

Tableau Coursework Brief

COMP0009: Logic

Deadline: 28 November 2025, 16:00 UK Time

1 Technical Brief

For this assignment, you will be asked to implement a propositional and first order logic tableau, using Python. Your program will be run on an unseen input file and tested against its corresponding model output. Your program will be expected to correctly identify the type of a propositional/first order logic formula, and identify whether it is satisfiable, not satisfiable, or (in the case of FOL tableau) satisfiability cannot be determined after introducing 10 new constants in the δ -expansions on any given open branch.

Below we define both the language of propositional and first order logic. We limit our propositional letters to p, q, r, s , and our binary connectives to conjunctions, disjunctions, and implications. White space or extra brackets are not allowed in formulas.

```
FMLA := PROP                                (Proposition)
      | ~FMLA                               (Negation)
      | (FMLA*FMLA)                         (Binary Connective)

PROP := a|b|c|d    * := &|\vee|\rightarrow   (and, or, implies)
```

Similarly for FOL, we limit our variables to x, y, z, w , no function symbols, and the only predicates are binary predicates A, B, C, D .

```
var := x|y|z|w
PRED := A|B|C|D    * := &|\vee|\rightarrow
FMLA := PRED(var,var) (Atom)
      | ~FMLA                               (Negation)
      | EvarFMLA                            (Existentially Quantified)
      | AvarFMLA                            (Universally Quantified)
      | (FMLA*FMLA)                         (Binary Connective)
```

To see some examples, see the sample input/output file. The input file should have the first line containing **PARSE** in order for the program to produce parser output, or **SAT** to produce satisfiability output (or both). The remaining lines should be propositional or first order formulas.

If a formula is parsed as a formula, you should construct a Tableau and see if it has an open branch. For propositional formulas, you will be asked to determine whether or not it is satisfiable. For first order logic formulas, you may assume there are no free variables. You will be asked to try and determine satisfiability, however, the tableau might never close. Thus, if you are required to add more than 10 new constants on all open branches under some reasonable fair schedule, you may leave satisfiability undetermined.

2 Assessment Criteria

You must submit a single file, `tableau.py` that must be able to run as is on the lab machines using the below command, provided there is an `input.txt` file in the same directory.

```
python3 tableau.py > output.txt
```

A test will pass if and only if the `output.txt` file matches the model output file. If your program does not finish computing within a minute, it will timeout and the test will fail. You will be awarded marks according to the table below:

Task	Marks
Correctly parse propositional formulas	20
Correctly determine SAT for propositional formulas	20
Correctly parse first order logic formulas	20
Correctly determine SAT for FOL sentences that will not result in a loop under any schedule.	20
Correctly determine SAT for any FOL sentence, up to assigning 10 variables	20

Your final grade will be the sum of the marks for each section's test.

IMPORTANT:

- Your file must not include any import statements. If your submission contains an import statement, it will be considered as cheating, you will receive a mark of 0.
- All submissions must be submitted as a single plain ASCII encoded file named `tableau.py` that uses either CRLF or LF line endings. Failure to adhere to the submission format requirements will additionally result in your receipt of a 0 mark. All of these settings can be easily set in Visual Studio Code and we suggest you set them there.
- You *must* use the template file `skeleton.py` as a starting point for the coursework. This is for your convenience. We will not provide you with any assistance if you choose to not use this template.

- You *must not* modify anything below the commented line which says

DO NOT MODIFY THE CODE BELOW THIS LINE!

Modifying the code below this line *may* lead to your coursework not working as intended and thus getting a mark lower than that which you might expect (including a 0 mark).

TESTING YOUR SUBMISSION

Before submitting your code for assessment, you will be able to test it as many times as you'd like via a simple web app. It will show your `output.txt` file for the public test cases. As there are many test cases, we find it can be easier to see the output on this web app. The web app will be made available for use by the 8th of November 2025 via the following hyperlink: <http://logic2023.cs.ucl.ac.uk/>

Make sure to test your submission from the cs network. You may choose to use the cs vpn services as described in the CSWall section of <https://tsg.cs.ucl.ac.uk/working-from-home/>. Note that we actively monitor these submissions and will investigate any suspected malicious use of this tool. Anyone found cheating will receive a mark of 0 and may be referred to the academic misconduct board.