# 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 : 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

* *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.
* *BranchContainsLiterali,j,k,w* = True iff branch ‘k’ of tableaux ‘j’ of formula ‘i’ contains literal ‘w’.
* *BranchContingentOnLiterali,j,k,w* = True iff branch ‘k’ of tableaux ‘j’ of formula ‘i’ is contingent on literal ‘w’.
* *BranchClosedOnVariablei,j,k,w* = True iff branch ‘k’ of tableaux ‘j’ of formula ‘i’ is closed on variable ‘w’.

# Constraints

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

*A branch is closed if and only if it’s closed on at least one pair of contradicting literals.*

*Constraint to say that at most one of the regular/negated tableaux for a particular formula can be closed.*

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

*)*

*Each of these constraints are added to enforce the result of the formula classification*

*Constraint to specify the variables a branch is closed on*

*Constraint to specify the variables the branch is contingent on*

*Constraints conditionally applied, depending on whether a literal () is in a branch.*

# Model Exploration

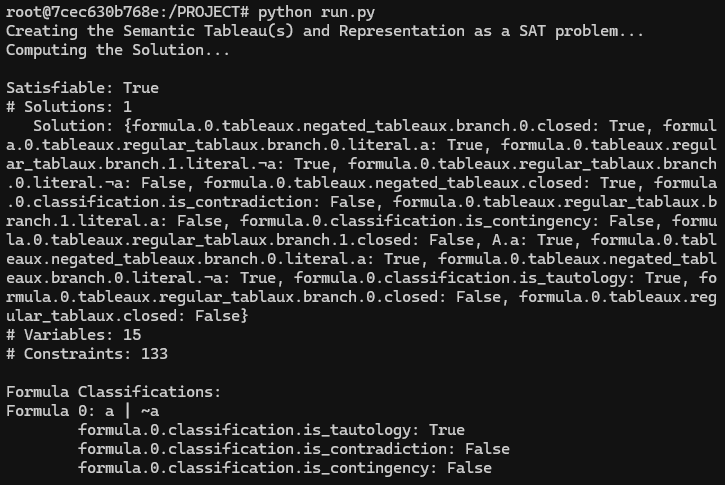
## Exploring the Initial Version of the Model

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 |

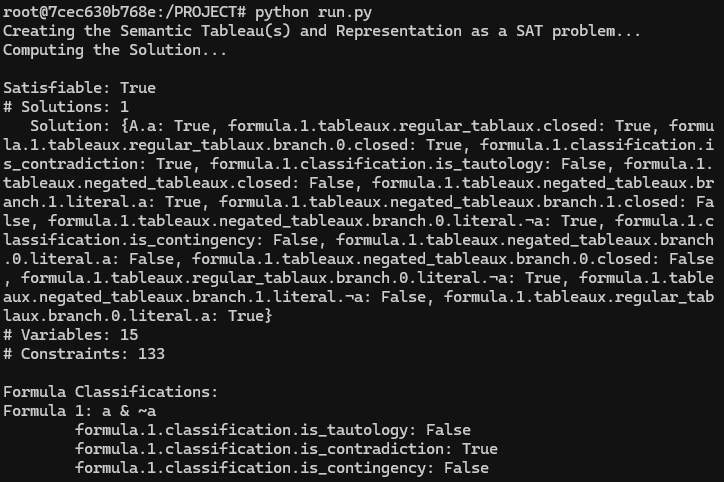
Running the model produced the following output:



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.

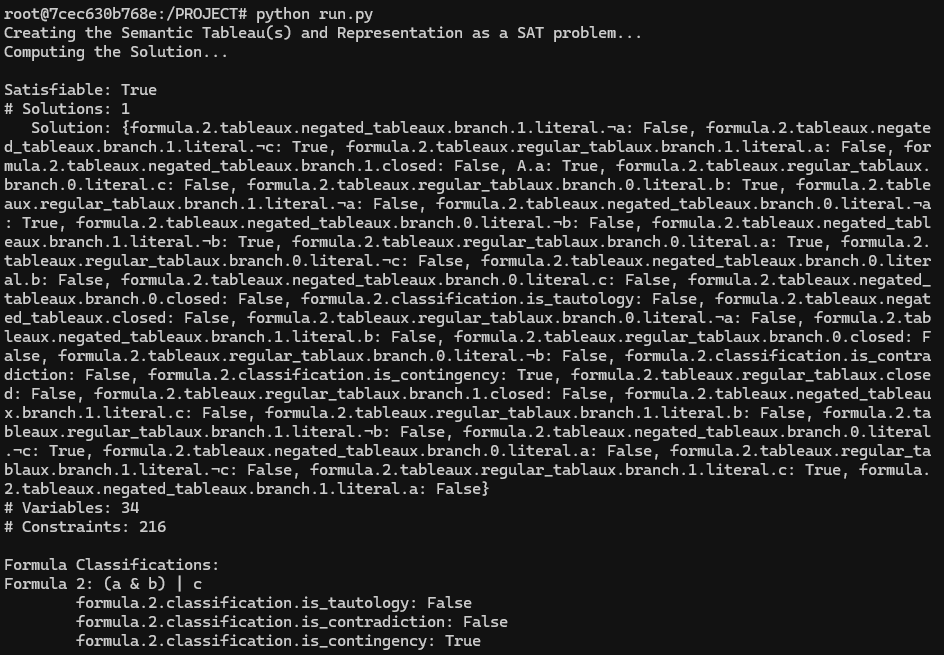
The second formula tested was “ which corresponds to CANDIDATE\_FORMULAS[1] and is expected to be a contradiction. The expected and actual results are detailed in the table and image below and show that the model is once again successful.

|  |  |
| --- | --- |
| 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.

There are still more questions that could be answered here, such as what variable assignments would make a contingent formula evaluate to true/false. What variables present in the formula are causing it to be a contradiction or tautology. These questions are explored in the updated version of the model in the following section.

## Exploring the Expanded Model

With the expansion of the model to extract additional information regarding the formulas and tableaux, additional properties can be explored. We will explore two formulas, a tautology and a contradiction, run them through the model, and remove variables from the formulas to see how that changes their classification.

First, considering formula 9 below. We see that it is a tautology caused separately by the variable a, and the variables b and e. This tells us that if all the instances of **a** and the related logical connected are removed from the formula, then it should no longer be a tautology. It also says that if all instances of both **b** and **e** are removed from the formula, it should also cease to be a tautology. These two modifications are explored with formulas 10 and 11 and they are shown in the figures below.

A computer screen shot of a program

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

Examining the above results for formulas 10 and 11, we see that after the modifications, the formula becomes a contingency. Highlighting the information shown about the contingency, we can see that formula 10 will be false if **d** is false as indicated by **[[d]]** and that it is true as long as d is true. The outer square brackets form a list representing a disjunction, and the inner square brackets represent a conjunction. The evaluation of the contingent formulas is essentially a compact form of a truth table.

Now to explore a contradiction formula. See the figure below of formula 12. It’s a contradiction as a result of the presence of variables **a** and **b**.

A computer screen with white text

Description automatically generated

Removing all the instances of variable a from formula 12 yielded formula 13 which is still a contradiction. This is because b is still present in the formula and the b was capable of causing the contradiction by itself. We also see with formula 14, removing b alone is also insufficient to change the classification of the original formula. It was not until all instances of both variables b and a were removed that the formula became a contingency.

A computer screen shot of a black screen

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

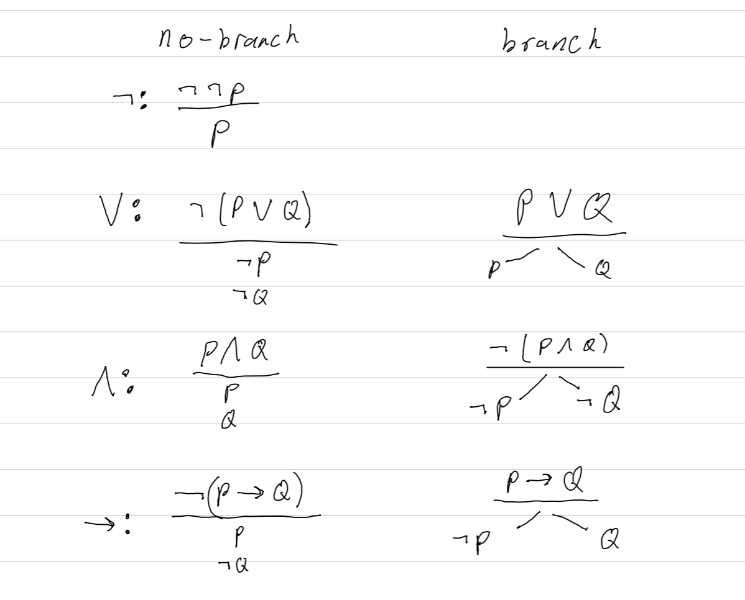
A screenshot of a computer program

Description automatically generated

This model exploration showed that for tautology formulas, removing the variables from at least one of the groupings caused it to no longer be a tautology, however with a contradiction, all variables involved in the causal groupings had to be removed to change the classification of the formula.

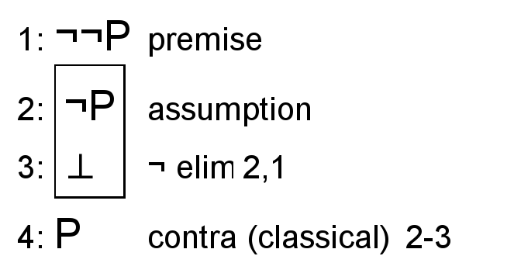
# Jape Proof Notes

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.



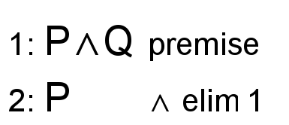
* **¬¬P ⊢ P:**

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.



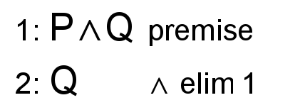
* **P ∧ Q ⊢ P:**

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



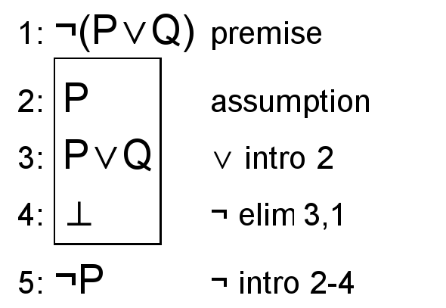
* **P ∧ Q ⊢ Q:**

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



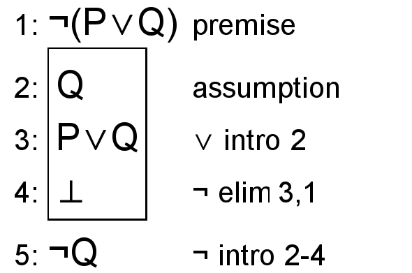
* **¬(P∨Q) ⊢ ¬P:**

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.



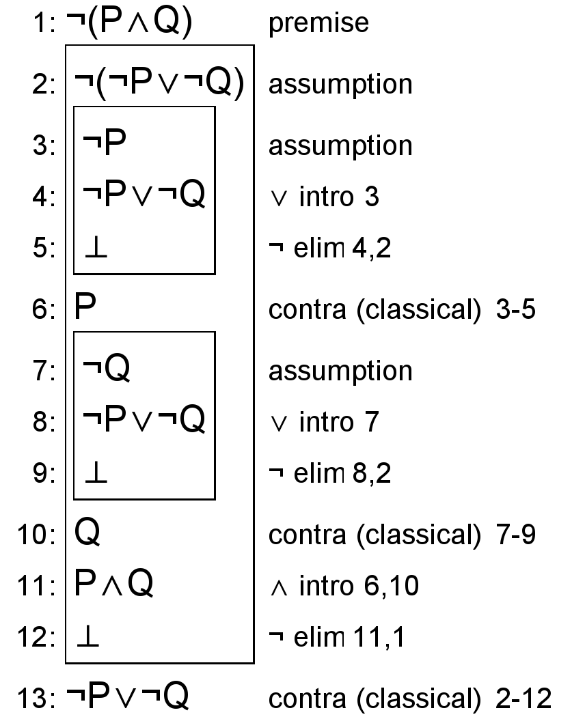
* **P∨Q ⊢ P∨Q**

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

Inserting image...

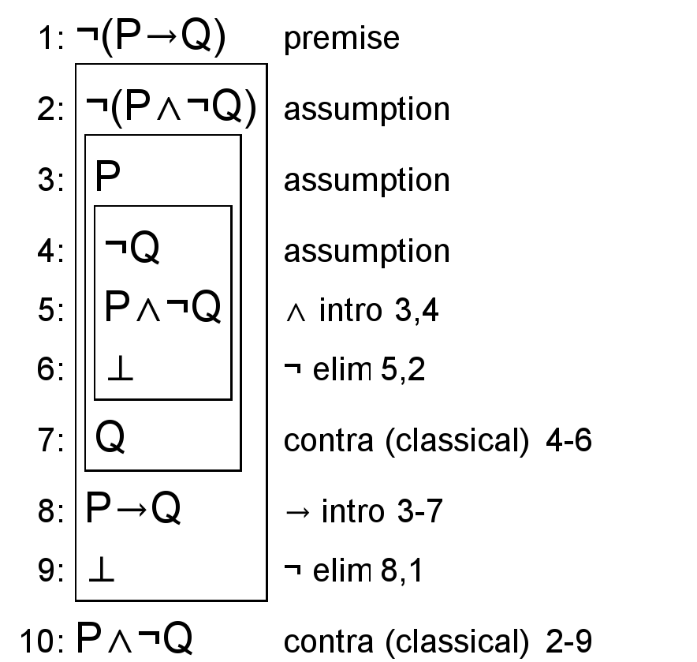
* **¬(P∧Q) ⊢ ¬P∨¬Q**

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.



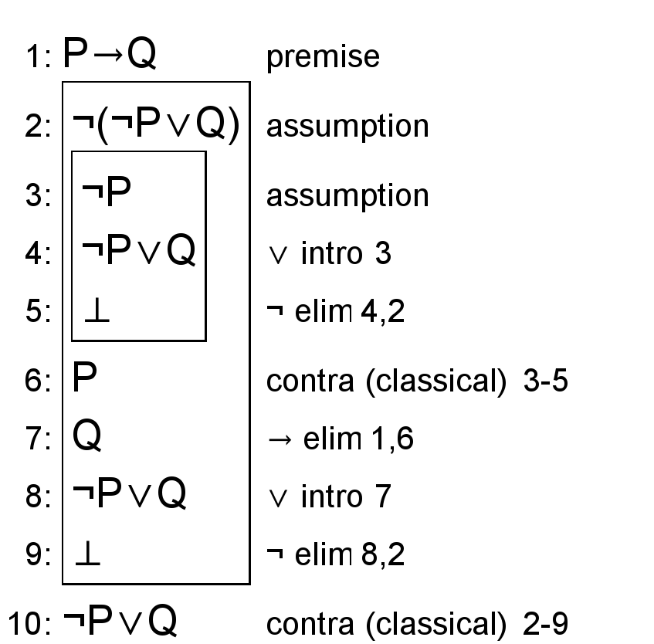
* **¬(P→Q) ⊢ P∧¬Q**

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.



# First-Order Extension

The following predicates and constraints model this situation using first-order logic. The constraints are made assuming a finite domain.

Predicates:

* : the tableau t contains branch b
* : literal l is in the branch b
* : the propositional variable/atom v is in the formula f.
* : defined by constraint
* : defined by constraint
* : defined by constraint
* Equality : “x is the same object in the domain as y”

Constraints:

* The nested bullets form conjunctions.
* + - This constraint is logically redundant due to the second existential formula above, but it is included for clarity.
    - Note: If an object in the domain of discourse is a negated formula variable, then there must exist a unique formula variable which is it’s opposite. This is a strict defining attribute, so converse is also true, hence the biconditional.
    - Note: If an object in the domain of discourse is a formula variable, then there exists a unique negated variable which is it’s opposite. This is a strict defining attribute, so converse is also true, hence the biconditional.

This project may also be extended to classify (as either tautology, contradiction, contingency) general first-order logic formulas as well. The target would be to consider general propositional logic formulas and not consider whether formulas were true under certain models/domains. 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.

Since predicate logic is undecidable, there would have to be some limit on the formula prover to be able to identify whether a given predicate logic formula could take infinite computations to verify, or if it would take way too many but finite computations.