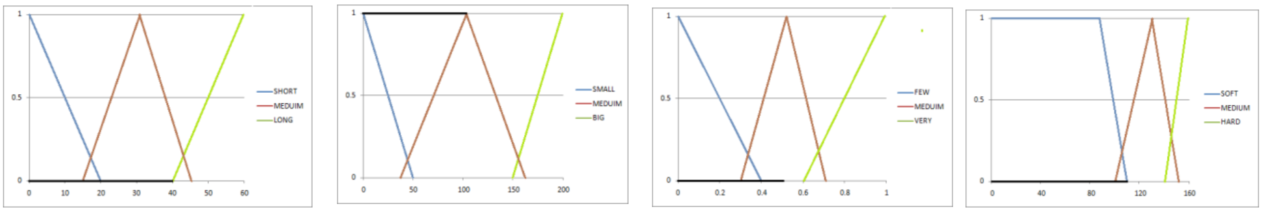
20. If Time is **LONG** and Size is **SMALL** and Hardness is **MEDIUM** then Fire is **SOFT**



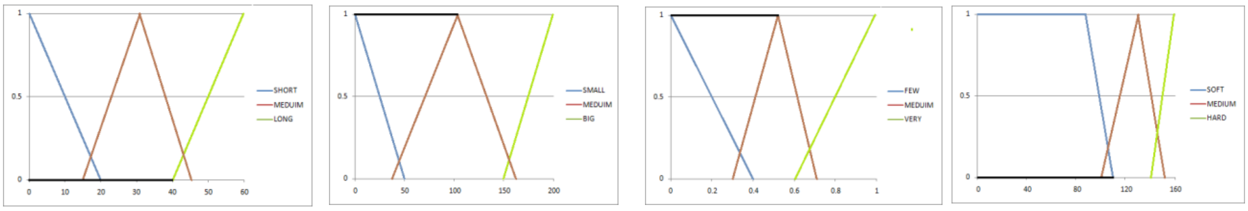
21. If Time is **LONG** and Size is **SMALL** and Hardness is **BIG** then Fire is **SOFT**



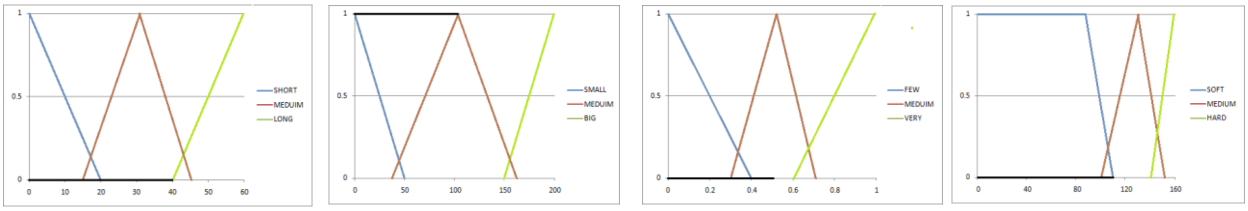
22. If Time is **LONG** and Size is **MEDIUM** and Hardness is **FEW** then Fire is **SOFT**



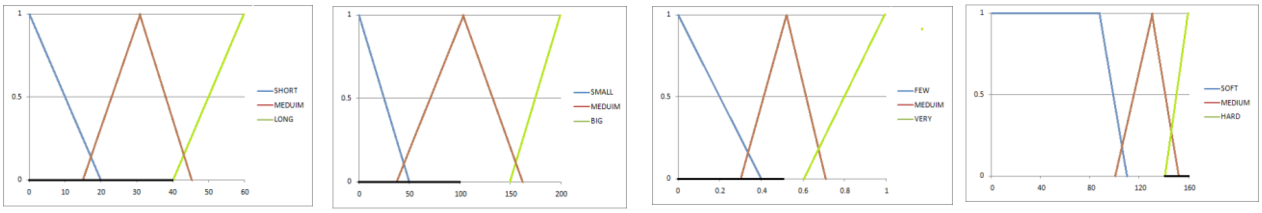
23. If Time is **LONG** and Size is **MEDIUM** and Hardness is **MEDIUM** then Fire is **SOFT**



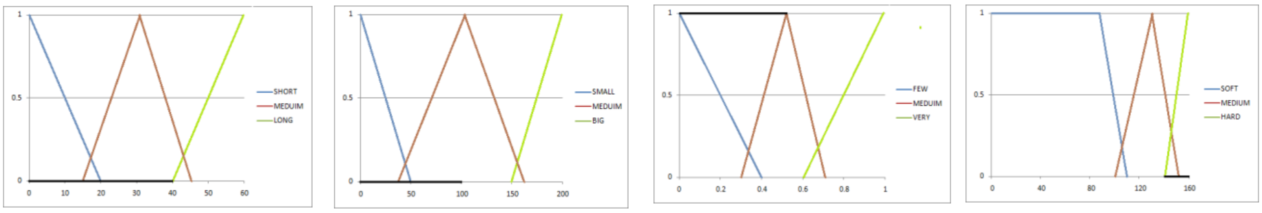
24. If Time is **LONG** and Size is **MEDIUM** and Hardness is **BIG** then Fire is **SOFT**



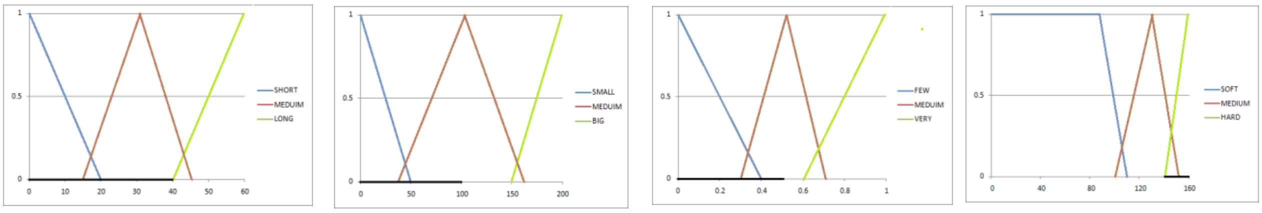
25. If Time is **LONG** and Size is **BIG** and Hardness is **FEW** then Fire is **HARD**



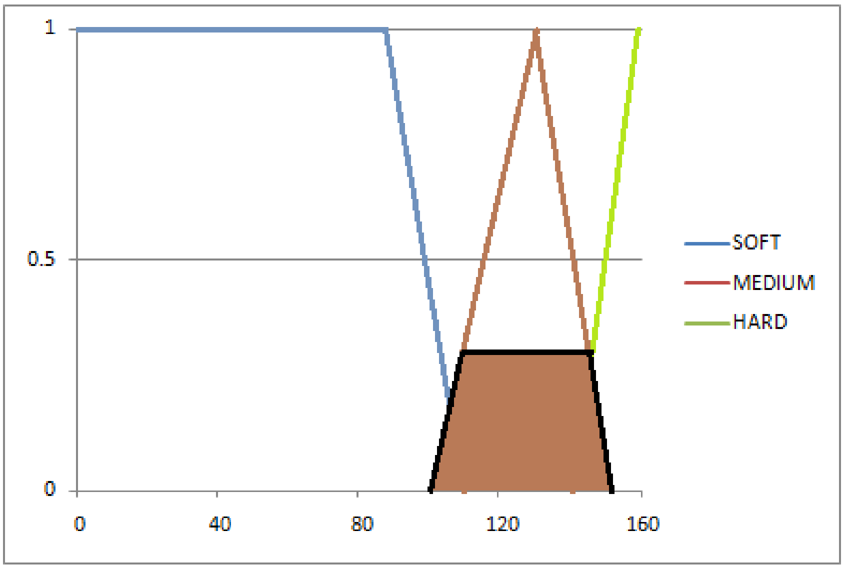
26. If Time is **LONG** and Size is **BIG** and Hardness is **MEDIUM** then Fire is **HARD**



27. If Time is **LONG** and Size is **BIG** and Hardness is **BIG** then Fire is **HARD**



Output ที่ได้



จากนั้นทำ defuzzification โดยการหา centroid

ซึ่งได้ค่าความแรงของไฟ = 124.99999999999984 °c

การทดลอง

ให้ Input Time = 20, Size = 190, Hardness = 0.9

Output fire = 151.81632653061232 °c

ให้ Input Time = 40, Size = 100, Hardness = 0.5

Output fire = 124.99999999999984 °c

ให้ Input Time = 41, Size = 150, Hardness = 0.5

Output fire = 95.14259453781511 °c

วิเคราะห์ผลการทดลอง

จากการทดลองโดยการปรับค่า เวลาที่จะใช้ (Time), ขนาดของเนื้อ(Size), ความแข็งของเนื้อ (Hardness) คำตอบที่ได้นั้นเป็นไปตามกฎที่ได้ตั้งไว้ข้างต้น ซึ่งกฎนี้สามารถปรับเปลี่ยนได้ตามความเหมาะสม

Code (https://github.com/porpeeranut/Computational_Intelligence_Assignment2)

```
//          Main.java

import java.util.ArrayList;
import java.util.HashMap;


public class Main {


        public static void main(String[] args) {
                HashMap<Enum, Graph> mfTime = new HashMap<Enum, Graph>();
                mfTime.put(TimeLevel.SHORT, new Graph(0, 20));
                mfTime.put(TimeLevel.MEDIUM, new Graph(15, 45));
                mfTime.put(TimeLevel.LONG, new Graph(40, 60));

                HashMap<Enum, Graph> mfSize = new HashMap<Enum, Graph>();
                mfSize.put(SizeLevel.SMALL, new Graph(0, 50));
                mfSize.put(SizeLevel.MEDIUM, new Graph(40, 160));
                mfSize.put(SizeLevel.BIG, new Graph(150, 200));

                HashMap<Enum, Graph> mfHardness = new HashMap<Enum, Graph>();
                mfHardness.put(HardnessLevel.FEW, new Graph(0, 0.4));
                mfHardness.put(HardnessLevel.MEDIUM, new Graph(0.3, 0.7));
                mfHardness.put(HardnessLevel.VERY, new Graph(0.6, 1));

                HashMap<Enum, Graph> mfFire = new HashMap<Enum, Graph>();
                mfFire.put(FireLevel.SOFT, new Graph(90, 110));
                mfFire.put(FireLevel.MEDIUM, new Graph(100, 150));
                mfFire.put(FireLevel.HARD, new Graph(140, 160));



                HashMap<Fuzzy, Data> input = new HashMap<Fuzzy, Data>() ;
                input.put(Fuzzy.TIME, new Data(mfTime, new Range(0, 60, 1)));
                input.put(Fuzzy.SIZE, new Data(mfSize, new Range(0, 200, 1)));
                input.put(Fuzzy.HARDNESS, new Data(mfHardness, new Range(0, 1, 0.05)));

                HashMap<Fuzzy, Data> output = new HashMap<Fuzzy, Data>();
                output.put(Fuzzy.FIRE, new Data(mfFire, new Range(0, 160, 1)));

                ArrayList<Rule> rules = new ArrayList<Rule>();
                /*addRule(TimeLevel.SHORT, SizeLevel.SMALL, HardnessLevel.FEW, FireLevel.SOFT, rules);
```

```
addRule(TimeLevel.SHORT, SizeLevel.SMALL, HardnessLevel.MEDIUM, FireLevel.MEDIUM, rules);
addRule(TimeLevel.SHORT, SizeLevel.SMALL, HardnessLevel.VERY, FireLevel.MEDIUM, rules);
addRule(TimeLevel.SHORT, SizeLevel.MEDIUM, HardnessLevel.FEW, FireLevel.MEDIUM, rules);
addRule(TimeLevel.SHORT, SizeLevel.MEDIUM, HardnessLevel.MEDIUM, FireLevel.HARD, rules);
addRule(TimeLevel.SHORT, SizeLevel.MEDIUM, HardnessLevel.VERY, FireLevel.HARD, rules);
addRule(TimeLevel.SHORT, SizeLevel.BIG, HardnessLevel.FEW, FireLevel.MEDIUM, rules);
addRule(TimeLevel.SHORT, SizeLevel.BIG, HardnessLevel.MEDIUM, FireLevel.HARD, rules);
addRule(TimeLevel.SHORT, SizeLevel.BIG, HardnessLevel.VERY, FireLevel.HARD, rules);

addRule(TimeLevel.MEDIUM, SizeLevel.SMALL, HardnessLevel.FEW, FireLevel.SOFT, rules);
addRule(TimeLevel.MEDIUM, SizeLevel.SMALL, HardnessLevel.MEDIUM, FireLevel.SOFT, rules);
addRule(TimeLevel.MEDIUM, SizeLevel.SMALL, HardnessLevel.VERY, FireLevel.MEDIUM, rules);
addRule(TimeLevel.MEDIUM, SizeLevel.MEDIUM, HardnessLevel.FEW, FireLevel.SOFT, rules);
addRule(TimeLevel.MEDIUM, SizeLevel.MEDIUM, HardnessLevel.MEDIUM, FireLevel.MEDIUM, rules);
addRule(TimeLevel.MEDIUM, SizeLevel.MEDIUM, HardnessLevel.VERY, FireLevel.HARD, rules);
addRule(TimeLevel.MEDIUM, SizeLevel.BIG, HardnessLevel.FEW, FireLevel.MEDIUM, rules);
addRule(TimeLevel.MEDIUM, SizeLevel.BIG, HardnessLevel.MEDIUM, FireLevel.HARD, rules);
addRule(TimeLevel.MEDIUM, SizeLevel.BIG, HardnessLevel.VERY, FireLevel.HARD, rules);

addRule(TimeLevel.LONG, SizeLevel.SMALL, HardnessLevel.FEW, FireLevel.SOFT, rules);
addRule(TimeLevel.LONG, SizeLevel.SMALL, HardnessLevel.MEDIUM, FireLevel.SOFT, rules);
addRule(TimeLevel.LONG, SizeLevel.SMALL, HardnessLevel.VERY, FireLevel.MEDIUM, rules);
addRule(TimeLevel.LONG, SizeLevel.MEDIUM, HardnessLevel.FEW, FireLevel.SOFT, rules);
addRule(TimeLevel.LONG, SizeLevel.MEDIUM, HardnessLevel.MEDIUM, FireLevel.SOFT, rules);
addRule(TimeLevel.LONG, SizeLevel.MEDIUM, HardnessLevel.VERY, FireLevel.MEDIUM, rules);
addRule(TimeLevel.LONG, SizeLevel.BIG, HardnessLevel.FEW, FireLevel.MEDIUM, rules);
addRule(TimeLevel.LONG, SizeLevel.BIG, HardnessLevel.MEDIUM, FireLevel.MEDIUM, rules);
addRule(TimeLevel.LONG, SizeLevel.BIG, HardnessLevel.VERY, FireLevel.HARD, rules);*/

addRule(TimeLevel.SHORT, SizeLevel.SMALL, HardnessLevel.FEW, FireLevel.SOFT, rules);
addRule(TimeLevel.SHORT, SizeLevel.SMALL, HardnessLevel.MEDIUM, FireLevel.SOFT, rules);
addRule(TimeLevel.SHORT, SizeLevel.SMALL, HardnessLevel.VERY, FireLevel.SOFT, rules);
addRule(TimeLevel.SHORT, SizeLevel.MEDIUM, HardnessLevel.FEW, FireLevel.HARD, rules);
addRule(TimeLevel.SHORT, SizeLevel.MEDIUM, HardnessLevel.MEDIUM, FireLevel.HARD, rules);
addRule(TimeLevel.SHORT, SizeLevel.MEDIUM, HardnessLevel.VERY, FireLevel.HARD, rules);
addRule(TimeLevel.SHORT, SizeLevel.BIG, HardnessLevel.FEW, FireLevel.HARD, rules);
addRule(TimeLevel.SHORT, SizeLevel.BIG, HardnessLevel.MEDIUM, FireLevel.HARD, rules);
addRule(TimeLevel.SHORT, SizeLevel.BIG, HardnessLevel.VERY, FireLevel.HARD, rules);

addRule(TimeLevel.MEDIUM, SizeLevel.SMALL, HardnessLevel.FEW, FireLevel.SOFT, rules);
```

```
        addRule(TimeLevel.MEDIUM, SizeLevel.SMALL, HardnessLevel.MEDIUM, FireLevel.SOFT, rules);
        addRule(TimeLevel.MEDIUM, SizeLevel.SMALL, HardnessLevel.VERY, FireLevel.SOFT, rules);
        addRule(TimeLevel.MEDIUM, SizeLevel.MEDIUM, HardnessLevel.FEW, FireLevel.MEDIUM, rules);
        addRule(TimeLevel.MEDIUM, SizeLevel.MEDIUM, HardnessLevel.MEDIUM, FireLevel.MEDIUM, rules);
        addRule(TimeLevel.MEDIUM, SizeLevel.MEDIUM, HardnessLevel.VERY, FireLevel.MEDIUM, rules);
        addRule(TimeLevel.MEDIUM, SizeLevel.BIG, HardnessLevel.FEW, FireLevel.HARD, rules);
        addRule(TimeLevel.MEDIUM, SizeLevel.BIG, HardnessLevel.MEDIUM, FireLevel.HARD, rules);
        addRule(TimeLevel.MEDIUM, SizeLevel.BIG, HardnessLevel.VERY, FireLevel.HARD, rules);

        addRule(TimeLevel.LONG, SizeLevel.SMALL, HardnessLevel.FEW, FireLevel.SOFT, rules);
        addRule(TimeLevel.LONG, SizeLevel.SMALL, HardnessLevel.MEDIUM, FireLevel.SOFT, rules);
        addRule(TimeLevel.LONG, SizeLevel.SMALL, HardnessLevel.VERY, FireLevel.SOFT, rules);
        addRule(TimeLevel.LONG, SizeLevel.MEDIUM, HardnessLevel.FEW, FireLevel.SOFT, rules);
        addRule(TimeLevel.LONG, SizeLevel.MEDIUM, HardnessLevel.MEDIUM, FireLevel.SOFT, rules);
        addRule(TimeLevel.LONG, SizeLevel.MEDIUM, HardnessLevel.VERY, FireLevel.SOFT, rules);
        addRule(TimeLevel.LONG, SizeLevel.BIG, HardnessLevel.FEW, FireLevel.HARD, rules);
        addRule(TimeLevel.LONG, SizeLevel.BIG, HardnessLevel.MEDIUM, FireLevel.HARD, rules);
        addRule(TimeLevel.LONG, SizeLevel.BIG, HardnessLevel.VERY, FireLevel.HARD, rules);



        SteakFuzzyLogic steakFuzzy = new SteakFuzzyLogic(input, output, rules);
        //System.out.println(steakFuzzy.defuz(20, 190, 0.9)+" °c\n"); //> 150
        //System.out.println(steakFuzzy.defuz(40, 100, 0.5)+" °c\n"); //>124
        System.out.println(steakFuzzy.defuz(41, 150, 0.9)+" °c\n"); //>95
    }

    static void addRule(TimeLevel tLevel, SizeLevel sLevel, HardnessLevel hLevel, FireLevel fLevel, ArrayList<Rule>
rules) {
        ArrayList<RuleData> ifRule = new ArrayList<RuleData>();
        ifRule.add(new RuleData(Fuzzy.TIME, tLevel));
        ifRule.add(new RuleData(Fuzzy.SIZE, sLevel));
        ifRule.add(new RuleData(Fuzzy.HARDNESS, hLevel));

        RuleData thenRule = new RuleData(Fuzzy.FIRE, fLevel);
        rules.add(new Rule(ifRule, thenRule));
    }
}
```

```
//    SteakFuzzyLogic.java
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map.Entry;

public class SteakFuzzyLogic {

    HashMap<Fuzzy, Data> input;
    HashMap<Fuzzy, Data> output;
    ArrayList<Rule> rules;
    double maxFireSoft = 0;
    double maxFireMed = 0;
    double maxFireHard = 0;

    public SteakFuzzyLogic(HashMap<Fuzzy, Data> input, HashMap<Fuzzy, Data> output, ArrayList<Rule> rules) {
        this.input = input;
        this.output = output;
        this.rules = rules;
    }

    double defuz(double time, double size, double hardness) {
        maxFireSoft = 0;
        maxFireMed = 0;
        maxFireHard = 0;
        int r = 1;
        boolean debug = false;
        for(Rule rule : rules) {
            double minInRule = 1;
            if (debug) {
                System.out.print ("#"+r+" ");
            }
            for(RuleData ruleData : rule.ifRule) {
                //    ifRule.add(new RuleData(Fuzzy.TIME, TimeLevel.SHORT));
                //    input.put(Fuzzy.TIME, new Data(mfTime, new Range(0, 60, 1)));
                double tmp = 0;
                if (ruleData.fuzzy == Fuzzy.TIME) {
                    if (ruleData.level == TimeLevel.SHORT) {
                        tmp =
input.get(Fuzzy.TIME).mf.get(TimeLevel.SHORT).getFuzzyValDown(time);
                        if (debug) {
                            System.out.print ("SHORT->"+tmp);
                        }
                    } else if (ruleData.level == TimeLevel.MEDIUM) {
                        tmp =
input.get(Fuzzy.TIME).mf.get(TimeLevel.MEDIUM).getFuzzyValTriangle(time);
                        if (debug) {
                            System.out.print ("MED->"+tmp);
                        }
                    } else if (ruleData.level == TimeLevel.LONG) {
                        tmp =
input.get(Fuzzy.TIME).mf.get(TimeLevel.LONG).getFuzzyValUp(time);
                        if (debug) {
                            System.out.print ("LONG->"+tmp);
                        }
                    }
                } else if (ruleData.fuzzy == Fuzzy.SIZE) {
                    if (ruleData.level == SizeLevel.SMALL) {
                        tmp =
input.get(Fuzzy.SIZE).mf.get(SizeLevel.SMALL).getFuzzyValDown(size);
                        if (debug) {
                            System.out.print (" SMALL->"+tmp);
                        }
                    } else if (ruleData.level == SizeLevel.MEDIUM) {
                        tmp =
input.get(Fuzzy.SIZE).mf.get(SizeLevel.MEDIUM).getFuzzyValTriangle(size);
                        if (debug) {
                            System.out.print (" MED->"+tmp);
                        }
                    } else if (ruleData.level == SizeLevel.BIG) {
                        tmp =
input.get(Fuzzy.SIZE).mf.get(SizeLevel.BIG).getFuzzyValUp(size);
                        if (debug) {
                            System.out.print (" BIG->"+tmp);
                        }
                    }
                } else if (ruleData.fuzzy == Fuzzy.HARDNESS) {
                    if (ruleData.level == HardnessLevel.FEW) {
                        tmp =
input.get(Fuzzy.HARDNESS).mf.get(HardnessLevel.FEW).getFuzzyValDown(hardness);
                        if (debug) {
                            System.out.print (" FEW->"+tmp);
                        }
                    } else if (ruleData.level == HardnessLevel.MEDIUM) {
                        tmp =
input.get(Fuzzy.HARDNESS).mf.get(HardnessLevel.MEDIUM).getFuzzyValTriangle(hardness);
                        if (debug) {
                            System.out.print (" MED->"+tmp);
                        }
                    } else if (ruleData.level == HardnessLevel.VERY) {
                        tmp =
input.get(Fuzzy.HARDNESS).mf.get(HardnessLevel.VERY).getFuzzyValUp(hardness);
                        if (debug) {
                            System.out.print (" VERY->"+tmp);
                        }
                    }
                }
                /*for (int i = 0;i < tmp * 100.0;i++) {
                    System.out.print("*");
                }
                System.out.println("*");*/


                if (minInRule > tmp && tmp >= 0)
                    minInRule = tmp;
            }

            if (rule.thenRule.level == FireLevel.SOFT) {
                if (debug) {
                    System.out.println(" soft = "+minInRule);
                }
                if (maxFireSoft < minInRule)
                    maxFireSoft = minInRule;
            } else if (rule.thenRule.level == FireLevel.MEDIUM) {
                if (debug) {
                    System.out.println(" med = "+minInRule);
                }
                if (maxFireMed < minInRule)
                    maxFireMed = minInRule;
            } else if (rule.thenRule.level == FireLevel.HARD) {
                if (debug) {
                        System.out.println(" heig = "+minInRule);
                }
                if (maxFireHard < minInRule)
                        maxFireHard = minInRule;
        }
        r++;
}
if (debug) {
        System.out.println(maxFireSoft);
        System.out.println(maxFireMed);
        System.out.println(maxFireHard);
}

double start = output.get(Fuzzy.FIRE).range.start;
double end = output.get(Fuzzy.FIRE).range.end;
double step = output.get(Fuzzy.FIRE).range.step;
double startSoft = output.get(Fuzzy.FIRE).mf.get(FireLevel.SOFT).x_start;
double endSoft = output.get(Fuzzy.FIRE).mf.get(FireLevel.SOFT).x_end;
double startMed = output.get(Fuzzy.FIRE).mf.get(FireLevel.MEDIUM).x_start;
double endMed = output.get(Fuzzy.FIRE).mf.get(FireLevel.MEDIUM).x_end;
double startHard = output.get(Fuzzy.FIRE).mf.get(FireLevel.HARD).x_start;
double endHard = output.get(Fuzzy.FIRE).mf.get(FireLevel.HARD).x_end;

double sum1 = 0;
double sum2 = 0;
for (double x = start;x <= end;x += step) {
        double ySoft = 0;
        double yMed = 0;
        double yHard = 0;
        if (x <= endSoft) {
                ySoft = output.get(Fuzzy.FIRE).mf.get(FireLevel.SOFT).getFuzzyValDown(x);
                if (ySoft > maxFireSoft)
                        ySoft = maxFireSoft;
        }
        if (x >= startMed && x <= endMed) {
                yMed = output.get(Fuzzy.FIRE).mf.get(FireLevel.MEDIUM).getFuzzyValTriangle(x);
                if (yMed > maxFireMed)
                        yMed = maxFireMed;
        }
        if (x >= startHard) {
                yHard = output.get(Fuzzy.FIRE).mf.get(FireLevel.HARD).getFuzzyValUp(x);
                if (yHard > maxFireHard)
                    yHard = maxFireHard;
            }

            double max = max(ySoft, yMed, yHard);
            /*for (int i = 0;i < max * 100.0;i++) {
                System.out.print(" ");
            }
            System.out.println("*");*/
            sum1 += max*x;
            sum2 += max;
        }
        if (sum2 == 0)
            sum2 = 1;
        return sum1/sum2;
    }

    double max(double ySoft, double yMed, double yHard) {
        if (ySoft > yMed) {
            if (ySoft > yHard)
                return ySoft;
            else
                return yHard;
        } else {
            if (yMed > yHard)
                return yMed;
            else
                return yHard;
        }
    }
}
```

```
//          Graph.java
import java.util.ArrayList;
import java.util.Map.Entry;
public class Graph {
        public double x_start;
        public double x_end;
        public Graph(double x_start, double x_end) {
                this.x_start = x_start;
                this.x_end = x_end;
        }
        double getFuzzyValDown(double x) {
                double m = (1/(x_start-x_end));
                double c = -m*x_end;
                double y = m*x + c;
                return (y > 1) ? 1 : (y < 0) ? 0 : y;
        }
        double getFuzzyValTriangle(double x) {
                double center = x_start+(x_end-x_start)/2.0;
                if (x < center) {
                        double x_end = center;
                        double m = (-1/(x_start-x_end));
                        double c = -m*x_start;
                        double y = m*x + c;
                        return (y > 1) ? 1 : (y < 0) ? 0 : y;
                } else {
                        double x_start = center;
                        double m = (1/(x_start-x_end));
                        double c = -m*x_end;
                        double y = m*x + c;
                        return (y > 1) ? 1 : (y < 0) ? 0 : y;
                }
        }
        double getFuzzyValUp(double x) {
                double m = (-1/(x_start-x_end));
                double c = -m*x_start;
                double y = m*x + c;

                return (y > 1) ? 1 : (y < 0) ? 0 : y;
        }
}
```

```java
//          Data.java
import java.util.HashMap;


public class Data {

        Enum fuzzy;
        public HashMap<Enum, Graph> mf;
        public Range range;

        public Data(HashMap<Enum, Graph> mf, Range range) {
                //this.fuzzy = fuzzy;
                this.mf = mf;
                this.range = range;
        }
}

//          TimeLevel.java
public enum TimeLevel {
        SHORT, MEDIUM, LONG
}

//          SizeLevel.java
public enum SizeLevel {
        SMALL, MEDIUM, BIG
}

//          HardnessLevel.java
public enum HardnessLevel {
        FEW, MEDIUM, VERY
}

//          FireLevel.java
public enum FireLevel {
        SOFT, MEDIUM, HARD
}

//          Fuzzy.java
public enum Fuzzy {
        TIME, SIZE, HARDNESS, FIRE
}
```

```
//      Range.java
public class Range {

        double start;
        double end;
        double step;

        public Range(double start, double end, double step) {
                this.start = start;
                this.end = end;
                this.step = step;
        }
}
```

```
//      Rule.java
import java.util.ArrayList;
import java.util.Map.Entry;

public class Rule {

        public ArrayList<RuleData> ifRule;
        public RuleData thenRule;

        public Rule(ArrayList<RuleData> ifRule, RuleData thenRule) {
                this.ifRule = ifRule;
                this.thenRule = thenRule;
        }
}
```

```
//      RuleData.java
RuleDatapublic class RuleData {

        public Enum fuzzy;
        public Enum level;

        public RuleData(Enum fuzzy, Enum level) {
                this.fuzzy = fuzzy;
                this.level = level;
        }
}
```