Satisfiability Of Modal Logic Formulas

Martin Stoev, Anton Dudov

2019-9-28

1 Formula Representation

Let $\mathbb{V}ar$ be the set of variables:

$$\mathbb{V}ar = \{p_0, p_1, p_2...\}$$

Let $\mathbb{C}_t$ be the set of Term constants:

$$\mathbb{C}_t = \{0, 1\}$$

1.1 Term recursive definition

- $a \in \mathbb{C}_t$ is a term
- $p \in \mathbb{V}ar$ is a term
- If x is a term, then $\bar{x}$ is a term as well
- If x and y are terms, then $x \sigma y$ is a term as well, where $\sigma \in \{\sqcap, \sqcup\}$

Let $\mathbb{C}_f$ be the set of formula constants:

$$\mathbb{C}_f = \{T, F\}$$

1.2 Formula recursive definition

- $a \in \mathbb{C}_f$ is a formula
- If x and y are terms, then C(x, y) is a formula
- If x and y are terms, then $x \leq y$ is a formula
- If x and y are terms, then $x \leq_m y$ is a formula
- If ϕ is a formula, then $\neg\phi$ is a formula as well
- If ϕ and ψ are formulas, then $\phi \, \sigma \, \psi$ is a formula as well, where $\sigma \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$

1.3 Definition: Atomic Formula

A formula will be called atomic formula if it is a constant or it is in one of the followings:

- C(x, y)
- $x \leq y$
- $x \leq_m y$

where x and y are terms.

2 Tableaux

The Tableaux process is decision procedure, which recursively breaks down a given formula into basic components based on which a decision can be concluded. The recursive step which breaks down a formula creates one or two new formulas, which in terms of their structure are simpler then the initial formula. Since the recursive step can create at most two new formulas, this means the recursive step will create at most two branches or a binary tree, where the nodes are the formulas and the links represent the recursive step. The different branches are considered to be disjuncted while nodes of the same branch are considered in conjunction. The procedure modifies the tableau in such a way that the formula represented by the resulting tableau is equivalent to the original one.

Contradiction may arise when in the same branch, on some step there exists a formula and the negation of the same formula. If in some branch there exists a contradiction, then that branch closes. If all branches close then the proof is complete.

The main principle of the tableaux is to break complex formulae into smaller ones until complementary pairs of literals are produced or no further expansion is possible.

2.1 Definition: Tableaux Step

The Tableaux Step takes as input a formula and a set of accumulated formulae and produces as output one or two new formulae, depending on the operation. The set of accumulated formulae consist of the broken down formulae by previous tableaux steps. The output of the tableaux step depends on the rule applied to the formula.

2.1.1 Rules

Negation

$$\frac{\neg\varphi, X}{\varphi, X}$$

And

$$\frac{\varphi \ \wedge \ \psi, X}{\varphi, \psi, X}$$

Or

$$\frac{\varphi \ \vee \ \psi, X}{\varphi, X \qquad \psi, X}$$

Implication

$$\frac{\varphi \ \rightarrow \ \psi, X}{\neg\varphi, X \qquad \psi, X}$$

Equivalence

$$\frac{\varphi \leftrightarrow \psi, X}{\varphi, \psi, X \qquad \neg\varphi, \neg\psi, X}$$

The final output of the Tableaux process is "False" when all branches are closed or a set of atomic formulae, when there exists a branch which is not closed.

For our usecases the functionality of the tableaux process shall be extended to achive better results, since if the branch is not closed, there are additional calculations needed in order to verify that there is no contradiction, namely to verify that there is no contradiction on Term level. This verification can be done in different manners, depending on the algorithm type. The best way to think about it is to have the tableaux process return a list, where each element is the set of atomic formulae found in a specific branch. This way the atomic formulae for each branch are produced, and afterwards can be used in different algorithms. This is just an example, a way of thinking about the problem the real implementation is much more space efficient.

2.2 Tableaux implementation

The programming implementation of the tableaux method follows the standard tableaux process explaned above.

2.2.1 Definition: Marked Formula

Let φ be a formula and X be the set of accumulated formulae, then φ is said to be marked as:

- true if and only if $\mathbb{T}\varphi \in X$
- false if and only if $\mathbb{F}\varphi \in X$

First interesting design decision is to keep all true formulae in one data set, and all false formulae in another data set. This enables fast searches wheter a formula has been marked as true or false. There exist 8 important collections of data used for storing formulae. This means that each collection represents a set of formulae:

- formulas_T_ - contains only formulae marked as true
- formulas_F_ - contains only formulae marked as false,
  For example, if $\neg\varphi$ is encountered as an output of the tableaux step, then only φ is inserted into the formula_F_
- contacts_T_ - contains only contacts formulae marked as true
- contacts_F_ - contains only contacts formulae marked as false
- zero_terms_T_ - contains only formulae of type $\varphi \leq \psi$ marked as true

- zero_terms_F_ - contains only formulae of type $\varphi \leq \psi$ marked as false
- measured_less_eq_T_ - contains only formulae of type $\varphi \leq_m \psi$ marked as true
- measured_less_eq_F_ - contains only formulae of type $\varphi \leq_m \psi$ marked as false