Satisfiability Of Connected Contact Logics

Anton Dudov

2019-9-28

Contents

1 Tableau Method For Classic Propositional Logic

1.1 What Is A Tableau?

A tableau method is a formal proof procedure. First, it could be used as a refutation procedure: to show a formula X is valid we begin with some syntactical expression intended to assert it is not. This expression is broken down syntactically, generally splitting things into several cases. This part of a tableau procedure - the tableau expansion stage - can be thought of as a generalization of disjunctive normal form expansion. Generally, it moves from formulas to subformulas. Finally, there are rules for closing cases: impossibility conditions based on syntax. If each case closes, the tableau itself is said to be closed. A closed tableau beginning with an expression asserting that X is not valid is a tableau proof of X.

There is a second way of thinking about the tableau method: as a search procedure for models meeting certain conditions. Each branch of a tableau can be considered to be a partial description of a model. In automated theorem-proving, tableaus can be used to generate counter-examples.

The connection between the two roles for tableaus - as a proof procedure and as a model search procedure - is simple. If we use tableaus to search for a model in which X is false, and we produce a closed tableau, no such model exists, so X must be valid.

1.2 Classical Propositional Tableaus

We will look into the signed tableau system for classical propositional logic.

First, we need syntactical machinery for asserting the invalidity of a formula, and for doing case analysis. For this purpose two signs are introduced: T and F, where these are simply two new symbols, not part of the language of formulas. *Signed formulas* are expressions of the form FX and TX, where X is a formula. The intuitive meaning of FX is that X is *false* (in some model). Similarly, TX intuitively asserts that X is *true*. Then FX is the syntactical device for (informally) asserting the invalidity of X. A tableau proof of X begins with FX.

Next, we need machinery (rules) for breaking signed formulas down and doing a case division. We will define rules for each logical operator ($\neg \wedge \vee \Rightarrow \Leftrightarrow$).

The treatment of **negation** is straightforward: from $T\ \neg X$ we get $F\ X$ and from $F\ \neg X$ we get $T\ X$. These rules can be conveniently presented as follows.

$$\frac{T\neg X}{FX} \qquad\qquad \frac{F\neg X}{TX}$$

The rules for **conjunction** are somewhat more complex. From truth tables we know that if $X \wedge Y$ is *true*, X must be *true* and Y must be *true*. Likewise, if $X \wedge Y$ is *false*, eigher X is *false* or Y is *false*. This involves a split into two cases. Corresponding syntactic rules are as follows.

$$\frac{TX \wedge Y}{\begin{array}{c} TX \\ TY \end{array}} \qquad\qquad \frac{FX \wedge Y}{FX \mid FY}$$

The rules for **disjunction** are similar. From truth tables we know that if $X \vee Y$ is *true*, eigher X is *true* or Y is *true*. This involves a split into two cases. Likewise, if $X \vee Y$ is *false*, X must be *false* and Y must be *false*. Corresponding syntactic rules are as follows.

$$\frac{TX \vee Y}{TX \mid TY} \qquad\qquad \frac{FX \vee Y}{\begin{array}{c} FX \\ FY \end{array}}$$

The rules for **implication**. From truth tables we know that if $X \Rightarrow Y$ is *true*, eigher X is *false* or Y is *true*. Likewise, if $X \Rightarrow Y$ is *false*, X must be *true* and Y must be *false*. Corresponding syntactic rules are as follows.

$$\frac{TX \Rightarrow Y}{FX \mid TY} \qquad\qquad \frac{FX \Rightarrow Y}{\begin{array}{c} TX \\ FY \end{array}}$$

The rules for **equivalence**. From truth tables we know that if $X \Leftrightarrow Y$ is *true*, eigher X is *true* and Y is *true* or X is *false* and Y is *false*. Likewise, if $X \Rightarrow Y$ is *false*, eigher X is *true* and Y is *false* or X is *false* and Y is *true*. Corresponding syntactic rules are as follows.

$$\frac{TX \Leftrightarrow Y}{\begin{array}{c|c} TX & FX \\ TY & FY \end{array}} \qquad\qquad \frac{FX \Leftrightarrow Y}{\begin{array}{c|c} TX & FX \\ FY & TY \end{array}}$$

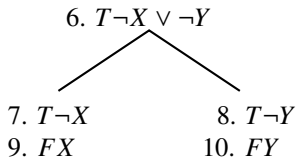
The standard way of displaying tableaus is as downward branching trees with signed formulas as node labels. Indeed, the tableau method is often referred to as the tree method. Think of a tree as representing the disjunction of its branches, and a branch as representing the conjunction of the signed formulas on it.

When using a tree display, a tableau expansion is thought of temporally, and one talks about the stages of constructing a tableau, meaning the stages of growing a tree. The rules given above are thought of as branch-lengthening rules. Thus, a branch containing $T \neg X$ can be lengthened by adding a new node to its end, with $F X$ as a label. Likewise a branch containing $F X \vee Y$ can be lengthened with two new nodes, labelled $F X$ and $F Y$ (take the node with $F Y$ as the child of the one labelled $F X$). A branch containing $T X \vee Y$ can be split - its leaf is given a new left and a new right child, with one labelled $T X$, the other $T Y$. This is how the schematic rules above are applied to trees.

An important point to note is that the tableau rules are non-deterministic. They say what can be done, not what must be done. At each stage, we choose a signed formula occurrence on a branch and apply a rule to it. Since the order of choice is arbitrary, there can be many tableaus for a single signed formula.

Here is the final stage of a tableau expansion beginning with the signed formula $F(X \wedge Y) \Rightarrow (\neg X \wedge \neg Y)$.

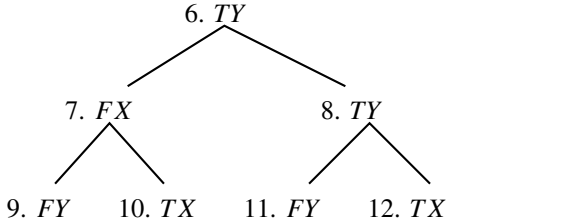
1. $F(X \wedge Y) \Rightarrow \neg(\neg X \vee \neg Y)$
2. $TX \wedge Y$
3. $F\neg(\neg X \vee \neg Y)$
4. TX
5. TY



In this we have added numbers for reference purposes. Items 2 and 3 are from 1 by $F \Rightarrow$. 4 and 5 are from 2 by $T\wedge$. 6 is from 3 by $F\neg$. 7 and 8 are from 6 by $T\vee$. 9 is from 7 by $T\neg$. 10 is from 8 by $T\neg$.

Finaly, the conditions for closing off a case (declaring a branch closed) are simple. A **branch is closed** if it contains TA and FA for some formula A. If each branch is closed, then the **tableau is closed**. A closed tableau for FX is a tableau proof of X, meaning that A is a tautology. The tableau displayed above is closed, so the formula $(X \wedge Y) \Rightarrow (\neg X \wedge \neg Y)$ has a tableau proof.

It may happen that no tableau proof is forthcoming, and we can think of the tableau construction as providing us with contraexamples. Consider the following attempt to prove $(X \Rightarrow Y) \Rightarrow ((Y \Rightarrow X) \Rightarrow \neg Y)$



Item 2 and 3 are from 1 by $F \Rightarrow$, as are 4 and 5 from 3. Item 6 is from 5 by $F\neg$. Items 7 and 8 are from 2 by $T \Rightarrow$, as are 9 and 10 from 4. Items 11 and 12 are also from 4 by $T \Rightarrow$. The leftmost branch is closed because of 6 and 9. The left-right branch is closed because of 7 and 10. The right-left branch is closed because of 8 and 11. But the rightmost branch is not closed. Notice that every non-atomic signed formula has had a rule applied to it on this branch and there is nothing left to do. (For clasical propositional logic it is sufficient to apply a rule to a formula on a branch only once.) In fact the branch yields a counterexample, as follows. Let υ be a propositional valuation that maps X to *true* and Y to *true* in accordance to 8 and 12. Now, we work our way back up the branch. Since $\upsilon(Y) = true$, $\upsilon(\neg Y) = false$, item 5. From $\upsilon(X) = true$ follows that $\upsilon(Y \Rightarrow X) = true$, item 4. From $\upsilon(Y) = true$ follows that $\upsilon(X \Rightarrow Y) = true$, item 2. Since $\upsilon(Y \Rightarrow X) = true$ and $\upsilon(\neg Y) = false$ we have $\upsilon((Y \Rightarrow X) \Rightarrow \neg Y) = false$, item 3. Finally, $\upsilon((X \Rightarrow Y) \Rightarrow ((Y \Rightarrow X) \Rightarrow \neg Y)) = false$, item 1.

Later, we are going to use the tableau method to produce a model in which the initial formula is valid.

From a different point of view, we can think of a classical tableau simply as a set of sets of signed formulas: a tableau is the set of its all branches, and a branch is the

set of signed formulas that occur on it. Semantically, we think of the outer set as the disjunction of its members, and these members, the inner sets, as conjunctions of signed formulas they contain. Considered this way, a tableau is a generalization of disjunctive normal form (a generalization because of formulas more complex than literals can occur). Now, the tableau construction process can be thought of as a variation of the process of converting a formula into a disjunctive normal form.

2 Contact Logics

2.1 Syntax

The language of contact logic consist of:

- *Boolean variables* (a denumeratable set)
- *Boolean constants*: 0 and 1
- *Boolean operations*:
  - $\sqcap$ boolean meat
  - $\sqcup$ boolean join
  - * boolean complement
- *Boolean terms* (or simply *terms*)
- *Propositional connectivies*: $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$
- *Propositional constants*: $\top$ and $\bot$
- *Modal connectives*: $\leq$(part-of) and C(contact)
- *Complex formulas* (or simply *formulas*)

Terms are defined in the following inductive process:

- Each Boolean variable is a term
- Each Boolean constant is a term
- If a is a term then $a*$ is a term
- If a and b are terms then $a \sqcap b$ and $a \sqcup b$ are terms

Atomic formulas are of the form $a \leq b$ and aCb, where a and b are terms.
Formulas are defined in the following inductive process:

- Each propositional constant is a formula
- Each atomic formula is a formula
- If ϕ is a formula then $\neg\phi$ is a formula
- If ϕ and ψ are formulas then $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \Rightarrow \psi)$ and $(\phi \Leftrightarrow \psi)$ are formulas

Abbreviations: $a = b \stackrel{\text{def}}{=} (a \leq b) \wedge (b \leq a)$, $a \not\leq b \stackrel{\text{def}}{=} \neg(a \leq b)$, $a \neq b \stackrel{\text{def}}{=} \neg(a = b)$

2.2 Relational semantics

Let $\mathcal{F}$ = (W, R) be a relational system with W $\neq \varnothing$ and R $\subseteq W^2$. We call such systems *frames*. Following Galton[TODO] we may give a spatial meaning of frames naming the elemnts of W, *cells* and the relation R, *adjacency relation*. Then $\mathcal{F}$ is called *adjacency space*. An example of adjacency space is the chess-board table, the cells are the squares, and two squares are adjacent if they have a common point.

Originally Galton assumed R to be a reflexive and symmetric relation but it is more natural for R to be an arbitrary relation. *Regions* in an adjacency space are arbitrary subsets of W and two sets a and b are in *contact* (aC_Rb) if for some $x \in a$ and $y \in b$ we have xRy. Another way to define this relation is the following. For a subset $a \subseteq W$ define as in modal logic $\langle R\rangle a = \{x \in W : (\exists y \in W)(xRy \text{ and } y \in a)\}$. Then aC_Rb iff $a \cap \langle R\rangle b \neq \emptyset$ iff $\langle R^{-1}\rangle a \cap b \neq \emptyset$, where R^{-1} is the converse relation of R. Note that if R is a symmetric relation then aCb iff $a \cap \langle R\rangle b \neq \emptyset$ iff $\langle R\rangle a \cap b \neq \emptyset$.

By a ***valuation*** of the Boolean variables in $\mathcal{F}$ we mean any function υ assigning to each Boolean variable a a subset $\upsilon(a) \subseteq W$. The valuation υ is then extended inductively to all Boolean terms as follows:

- $\upsilon(0) = \emptyset$
- $\upsilon(1) = W$
- $\upsilon(a \sqcap b) = \upsilon(a) \cap \upsilon(b)$
- $\upsilon(a \sqcup b) = \upsilon(a) \cup \upsilon(b)$
- $\upsilon(a*) = W \setminus \upsilon(a) = -\upsilon(a)$

The pair $\mathcal{M} = (\mathcal{F}, \upsilon)$ is called ***model***. The truth of a formula ϕ in $\mathcal{M}$ ($\mathcal{M} \models \phi$ or $\mathcal{F}, \upsilon \models \phi$) is defined inductively as follows:xtended inductively to all Boolean terms as follows:

- For atomic formulas:
  - $\mathcal{M} \models \top$
  - $\mathcal{M} \not\models \bot$
  - $\mathcal{M} \models a \leq b \iff \upsilon(a) \subseteq \upsilon(b)$
  - $\mathcal{M} \models aCb \iff \upsilon(a)\, C_R\, \upsilon(b \iff (\exists x \in \upsilon(a))(\exists y \in \upsilon(b))(xRy)$
- For complex formulas:
  - $\mathcal{M} \models \neg\phi \iff \mathcal{M} \not\models \phi$
  - $\mathcal{M} \models \phi \wedge \psi \iff \mathcal{M} \models \phi$ and $\mathcal{M} \models \psi$
  - $\mathcal{M} \models \phi \vee \psi \iff \mathcal{M} \models \phi$ or $\mathcal{M} \models \psi$
  - $\mathcal{M} \models \phi \Rightarrow \psi \iff \mathcal{M} \not\models \phi$ or $\mathcal{M} \models \psi$
  - $\mathcal{M} \models \phi \Leftrightarrow \psi \iff (\mathcal{M} \models \phi$ and $\mathcal{M} \models \psi)$ or $(\mathcal{M} \not\models \phi$ and $\mathcal{M} \not\models \psi)$

Let us note that in the above semantics we evaluate formulas not locally at points[TODO ??], as it is in the standard modal semantics, but globally in the whole model and this is one of the main differences of the present modal approach with the standard Kripke approach.

A model $\mathcal{M}$ is a ***model of a formula*** ϕ if ϕ is *true* in $\mathcal{M}$.

$\mathcal{M}$ is a model of a set of formulas *A* if $\mathcal{M}$ is a model of all formulas from A.

A formula ϕ is *true* (or valid) in a frame $\mathcal{F}$ ($\mathcal{F} \models \phi$), if $\mathcal{M} \models \phi$ for all models $\mathcal{M}$ based on $\mathcal{F}$, i.e. for all valuations υ we have $\mathcal{F}$,$\upsilon \models \phi$.

3 Connected Contact Logics

4 Implementation Introduction

5 Tableaux Implementation

6 Model Implementation

7 Connected Contact Logics Implementation

References

[1] Handbook of Tableau Methods M. DâĂŹAgostino, D. M. Gabbay, R. HÂĺahnle and J. Posegga, eds.

[2] Modal Logics for Region-based Theories of Space Philippe Balbiani, Thinko Tinchev, Dimiter Vakarelov