How To Use Truth Tree 2

1. Open the program
    a. If an error occurs you may not have .Net 4 installed. Run the provided "dotNetFx40_Full_setup.exe" to install it. Retry opening the program
    b. If this did not work feel free to email me
2. You should now see a window as in Figure 1.
    a. Enter your premises into the box labeled "Premises" and your conclusion into the one marked "Conclusion".
    b. You must use prefix form and place an underscore before any variable name.
3. Press the "Parse" button
4. The selected tab should switch from "Input" to "Results" and you will see how your input has been interpreted as seen in Figure 2.
    a. If there are any mistakes select the "Input" tab and proceed to Step 2.
5. Press the "Satisfy" button to begin the proof generation.
    a. This operation may take a long time so if you wish to stop it you may press the "Stop" button.
6. Once the proof generation is complete the tab will switch from "Results" to "Proof" which is visible in Figure 3.
    a. If the conclusion follows from the premises then you will see that the conclusion is listed as "Valid". You may then press either of the "Show Tree" or "Show Proof" buttons to view the generated tree and proof respectively as show in Figure 5.
    b. If the conclusion does not follow from the premises then the conclusion will be show as "Invalid" which is shown in Figure 4. Since a proof cannot be generated for an invalid argument only the "Show Tree" button will be active.
    c. Should you wish to save either the tree or proof as an image you may select the "Save Tree" or "Save Proof" buttons which will open windows allowing you to choose where to save them.

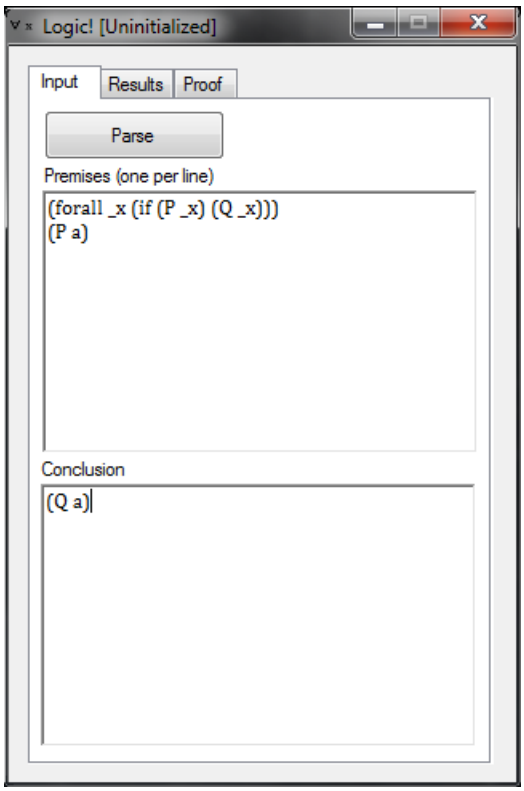

Figure 1

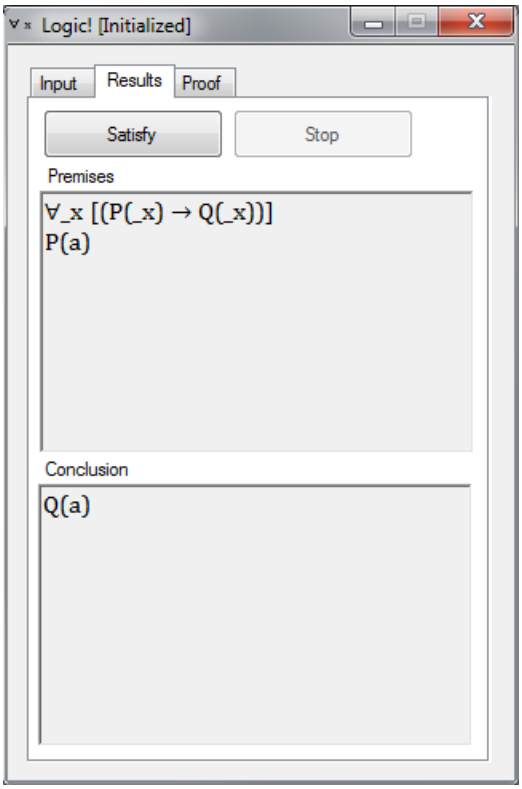

Figure 2

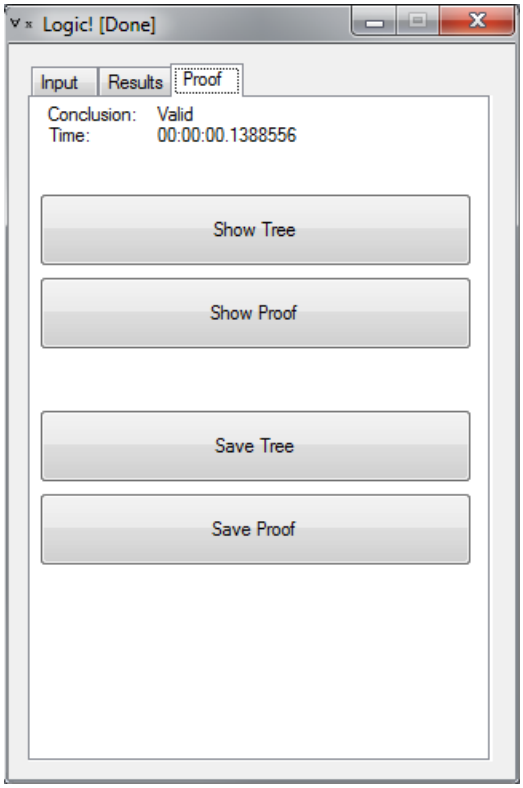

Figure 3

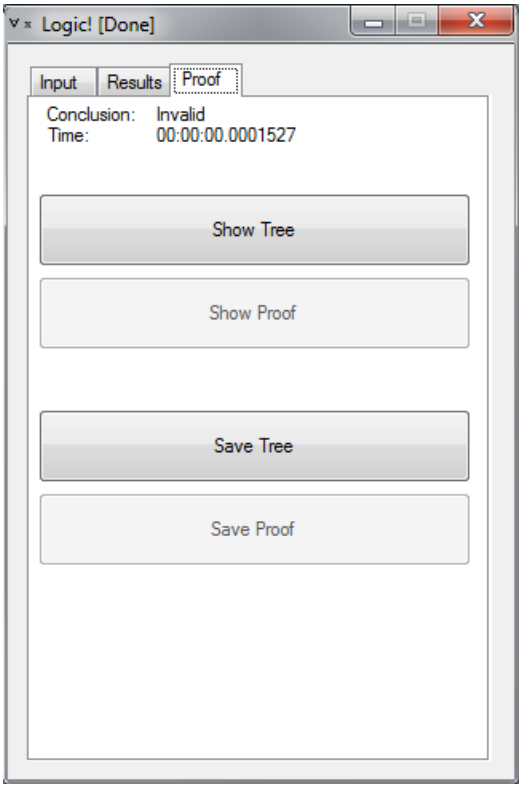

Figure 4

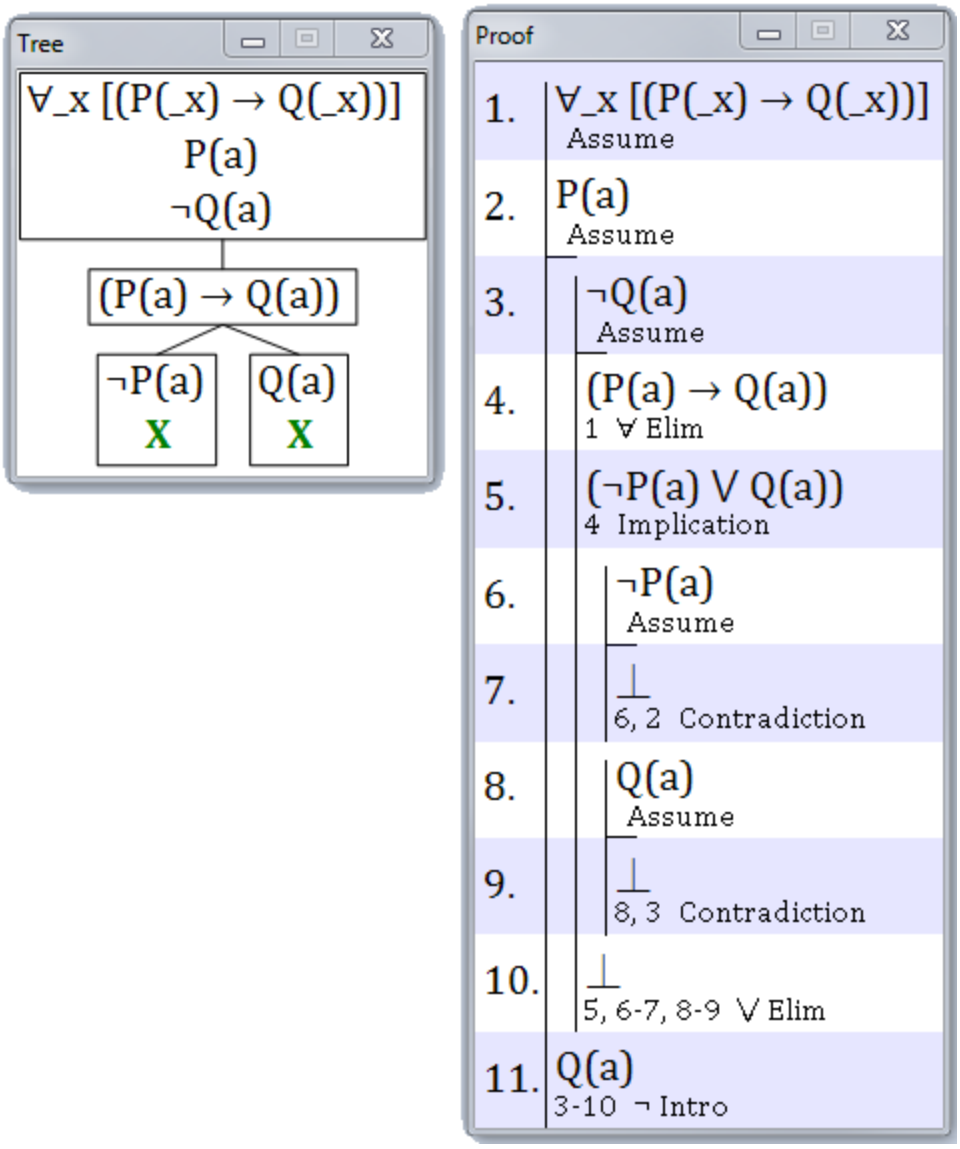
Tree
∀_x [(P(_x) → Q(_x))]
P(a)
¬Q(a)
(P(a) → Q(a))
¬P(a)
X
Q(a)
X
Proof
1. ∀_x [(P(_x) → Q(_x))] Assume
2. P(a) Assume
3. ¬Q(a) Assume
4. (P(a) → Q(a)) 1 ∀ Elim
5. (¬P(a) ∨ Q(a)) 4 Implication
6. ¬P(a) Assume
7. ⊥ 6, 2 Contradiction
8. Q(a) Assume
9. ⊥ 8, 3 Contradiction
10. ⊥ 5, 6-7, 8-9 ∨ Elim
11. Q(a) 3-10 ¬ Intro

Figure 5

Parsing Grammar

```
Initial: line

line : /* Empty */
     | formula
     | error
     ;
term : Identifier
     | Variable
     | LParen Identifier termlist RParen
     ;
termlist : termlist term
         | /* Empty */
         ;
formula : atomicwff
        | complexwff
        ;
atomicwff : False
          | LParen Equals term term RParen
          | LParen Identifier termlist RParen
          | Identifier
          ;
complexwff : LParen Not formula RParen
           | LParen Iff formula formula RParen
           | LParen If formula formula RParen
           | LParen And formulalist RParen
           | LParen Or formulalist RParen
           | LParen Exists Variable formula RParen
           | LParen Forall Variable formula RParen
           ;
formulalist : formulalist formula
            | /* Empty */
            ;

And:        "and"
Or:         "or"
Not:        "not"
Iff:        "iff"
If:         "if"
Forall:     "forall"
Exists:     "exists"
Equals:     "="
False:      "false"
LParen:     "(" | "[" | "{"
RParen:     ")" | "]" | "}"
Identifier: [a-zA-Z][a-zA-Z0-9]*
Variable:   _[a-zA-Z][a-zA-Z0-9]*
```