# Project Summary

The goal of this project is to classify a propositional logic formula as either a contradiction, tautology, or contingency using the Method of Analytic Tableaux (<https://en.wikipedia.org/wiki/Method_of_analytic_tableaux>) and a satisfiability solver.

A black line with letters and numbers

Description automatically generatedSpecifically, we take a propositional logic formula and its negation, produce the two corresponding tableaus (which are represented as trees) (regular\_tableaux and negated\_tableaux). Then, represent the two tableaus as a combination of propositions and relate them to the classification of the original formula. Finally, feed the propositions into a satisfiability solver to determine the classification of the original formula (contradiction/tautology/contingency).

Figure 1: Tableau generated for the formula .

A tableau is constructed by repeatedly applying pre-defined inference rules to a given formula to produce several simpler formulas which are eventually all literals (e.g. “a”, “not a”). The literals can then trivially be checked to see if they form a contradiction using negation elimination (the law of non-contradiction). Three of the 7 inference rules introduce branches into the structure of the tableau tree due to being disjunctive in nature. Each branch of a tableau can contain different formulas. For the initial formula to be a contradiction, every branch of the tableau tree must result in a contradiction. If at least one branch of the tableau does not result in a contradiction, then the initial formula is not a contradiction.

# Propositions

* *LiteralPropositioni,j,k,p*= True if the literal proposition ‘p’ is contained in branch ‘k’ of tableaux ‘j’ for formula ‘i’,.
* *BranchClosed*i, j, k = True if branch ‘k’ in tableaux type ‘j’ in formula ‘i’ contains both an atomic proposition and its negation. (For example, ‘a’ and ‘¬a’ in the same branch’)
* *TableauxClosed*i,j = True iff all branches contained in tableaux ‘j’ for formula ‘i’ are closed.
* *FormulaClassificationi, j =* True iff the formula ‘i’ has classification ‘j’ depending on if the regular tableaux and negated tableaux are open or closed:
  + Tautology= True iff regular tableaux is NOT closed and negated tableaux is closed.
  + Contradiction = True iff regular tableaux is closed and negated tableaux is NOT closed.
  + Contingency = True iff neither regular tableaux nor negated tableaux is closed.

# Constraints

*If an atom/negation of an atom is within a branch.*

*If an atom/negation of an atom is not within a branch.*

*The tableaux is closed if and only if all branches within the tableaux are closed as well.*

*Any branch is closed if and only if there is a pair of contradicting literals in the branch.*

*constraint to say if the regular/negated tableaux is closed, at most one of these constraints must be true.*

*Constraint to describe the classification of the tableaux, exactly one of these must be true.*

*)*

*Each of these constraints can be added based on the result of the classification*

# Model Exploration

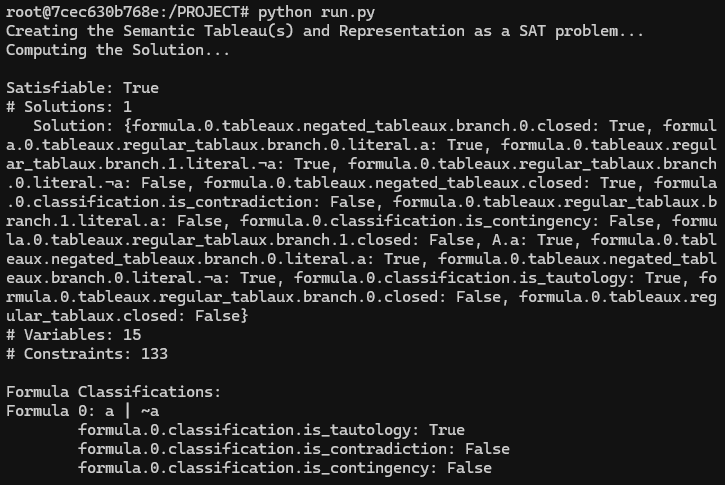
*List all the ways that you have explored your model – not only the final version, but intermediate versions as well. See (C3) in the project description for ideas.*

The initial version of our model takes a single string representing a propositional logic formula formed with atomic propositions and the connectives negation, conjunction, disjunction, implication. The goal of the model is to classify the input formula as either tautology, contradiction, or contingency. The classification of a particular formula is represented by the propositions “formula.{formula\_id}.classification.is\_tautology”, “formula.{formula\_id}.classification.is\_contradiction”, and “formula.{formula\_id}.classification.is\_contingency” where {formula\_id} is the integer index where the formula is stored in the list CANDIDATE\_FORMULAS.

The first formula tested was “” which corresponded to CANDIDATE\_FORMULAS[0]. This is an example of one of the well-known tautologies called the Law of Excluded middle, so the expected result is for the model to classify this formula as a tautology. This would be reflected in the theory with the corresponding valuation in the table below:

|  |  |
| --- | --- |
| Proposition | Valuation (True / False) |
| formula.0.classification.is\_tautology | True |
| formula.0.classification.is\_contradiction | False |
| formula.0.classification.is\_contingency | False |

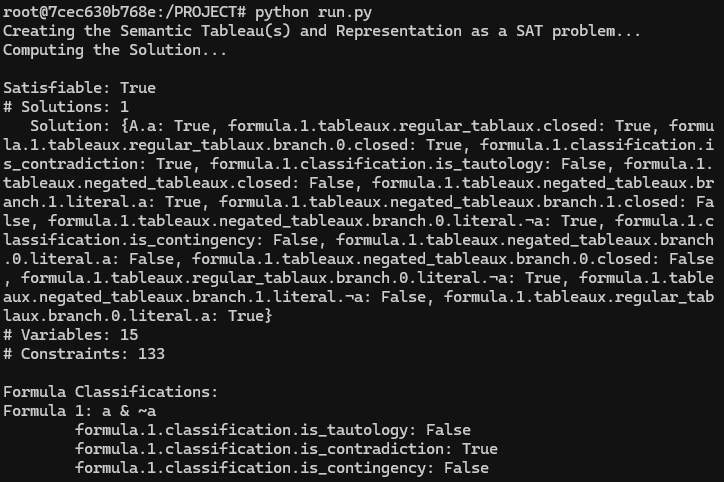
Running the model produced the following output:



It’s not easy to pick out the propositions of interest from the solution set with 15 variables, so the relevant information is summarized at the bottom. The Formula Classifications show that the output corresponds with what was expected in the table above.

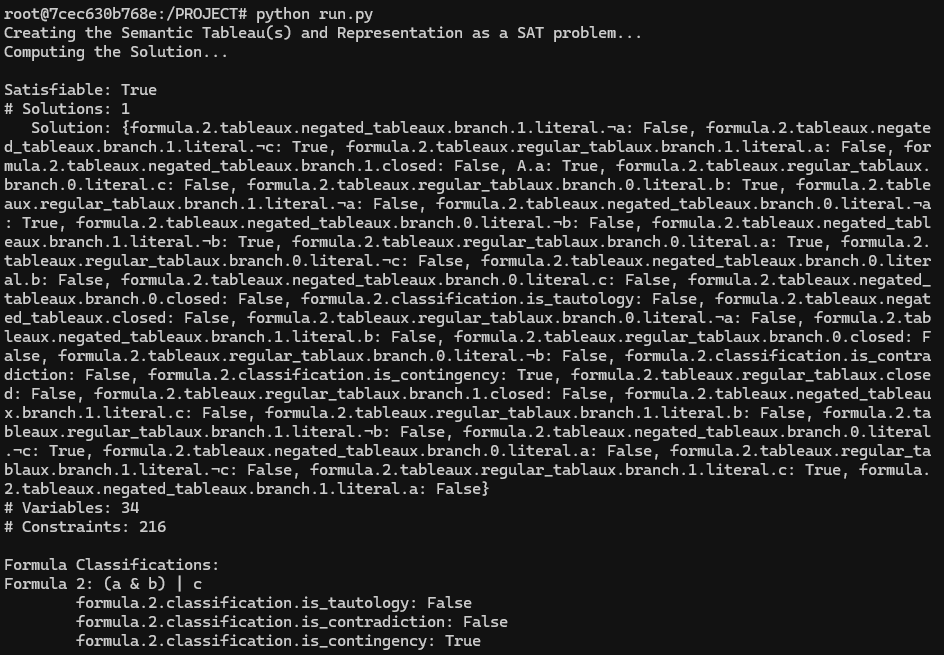
The second formula tested was “ which corresponds to CANDIDATE\_FORMULAS[1] and is expected to be a contradiction. The expected and actual results are detailed in the table and image below and show that the model is once again successful.

|  |  |
| --- | --- |
| Proposition | Valuation (True / False) |
| formula.1.classification.is\_tautology | False |
| formula.1.classification.is\_contradiction | True |
| formula.1.classification.is\_contingency | False |



One last test case to demonstrate a contingency, the formula tested was “” corresponding to CANDIDATE\_FORMULAS[2]. The expected and actual results are detailed in the following table and image, showing the model to produce the correct result.

|  |  |
| --- | --- |
| Proposition | Valuation (True / False) |
| formula.1.classification.is\_tautology | False |
| formula.1.classification.is\_contradiction | False |
| formula.1.classification.is\_contingency | True |



Additional testing on large formulas was also done, however the Solution set was omitted from being printed due to the large number of variables. The relevant tests are shown below.

A computer screen with white text

Description automatically generated

A screen shot of a computer

Description automatically generated

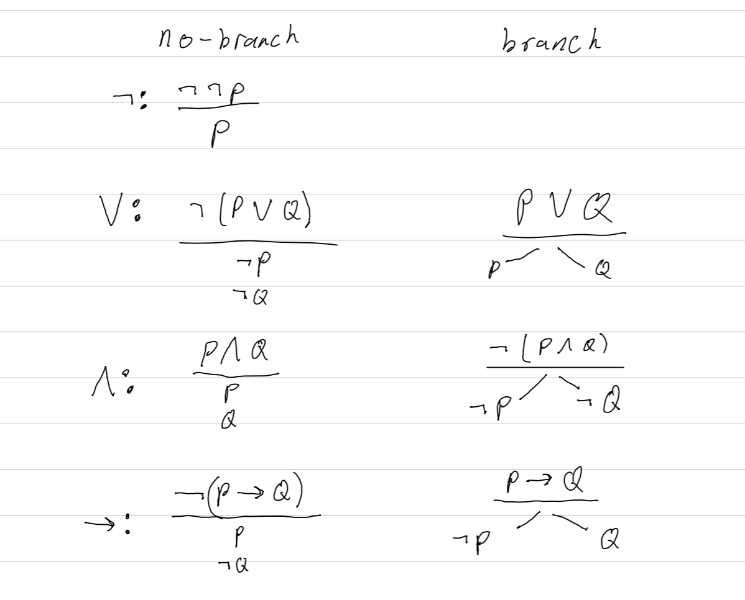
A computer screen with white text

Description automatically generated

Formula 5 is a contingency since it is a disjunction of 52 different atomic propositions. Formula 6 is a tautology since it is a disjunction of 52 different atomic propositions with the negation of one of them at the end. Formula 7 is a contradiction since it is a conjunction of 52 different atomic propositions added with the negation of one of them at the end.

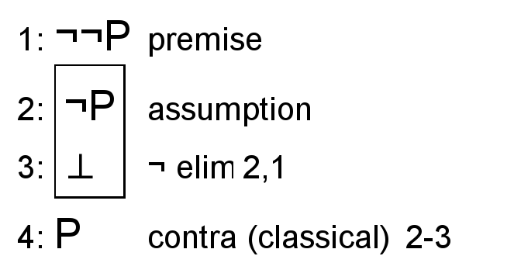
# Jape Proof Notes

This section has all the jape sequents along with their proofs that we used throughout the project to construct the analytic tableaux that determines propositional logic formulas as contradiction, tautologies and contingency. We breakdown the starting formula into simpler components by applying the following inference rules. Three of the sequents are used to introduce new branches in the tableaux. The simplification process by applying inference rules continues until we reach atomic propositions or contradiction.



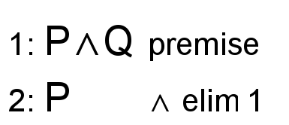
* **¬¬P ⊢ P:**

This is known as **Double Negation Elimination** which states that if the negation of the negation of a proposition P is true, then the proposition P must also be true.



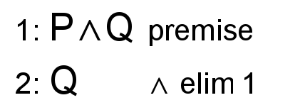
* **P ∧ Q ⊢ P:**

This sequent states that the if the conjunction of two propositions P and Q is true, then the proposition P should also be true.



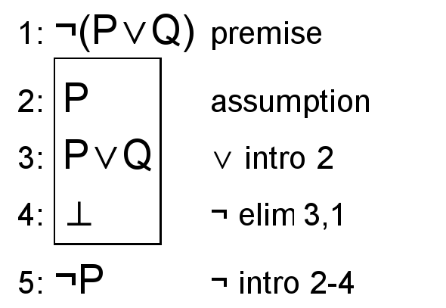
* **P ∧ Q ⊢ Q:**

This sequent states that if the conjunction of two propositions P and Q is true, then the proposition Q should also be true.



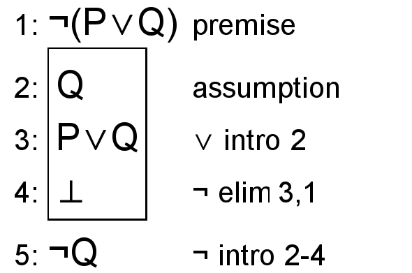
* **¬(P∨Q) ⊢ ¬P:**

If the negation of the disjunction of two propositions P and Q is true, then P should be false, or negation of P should be true.



* **¬(P∨Q) ⊢ ¬Q**

If the negation of the disjunction of two propositions P and Q is true, then Q should be false, or negation of Q should be true.



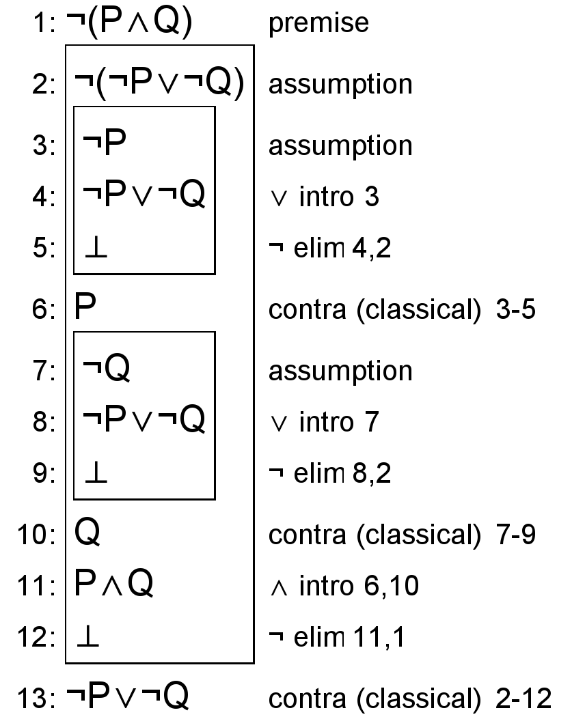
* **P∨Q ⊢ P∨Q**

If the disjunction of the propositions P or Q is true, that implies that the propositions P or Q are also true.

Inserting image...

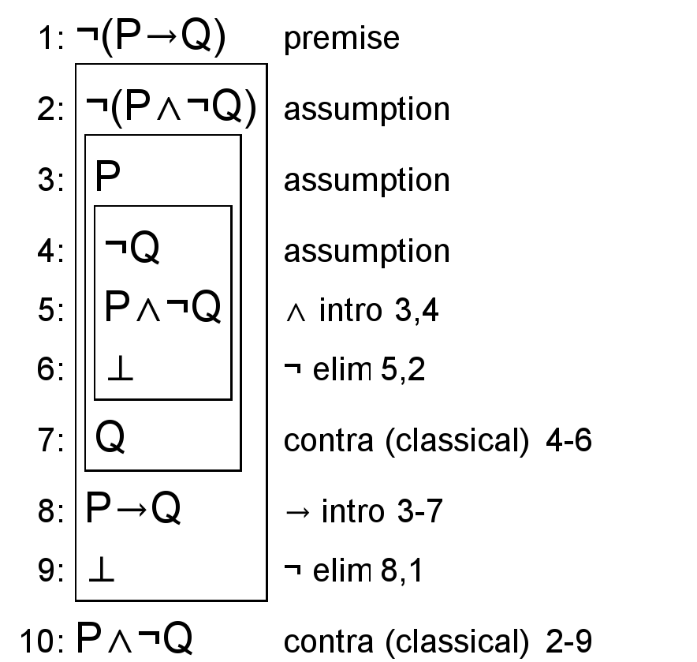
* **¬(P∧Q) ⊢ ¬P∨¬Q**

If the negation of the conjunction of two propositions P and Q is true, then either one of the proposition P or Q should be False or the negation of one of the propositions should be true.



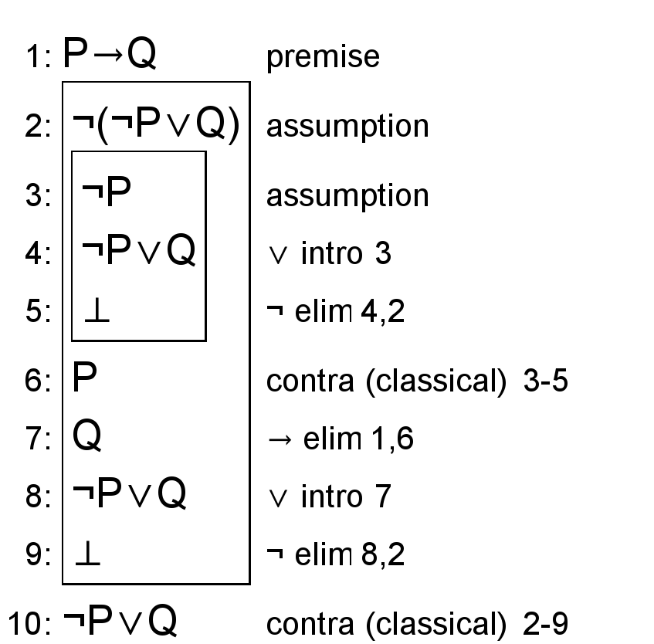
* **¬(P→Q) ⊢ P∧¬Q**

If the negation of the implication P→Q which translates to “if P then Q” is true, then P should be true, and Q should be false, or the conjunction of P and negation of Q must be true.



* **P→Q ⊢ ¬P∨Q**

If the implication of the two propositions P and Q which translates to “if P then Q” is true, then either P should be false or Q should be true or we can say disjunction not P or Q should be true.



# Requested Feedback

*2-3 questions we’d like the TA’s and other students to comment on.*

A significant amount of the work for this project was in generating the tableau for a given formula. It appears that this part of the project doesn’t fit nicely into any of the sections except for part of the summary and the Jape proofs for the inference rules. Would it be beneficial to elaborate on that part of the project more in perhaps a dedicated section?

Which sections could we add more detail to explain more clearly?

For the model exploration section, would it be helpful to add more detail on the intermediate outputs from the tableau branch generation (prior to the logic encoding)? What other aspects of the model do you think would be interesting to explore?

# First-Order Extension

*Describe how you might extend your model to a predicate logic setting, including how both the propositions and constraints would be updated.* ***There is no need to implement this extension!***

To model the constraints of this scenario using predicate logic. The first and last constraint types for presence of literals in branches and formula classifications would remain unchanged since they inherently do not involve quantification over variables. The second type of constraint of whether a branch of a particular tableaux for a particular formula is closed could be written as

The other constraint for checking if an entire tableau is closed is also easily written in predicate logic.

This project may also be extended to apply to classify (tautology, contradiction, contingency) general predicate logic formulas as well. The target would be to consider general propositional logic formulas and not consider whether formulas were true under certain models. The formulas would have to be expanded to support quantifier symbols and predicate symbols would have to be extended to have the ability to include variables as arguments.

Furthermore, additional inference rules to simplify the quantified formulas would have to be developed. And considering that it may not be possible to simplify predicate logic formulas into literals, the branch closure condition would need to be expanded beyond checking for matching literals.