Planning and Reasoning

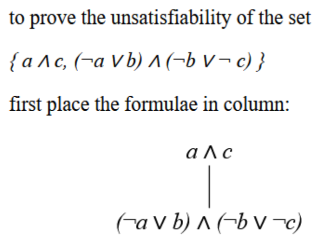
Sveva Pepe

Wednesday 2nd December, 2020

1 Propositional Tableau

Tableau method is completely different method because instead of trying working on variables, like other methods (GSAT, UP and DPLL), we proceed from the formula to the single variable. We are trying to break the formula into parts until the satisfiability can establish very simply. The idea is that we have a set of formulas (not set of clauses) and we want to establish the common satisfiability. We want to establish that all formula in the set if it is satisfied at the same time.

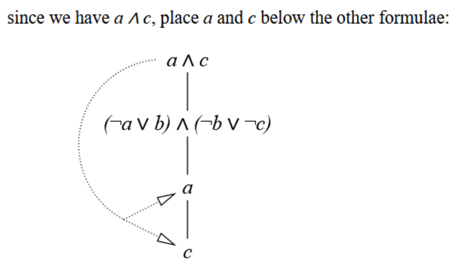
Example Propositional Tableau

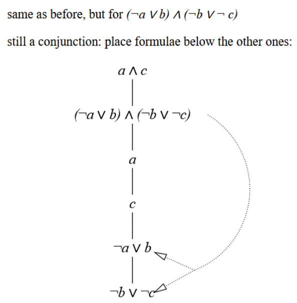

to prove the unsatisfiability of the set

$\{ a \wedge c, (\neg a \vee b) \wedge (\neg b \vee \neg c) \}$

first place the formulae in column:

The first idea is that if we have a set of formula we place them one under the other. "," is an "and" and when we have an "and" we put the formulas in column, like in the figure. The line between two formulas has the meaning that both formula has to be satisfied at the same time. It is a kind of graphical way to reproduce a conjunction.

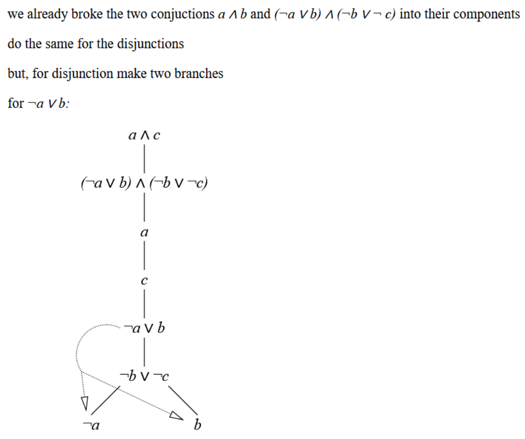

since we have $a \wedge c$, place a and c below the other formulae:


same as before, but for $(\neg a \vee b) \wedge (\neg b \vee \neg c)$

still a conjunction: place formulae below the other ones:

Now when I have disjunction?

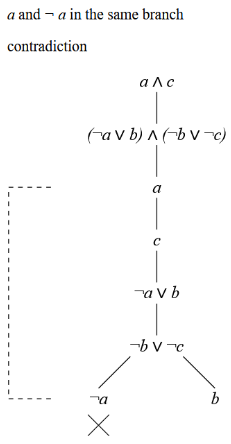
We need to create alternative because, for the the case of $\neg a \vee b$ we cannot put $\neg a$ and b in the same line because it is a disjunction and not a conjunction, so I split the line. In this case we split the line in two possibilitiesa and we do this for all disjunctions that appear in the formula.


we already broke the two conjuctions $a \wedge b$ and $(\neg a \vee b) \wedge (\neg b \vee \neg c)$ into their components

do the same for the disjunctions

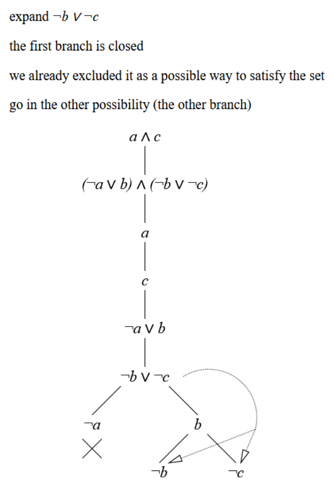
but, for disjunction make two branches

for $\neg a \vee b$:


a and $\neg a$ in the same branch

contradiction

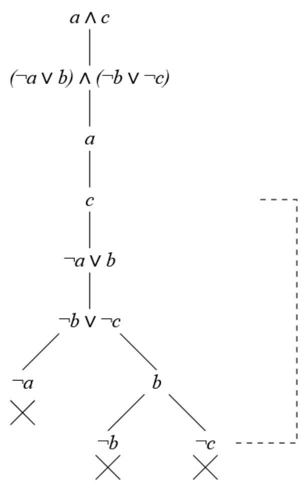
We can see that in the path, with $\neg a$ we have a contraddiction because we have a and $\neg a$ at the same time, so we make a cross, we close the branch. There is no way to satisfy formula in that particular path. X means that there is no model that could satisfied a and $\neg a$. This path is one of the possibilities but not actual possibility because it has a contraddiction inside. Ok we continue considering that we need to break again $\neg b \vee c$.


expand $\neg b \vee \neg c$

the first branch is closed

we already excluded it as a possible way to satisfy the set

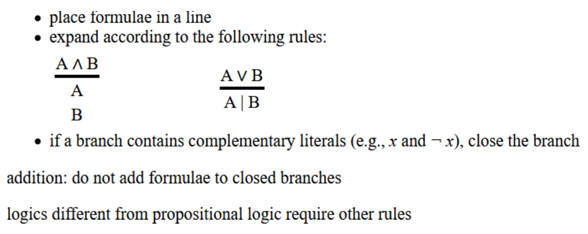
go in the other possibility (the other branch)



We have b and $\neg b$ in one of the two possibilities, so we have a contraddiction so we mark X, but in the other possibility we have a contraddiction because we have c and $\neg c$.

What we have done?
We have split the formula and also split the various possibilities to satisfied the formula.
In this particular case, I end up with situation in which all possibilities prove actually that formula is unsatisfiable. All possible ways to satisfy the formula are really not available because there is a contraddiction in each of them. The conclusion is simply that the formula is unsatisfiable because all possibilities are excluded.

1.1 Tableau Rules


- place formulae in a line
- expand according to the following rules:

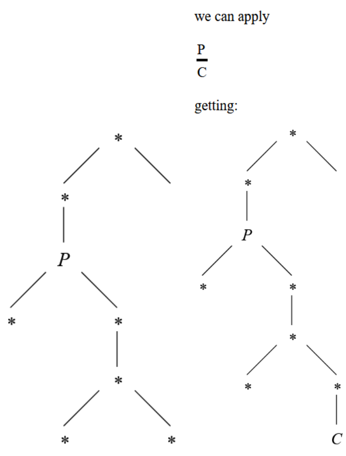
$$\frac{A \wedge B}{\begin{matrix} A \\ B \end{matrix}} \qquad \frac{A \vee B}{A \mid B}$$

- if a branch contains complementary literals (e.g., x and $\neg x$), close the branch

addition: do not add formulae to closed branches

logics different from propositional logic require other rules

We start from set of formulas and I want to check if all of them can be satisfied at the same time. So, i try to do the expansion of these formulas splitting them as much as possible because I wanto to check satisfiability just by looking at the individual possibilities and just by checking whatever have a pair of complementary literals.


we can apply

$$\frac{P}{C}$$

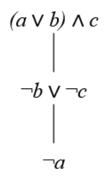
getting:

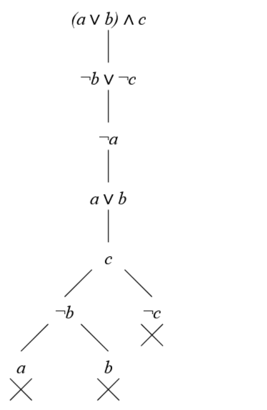
We have formula P somewhere in the middle of the tree.
What can we do now? Where we can expand P?
Suppose that we do not have close any branch. Consider that P = a,b we put a and b in every non close branch below P. In the case of C we put C there because the other branch below P are, for instance, closed.

Another Example

$\{(a \vee b) \wedge c, \neg b \vee \neg c, \neg a\}$

first step: place formulae in a line





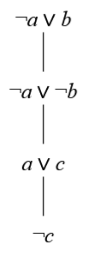
all branches close ⇒ set is unsatisfiable

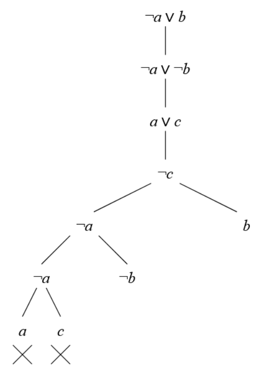
What happen if some branches do not close?
Remember that every branch is a possibility, a way to satisfy the formula. Closing the branch means that this branch is not a real way to satisfy the formula because I should be able to make same variable true and false and this is not possible. When I close all branches, all possibilities are excluded, but if some branches remain open this depends.

Why?
Because if I already expanded every formula in every possible way then, if I didn't run into a contraddiction, this may be a possible way to satisfy the formula.

Example 3

$\{\neg a \vee b, \neg a \vee \neg b, a \vee c, \neg c\}$



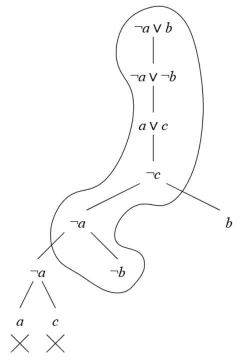


every formula has been expanded at least once

tableau not closed

yet, set is unsatisfiable

consider the branch ending in b



each branch is a different way to satisfy the set

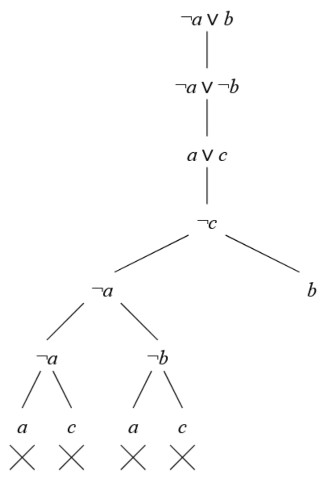
the formulae in this branch are: $\neg a \vee b, \neg a \vee \neg b, a \vee c, \neg c, \neg a, \neg b$

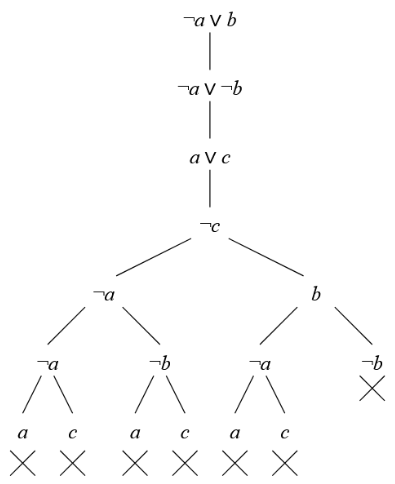
for $\neg a \vee b$ we took $\neg a$

for $\neg a \vee \neg b$ we took $\neg b$

no choice have been made for $a \vee c$

choose either a or c



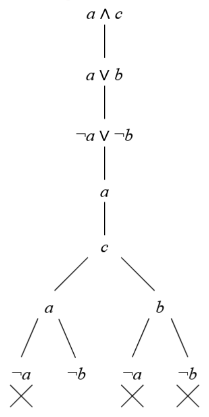


The size of the tableau depends on the order of the application of the rules but in this particular case there is no smaller tableau. The point is that in every non-closed branch, everyformula has to be expanded once. In Propositional case once means exactly one.

Open Branches

$\{a \wedge c,\ a \vee b,\ \neg a \vee \neg b\}$

expand every formula in every branch



in the second branch ($a \wedge c$ ... $\neg$ b) every formula has been expanded once

$a \wedge c$

taken both a and c

$a \vee b$

chosen a

$\neg a \vee \neg b$

chosen $\neg b$

this is a way to satisfy the set that does not lead to contradiction

the set is satisfiable

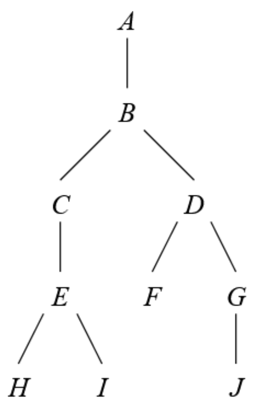
model: take the literals in the branch

$a,\ c,\ a,\ \neg b$

model: $\{a{=}true,\ b{=}false,\ c{=}true\}$

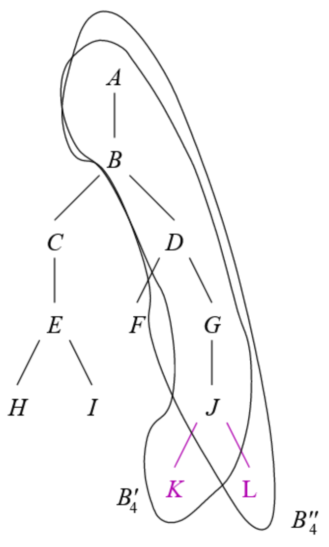
It is a good idea ti expand conjunction first because there is no alternative made out of the conjunction; instead, for disjunction we have alternatives so we have doubling the number of literals.

1.2 Semantics



We can see that the expansion of the formula is consist of a set of branches. In this case we have 4 set because we have 4 branches, 4 alternatives. For each of them I consider the conjunction of the formula in the branches, so each of them is represented by the formula in the branch. The first branch is: $B_1 = A \wedge B \wedge C \wedge E \wedge H$
The second branch is: $B_2 = A \wedge B \wedge C \wedge E \wedge I$
The third branch is: $B_3 = A \wedge B \wedge D \wedge F$
The fourth branch is: $B_4 = A \wedge B \wedge D \wedge G \wedge J$
The all tableau is an alternative $B_1 \vee B_2 \vee B_3 \vee B_4$

Now, suppose that $D = K \vee L$ and the branch that contains J is not closed yet. Since D is an alternative at the end of J we put 2 alternative, this means that the semantics changes because the tableau changes. This because the new tableau is not anymore an alternative between 4 possibilities but it becomes an alternative between 5 possibilities.

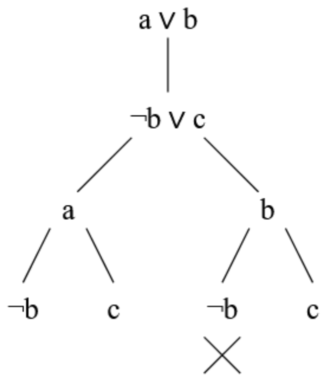


We need to write the formula that represented the semantic of the tableau also for the new tableau. The semantic of the tableau with 4 alternatives was: $B_1 \vee B_2 \vee B_3 \vee B_4$
B_4 is replaced by two new formulas: $B_4 = A \wedge B \wedge D \wedge G \wedge J$
$B'_4 = A \wedge B \wedge D \wedge G \wedge J \wedge K = B_4 \wedge K$
$B''_4 = A \wedge B \wedge D \wedge G \wedge J \wedge L = B_4 \wedge L$
The new semantic is: $B_1 \vee B_2 \vee B_3 \vee B'_4 \vee B''_4$
Equivalent to: $B_1 \vee B_2 \vee B_3 \vee (B_4 \wedge K) \vee (B_4 \wedge L)$
Equivalent to: $B_1 \vee B_2 \vee B_3 \vee (B_4 \wedge (K \vee L))$
This is like a forma proof of the *correctness* of the method. When I do an expansion what I did is basically to changes the syntattic structure of the formula, we just add something which is essentially equivalent to the original one because the new semantics of the tableau is just rearranging of the formula into an equivalent one. I increase the number of branches, the formula becomes longer but simpler because the subformula is expanded.

The point is that we want to make the formula more and more explicit, larger but at the same time simpler to check the satisfiability.
We make the formula larger but itis still equivalent to the original formula, so if I detect unsatisfiability in the resulting formula then the original formula is unsatisfiable as well.
The problem is not so much the correctness of the algorithm because the expansion in this way is basically the proof of correctness. Proving the completeness of the method is very complicated.

1.3 Partial Models

There are two possible endings. The first is that all branches are closed, which means that the formula is unsatisfiable because all possibilities are excluded, no one possibilities are achievable. The second ending is that some branches are closed and some other branches not, they remain open even though I expanded every formulas.



In the figure we can see that we have expand all formula. Each branch is a possibility, in particular, if the branch is opne we can find a model. We just look at the variables in each of this open possibility. We collect the literals and find the model (it should be partial). For instance, in the first open branch (start from the left) we have "a" and "¬b" which means $\{a = true, b = false\}$. In the second open branch we have $\{a = true, c = true\}$. Finally, the third open branch we have $\{b = true, c = true\}$. We do not consider the closed branch because it is an excluded alternative, impossible alternative.
If we look at the three models that we found we can see that they are not complete models, these are partial models because, for example in the first one model "c" is missing.
Furthermore, these partial models are not mutually exclusive because, for instance there is a model that satisfied both first and second open branches: $\{a = true, b = false, c = true\}$. It is *consistent* with two partial models that we found.

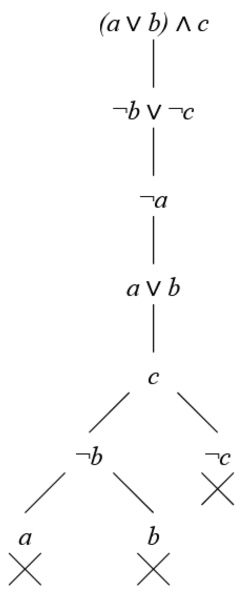
In DPLL (also Backtracking) instead, when we have an open branch[1] and you have a partial model of it. This

[1]branch that it is not closed in the recursive tree

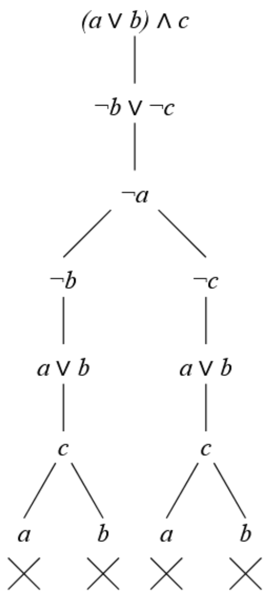
partial model is inconsistent with each other one. For instance, if you get two partial models into different branches they are not consistent with each other, always.

Why this?
Because the branching in DPLL means that somewhere a variables is evaluate true in one branch and false in another branch so there is no way that both of them satisfies both banches at the same time.
In the case of the Tableau method this is not the case because the branch is an alternative but not a mutual exclusive alternative.

1.4 Policy



conjunctions first



disjunctions first

In the picture pn the left, you can see the tableau obtianed just by apply the rules in a certain order. We expand conjunctions first and then disjunctions.
What happen if we do in the order way around?
In the picture on the right we have the tableau obtained by expanding disjunctions first.
Any difference?
Yes, in the picture on the left I only expanded the conjunction once.
What's the problem?
If I first expand disjunctions I branch and in both branches I have to expand the conjunctions, I'm duplicated them. Instead, if I first expand conjunction then disjunctions are still to be expanded once. so, it is better to expand conjunctions first.

For propositional logic making the wrong choice just descrease less the efficiency. But if I expand first disjunctions, this kind of wrong policy it doesn't effect the correctness or the completeness. The algorithm is still able whatever formula is satisfiable or not, always. It just increase the running time, increasing the size of the tableau, which automatically leads to an increase in the running time. it is not correctness o completeness that is perfect if we expand disjunctions first, but just the efficiency.
In FOL[2], if I do things in wrong sequence this may hurt correctness or completeness. This because FOL is something that is related to correctness of the method and not just the running time. Instead, propositional logic is just a matter of running time.

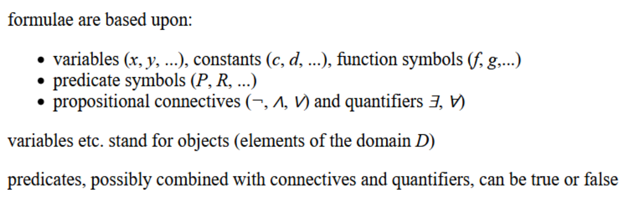
1.5 Entailment

If you want to check entailment of satisfiability, $A_1, ..., A_n \rightarrow B$, ne need to check the unsatisfiability of the negated thesis, so $\{A_1, ..., A_n, \neg B\}$ is unsatisfiable.

2 Recap FOL

Remember that in Propositional logic we have simple statements like true or false. Every variable in Propositional logic is true or false. So, in Propositional logic we have a set of paths that you want to make it true or false. Instead, in FOL I refined this concept by saying that things are not true or false but may be true or false depending on the object that are applied to. For example, in Propositional logic we can say a fact like "it's raining", instead, in FOL we can say "it's raining today, it's raining tomorrow, it's raining yesterday". So, my objects could be days and the statements are not something that it is true or false, but they are true in certain object and false for some other objects. In addition to that I have also functions.

[2]first order logic


formulae are based upon:

- variables (*x*, *y*, ...), constants (*c*, *d*, ...), function symbols (*f*, *g*,...)
- predicate symbols (*P*, *R*, ...)
- propositional connectives (¬, ∧, ∨) and quantifiers ∃, ∀)

variables etc. stand for objects (elements of the domain *D*)

predicates, possibly combined with connectives and quantifiers, can be true or false

The variables are the objects. In propositional logic variables are facts that can be true or false. In FOL variables are not facts, every variable is a representation of an object. The constants also represent objects and functions is something that given an object returns another object. All of these elements evaluate objects in a domain. This is only related to object.
As far as we consider only *variables, constants* and *function symbols* this is not true or false, just the object. FOL is still and object about true and false facts but this is not something pertaining to variables, constants and function symbols.

True and false entering into play only when we consider *predicate symbols* and *propositional connective.* These are about the objects but not the objects themselves.
When we consider the semantics of FOl we have to keep into account that we have two values:

- true and false
- objects (elements of the domain)

The point is that the parser of FOL formula evaluate either true or false, or object of the domain. The difference between these evaluation is what you apply. For example, if you have *terms*[3], so every form that contains only variables, constants or function symbols evalutates to an element of the domain (represntation of element of the domain). Everything that only contains constants, variable and function symbols is a representation of an object of the domain. Given terms we can apply predicates,combine them in every possible ways, and this is something that could be true or false.

[3]variable, constants and function symbols