Satisfiability Of Qualitative Contact Logics

Martin Stoev

2019-9-28

Contents

1 Tableaux Method **3**
1.1 Propositional logic tableau 3
1.1.1 Rules 4

2 Region-based Contact Logics **7**
2.1 Syntax 7
2.2 Semantics 8

3 Quantative Contact Logics **9**

4 Implementation Introduction **10**

5 Tableaux Implementation **11**

6 Model Implementation **12**

7 Quantative Contact Logics Implementation **13**

1 Tableaux Method

When working with logical formula it is best to define a structural way to decomposition the formula and validate some properties on parts of the whole formula. The ability from parts of the formula to conclude properties for the formula itself leads to well structured and performant algorithms.

The Tableau method is a formal proof procedure and can be found in many variants. Formally, this procedure is used to refute the validity of formulas. Specifically, if a formula X is given and we want to prove that X is valid, then we construct a new formula that will represent the invalidity of X. This new formula is usually created with additional syntactic expressions. Proving that the newly created formula is not valid, it implies proof of the validity of X. The proofing process uses steps to break down the given formula into simpler formulas. If these steps are applied multiple times, the proof of the initial complex formula comes down to a proof using syntactic comparisons of simple formulas.

For our purposes, we will use a slightly different version of the method described above, namely with the help of the decomposition steps we will try to find a model with certain properties in which the given formula will not be valid.

1.1 Propositional logic tableau

The tableau for propositional logic comes down to decomposing the operations a, b, c. Contained in the formula itself. To reach this goal we need to improve our syntactical arsenal, namely to add the ability for asserting the validity of a formula. In order to do this we shall introduce two signs. These signs when combined with a formula will assert wheter the formula is true or false. Formally, the two signs are { T, F }. Let φ be a formula, then

- Tφ - asserts the validity of the formula φ
- Fφ - asserts the invalidity of the formula φ

The tableau of a formula φ begins with Fφ. The intuition behind it is that we start with an assertion that φ is not valid in some model, and by continuing the tablea process we seek for contradictions in the subformals. If such contradiction exists then the initial guess that the formula is not valid is not true meaning that the formula is valid. On the other hand if a contradiction does not exist then the formula is not valid, additionaly a model can be constructed in which the formula φ is not valid.

Next we need to add the decomposition process in our arsenal, thus beeing able to apply the complete tableau process on a propositional formula. One formula is decomposed into subformulas by one of the tableau rules. There are two rules for each operation in the logic. In the case of propositional logic with the following operations $\{\neg, \vee, \wedge, \rightarrow, \leftrightarrow\}$ there are ten rules. Each rule decomposes the formula in at most two subformulas which depends on the arity of the operation. The decompositioning process forms a branch. A branch contains all signed subformulas which have been asserted in the decompositioning process.

A tableau rule which decomposes the formula in two subformulas may require the assertion of only one of the subformulas to be satisfied. In this case we will say that the branch is split into two branches. However in some terminologies this is noted as a new branch spawned from the original branch, or that two new branches are created . For the sake of simplicity we will use the terminology of branch splitting. In a scenario where the branch is split all asserted formulas in the branch are copied in the splitted branch together with the newly asserted subformula from the tableau rule.The newly asserted subformulas depend on the tableau rule.

1.1.1 Rules

Negation The rules for negations are straightforward. When a negation operation is encountered the sign of captured the formula changes.

$$\frac{\mathbb{T}(\neg\varphi), X}{\mathbb{F}(\varphi), X} \qquad\qquad \frac{\mathbb{F}(\neg\varphi), X}{\mathbb{T}(\varphi), X}$$

Conjunction This is a binary operation which means that the output of these rules is more complex than the negation rules. The conjunction rule for a formula signed as valid decomposes the formula in two subformulas and assert the validity of both of them. On the other hand the conjunction rule for a formula signed as invalid splits the branch into two branches and asserts the invalidity of the subformulas in each branch respectively.

$$\frac{\mathbb{T}(\varphi \ \wedge \ \psi), X}{\mathbb{T}\varphi, \mathbb{T}\psi, X} \qquad\qquad \frac{\mathbb{F}(\varphi \ \wedge \ \psi), X}{\mathbb{F}\varphi, X \qquad \mathbb{F}\psi, X}$$

Disjunction This is a binary operation which means that the output of these rules is more complex than the negation rules. The disjunction rule for a formula signed as valid splits the branch into two branches and asserts the validity of the subformulas in each branch respectively. On the other hand the disjunction rule for a formula signed as invalid decomposes the formula in two subformulas and assert the invalidity of both of them.

$$\frac{\mathbb{T}(\varphi \ \vee \ \psi), X}{\mathbb{T}\varphi, X \qquad \mathbb{T}\psi, X} \qquad\qquad \frac{\mathbb{F}(\varphi \ \vee \ \psi), X}{\mathbb{F}\varphi, \mathbb{F}\psi, X}$$

Implication The implication as well is a binary operation and depending on the asserting sign it decomposes the formula in two subformulas or splits into two separate branches.

$$\frac{\mathbb{T}(\varphi \ \rightarrow \ \psi), X}{\mathbb{F}\varphi, X \qquad \mathbb{T}\psi, X} \qquad\qquad \frac{\mathbb{F}(\varphi \ \rightarrow \ \psi), X}{\mathbb{T}\varphi, \mathbb{F}\psi, X}$$

Equivalence Both of the equivalence rules split the branch into separate branches, the details can be observed in the following equations.

$$\frac{\mathbb{T}(\varphi \leftrightarrow \psi), X}{\mathbb{T}\varphi, \mathbb{T}\psi, X \quad \mathbb{F}\varphi, \mathbb{F}\psi, X} \qquad \frac{\mathbb{F}(\varphi \leftrightarrow \psi), X}{\mathbb{T}\varphi, \mathbb{F}\psi, X \quad \mathbb{F}\varphi, \mathbb{T}\psi, X}$$

Closed branch A branch is said to be closed if and only if it contains the same formula signed as valid and invalid.

Closed tableau A tableau is said to be closed if after the decomposition process all branches are closed.

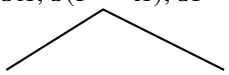
Tautology formula Now we have the total arsenal required to validate if one propositional formula is tautology or not. The process to determin if a formula is tautology is as follows: Let φ be a formula

1. Sign the formula as invalid, namely let $\mathbb{F}\varphi$ bet the initial formula in the tableau process
2. Execute the tableau process until the tableau contains only simple formulas in each branch
3. If all branches are closed, then the formula is a tautology otherwise it is not.

The tableau method is best explained through examples. First let us see an example of a formula which is a tautology.

Consider the following formula $((\neg X \wedge (Y \rightarrow X)) \rightarrow \neg Y$

1. $\mathbb{F}(((\neg X \wedge (Y \rightarrow X)) \rightarrow \neg Y)$
2. $\mathbb{T}((\neg X \wedge (Y \rightarrow X)), \mathbb{F}\neg Y$
3. $\mathbb{T}((\neg X \wedge (Y \rightarrow X)), \mathbb{T}Y$
4. $\mathbb{T}\neg X, \mathbb{T}(Y \rightarrow X), \mathbb{T}Y$
5. $\mathbb{F}X, \mathbb{T}(Y \rightarrow X), \mathbb{T}Y$

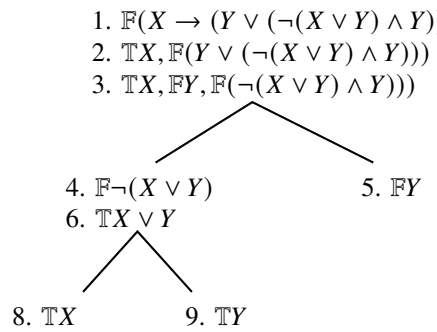


6. $\mathbb{F}Y$ 7. $\mathbb{T}X$

The first step of our tableau method for proving if a formula is a tautology is to sign it as not valid and analyze it. This means that at 1. this signed formula is added to the root of the tableau method. In the next step there is only one possibility, namely to decompose the only formula currently in the tableau process. This formula is decomposed to two formulas present in 2. At this point there are 2 possible outcomes, the first one is to decompose the formula $\mathbb{T}((\neg X \wedge (Y \rightarrow X))$. The second one is to decompose the formula $\mathbb{F}\neg Y$. In this example we have chosen the second approach and the result is present in step 3, namely $\mathbb{F}\neg Y$ after a tableau step is $\mathbb{T}Y$. From step 3 to step 4 there is only one possibility. The simple $\mathbb{T}\neg X$ formula is decomposed to $\mathbb{F}X$. in step 5. Now the decomposition of the formula $\mathbb{T}(Y \rightarrow X)$ will split the tableau branch into two branches. $\mathbb{F}Y$ will be added in the one of the branches, which in turn will contradict with $\mathbb{T}Y$ which can be observed from step 3. This means that the branch closes. In the

other branch $\mathbb{T}X$ is added which contradicts with $\mathbb{F}X$ from the previous step. This as well means that the second branch closes. Having a tableau with all branches closed proves that the initial assumption to sign the formula as invalid was wrong, hence the formula is a tautology.

The following tableau method is not closed and after the tableau process reaches all simple formulas a model in which the formula is not valid can be constructed. The formula used is $X \rightarrow (Y \vee (\neg(X \vee Y) \wedge Y))$.



In step 1 the input formula is signed as not valid and added to the tableau. This signed formula is decomposed to two signed formulas in step 2. Where one of them is already a simple formula and can not be decomposed in simpler signed formulas. The second one is decomposed and the results can be seein in step 3. Again one of the new signed formulas is a simple formula. The only formula that can be decomposed is $\mathbb{F}(\neg(X \vee Y) \wedge Y)))$ which splits the tableau branch.In the splitted branch $\mathbb{F}Y$ is added, this is step 5. This formula does not cause a contradiction in the branch and since thiere are no more possible decompositions it follows that the branch is not closed. At this point we knoww that the tableau is not closed. However for the purposes of this example let us examine the rest of the tableau process. In step 4 we can observe the additional signed formula which was added to the main branch. This signed formula is decomposed with the usage of one of the negation rules, as seen in step 6. Applying the disjunction rule for a formula signed as true on the formula in step 6 cases a branch split. Only one of the branches closes, namely the formula added in step 8. From this follows that the tableau is not closed and that the initial assumtion was true, namely the formula is not a tautology.

2 Region-based Contact Logics

The region-based theory of space has the notion of region and relations between regions as one of the basic primitive notions of the theory. A region is simply defined as a set of elements in some space. Union and intersection are used as primary operations over regions. Two basic relations are introduced such as part-of and the contact relation. The part-of relation constructs the structural dependencies between regions, namely the part-of relations defines the mereology of reagions. The contact relation defines the topological relation of connection. We will abstract ourselfs from the inner relation between two connected regions, namely this relation might be defined as "region A and region B are neightbours". What we will use for this inner relation is its reflexivity and symmetry.

2.1 Syntax

Variable represents a region. Two predefined variables shall be used in order to simplify the notation.

- W represents the whole world, namely the biggest region which contains all elements.
- $\emptyset$ represents the empty region

Let $\mathbb{V}ar$ be a countable set of all variables.

Boolean constants Boolean constants are defined for W and $\emptyset$, namely 1 represents the world, while 0 represents the empty region.

Boolean operations are operations over regions. The followings are boolean operations:

- $\sqcap$, denotes boolean intersection
- $\sqcup$, denotes boolean union
- * , denotes boolean complement

Term is defined with the following inductive definitions:

- Boolean constant is a term
- $p \in \mathbb{V}ar$ is a term
- If x is a term, then *x is a term
- If x and y are terms, then $x \ \sigma \ y$ is a term,
  where $\sigma \ \in \ \{\sqcap, \sqcup\}$

Propositional constants $\top$ and $\bot$

Propositional connectives $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$

Atomic Formulas A formula φ is called atomic, if it has one of the following forms:

- C(a, b), where a and b are terms
- a <= b, where a and b are terms

Formula is defined by the following inductive definition:

- Each propositional Constant is a formula
- Each atomic formula is a formula
- If φ is a formula, then $\neg\varphi$ is a formula as well
- If φ and ψ are formulas, then $\varphi\ \sigma\ \psi$ is a formula as well, where $\sigma \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$

Zero term formula Let a and b be two terms, then

$$a \leq b \iff a \sqcap \neg b = \emptyset \tag{1}$$

$a \sqcap \neg b = \emptyset$ will be called zero term formula. Since $a \sqcap \neg b$ is a new term, it can be assigned a variable $s = a \sqcap \neg b$, and now the zero term formula above can be written as $s = 0$ which is with better readability and can be easirly grasped in proofs and definitions.

2.2 Semantics

3 Quantative Contact Logics

4 Implementation Introduction

5 Tableaux Implementation

6 Model Implementation

7 Quantative Contact Logics Implementation