Specification of Group Project

"From Propositional logic to MP network"

COMP305 BIOCOMPUTATION

Lecturer: Chao Huang

1. Motivation

The aim of this project is to automatically transform a propositional logic formula to an equivalent MP neural network. This project is theoretically based on the fact that the propositional logic is a subset of MP neural networks in terms of representation power. It is deeply related to a key topic in the module, the ability and limitation of a neural network, and also touches on other learning outputs due to the special role of the McCulloch-Pitts neuron model, i.e., it is a special case of perceptron model.

2. Project outputs

Each group needs to provide three outputs.

- A program. Transform a propositional logic formula into a MP neural network.
    - Programming language: JAVA.
    - Input (See Appendix 2)
        - A variable declaration statement as "string x", "string a1".
        - A string that represents a propositional logic formula, where the logic operations ∧,∨, ¬ are represented as &&,||,!.
        - The string of propositional logic formula only includes the follow operations: (), ∧,∨, ¬.
        - No ∀, ∃ quantifiers, as they are for predicate logic.
        - No ←, → for simplicity.
    - Output (See Appendix 3)
        - It is a figure visualizing the MP network which is equivalent to the input.
- A report. A brief report in 2—3 pages that includes the following points.
    - Introduction of MP neuron and this project.
    - A structure diagram of the program.
    - Introduction of how the program is implemented.
    - Three test cases for your tool, each including the input and the expected output.
    - Contribution of each group member.
- A presentation. A **video** presentation that is 10 – 15 minutes long and includes the following points.
    - Introduction of MP neuron and this project.
    - Introduction of how the program is implemented.
    - Contribution of each group member.
    - Running one of the examples mentioned in the report.

3. Group and Submission

Group: 2 – 4 members. Self-selecting.
Important dates:

- Project specification released: 11 Oct. 2021 -- 14 Oct. 2021
- Group registration: 5 pm by the end of the tutorial session, 21 Oct. 2021
- Deadline: 5 pm by the end of the tutorial session, 18 Nov. 2021

How to register: Students can sign up on their own on Canvas system.

Submission: Once the groups are created on Canvas, the module coordinator will set assignments to be "group submission" assignments, which means each group will have one submission for all users of that group. Each group needs to submit a single zip package that contains the code, the report, and a link to the video presentation.

4. Assessment

This project: 15% of the final marks. (100% of this project = 15% of the final marks.)
Two criterions to assess the performance of your group:

- Implementation. The quality of the overall implementation of the program is evaluated based on the learning outcomes mentioned in Motivation section by 5 test cases.
    - Correct structure of a MP neuron;
    - Correct structure of a MP neural network;
    - Correct MP neural network that is equivalent to the given logic formula. Determined by the outputs of 5 test cases. If they all failed, then the three test cases provided by the group will be used. If the three test cases provided by the student group all pass, it will be equivalent to the case of that one official test case passes.
    - Optimize the neural network structure, i.e., when you can use a MP neuron with multiple inputs, try not to use a sequence of 2-input MP neurons (See Example 2 in Appendix). You DO NOT HAVE to simplify the given logic formula to achieve the network construction with the minimal number of neurons.
- Presentation. The quality of both report and the video presentation: Whether the key points mentioned in Section 2 are all addressed.

The marks for individual members are based on the assessment of the group, but may go up or down based on individual contributions.

If a student in any particular group has a declared disability, the module coordinator will monitor the peer moderation process to ensure that the particular student has not been treated unfairly as a result of their disability.

Feedback: The explanation on assessment will be given to all students within the group.

5. Final Caution

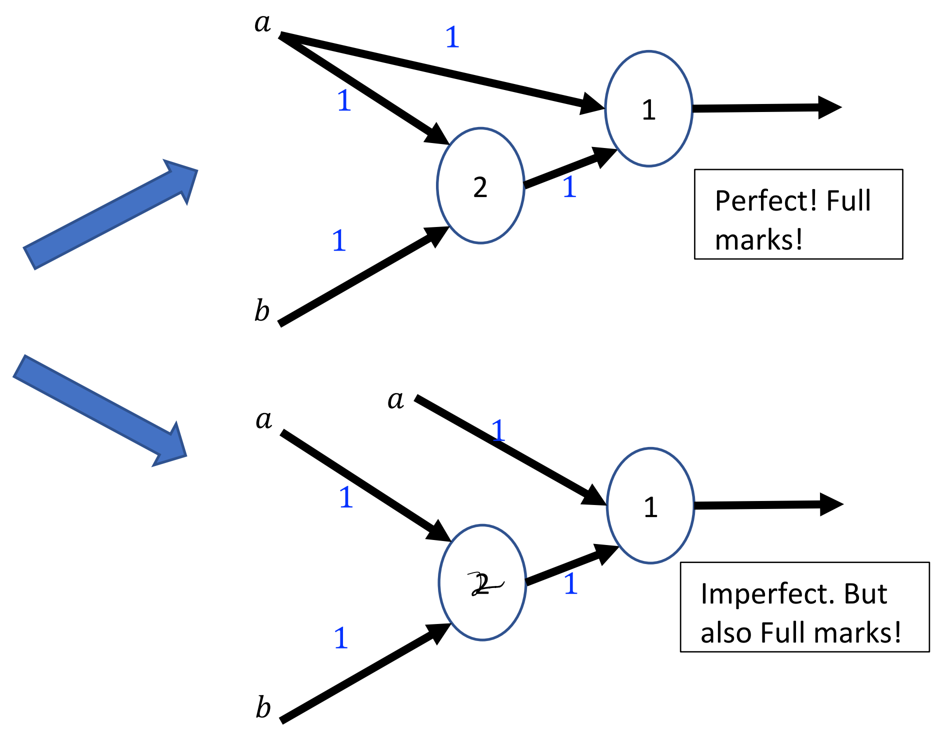
- Students should ensure that the final piece of submitted work is academically sound, without elements of plagiarism or other features of poor practice.
- Students can ask demonstrator or the module coordinator for advice, and can also see the tips mentioned in the appendix and the lecture slides for reference.
- All members of a group are responsible for monitoring the group's progress and should be prepared to access support from the module coordinator if the group is not functioning effectively.
- Individual students who feel that their group is not functioning effectively and/or are not happy with their treatment by the rest of the group should inform the coordinator concerned as soon as possible.
- how claims for extenuating circumstances will be handled is shown in Section 6, Appendix G, Code of Practice on Assessment.

Appendix.

1. Some examples

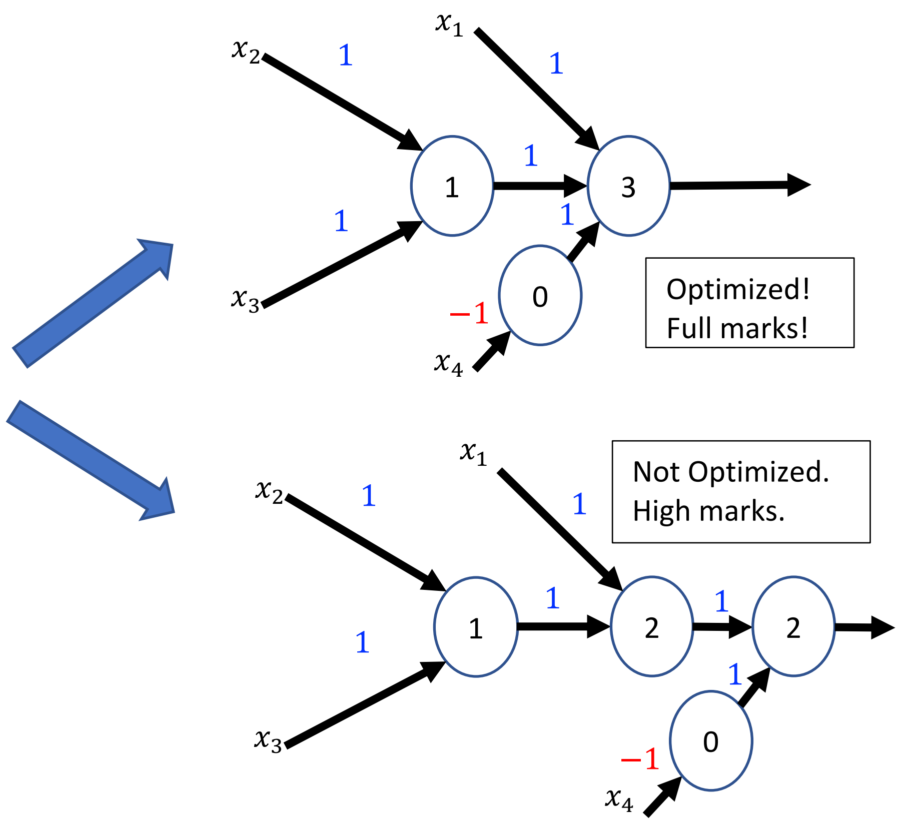
Example 1:

$a \vee (a \wedge b)$



Example 2:

$\neg\ x_1 \wedge \underline{(x_2 \vee x_3) \wedge \neg x_4}$



2. Input Parsing

- Students can freely use any technique to parse the input, e.g., an existing library, or implementing the input parsing themselves.
- In the following, a JAVA library to parse the input and the guidance on use is given for students' reference.

2.1. Specification of input parsing module

- Input: A java class file that contains input (variable) declaration, and the propositional logic formula.
- Output: A tree structure (Abstract syntax tree, AST) of Boolean propositional logic formula, represented by a root node (the detailed components of the formula can be traversed by checking the children of the root node).

2.2. IDE and Library

- Library: **JAVA PARSER** http://javaparser.org/
- Recommended IDE for programming: **IntelliJ IDEA** https://www.jetbrains.com/idea/
- How to import JAVA PARSER dependency to your own project in IntelliJ IDEA:
    - Click "File" in the tool bar and choose "Project Structure".
    - Project Settings → Modules→ "+" → 2 Library → From Maven.
    - Enter "com.github.javaparser:javaparser-symbol-solver-core:3.23.1" → Apply →OK.
- An example of how to obtain a root of the tree parsed from a logic formula can be found in Group_project_experiment.zip on Canvas.

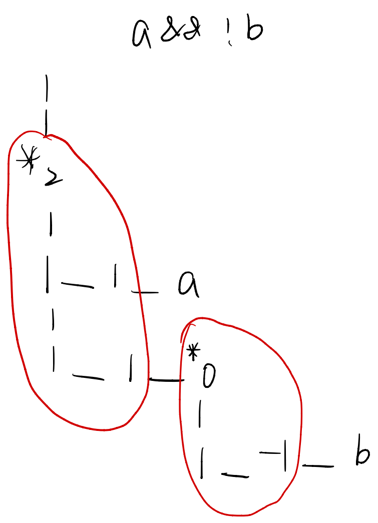
2.3. Concepts and Reading List

- Abstract syntax tree (AST): https://en.wikipedia.org/wiki/Abstract_syntax_tree
- JavaParser: Visited (User manual for JAVA PARSER): https://leanpub.com/javaparservisited (It is free)

3. Visualizing the Output

The output MP network can be demonstrated in the standard output. Here is an example output of the first example in Appendix 1.

```
|
|
*1
|
|__1__*2
|      |
|      |__1__a
|      |
|      |__1__b
|
|__1__a
```



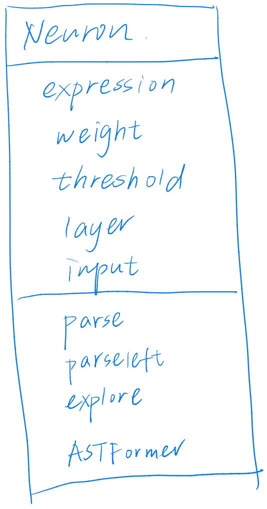
Here each number with a * presents the threshold, and each number without a * presents a weight, a,b are the variables in the propositional logic formula. Each single neuron with n inputs is represented as the following structure:

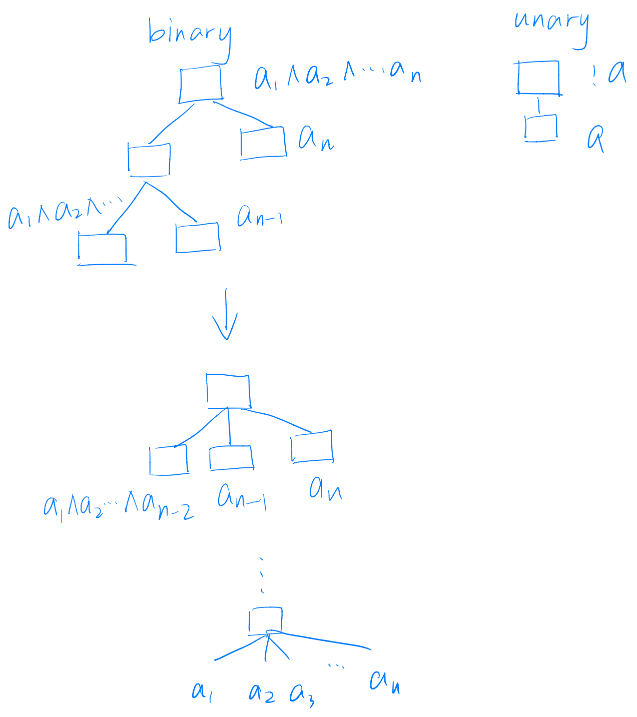
```
*threshold
|
|__weight 1__input 1
|
|__weight 2__input 2
…
|
|__weight n__input n
```

Note that each input could be the output of another neuron.
Finally, the output of the overall neural network is represented by two additional | in two lines, that is:

```
|
|
*threshold of the output neuron
|
|
…
```

Other formats of outputs are also accepted. In such cases, students need to clearly explain how the output matches a MP neural network.




binary
$a_1 \wedge a_2 \wedge \cdots a_n$
a_n
$a_1 \wedge a_2 \wedge \cdots$
a_{n-1}
unary
$!a$
a
$a_1 \wedge a_2 \cdots \wedge a_{n-2}$
a_{n-1}
a_n
a_1
a_2
a_3
$\cdots$
a_n

leftparse the left most neuron to find same operator.