KRR: Activity 1 – Semantic Tableaux Method for Propositional Logic

Alexandru Sorici
12.10.2020

Introduction

Boolean logic formula satisfiability is one of the problems that existed at the very beginning of AI research (symbolic AI), and which exists today as separate lines of research in, for example, *circuit design*, *planning for industrial applications* or *automated theorem proving* (e.g. inference engines in ontology-based reasoning).

Many types of algorithms have been proposed for solving satisfiability problems: from naive table-of-truth methods, to optimized variations of algorithms such as *rule systems*, *resolution*, *semantic tableaux*, *backtrack search* or *conflict-driven clause learning.*

In this activity we explore the basics of the ***semantic tableaux*** method, applied to *propositional logic.* This method is also used, under variations due to syntactic operators and semantic interpretation, in current reasoning systems for description-logic (ontology) inference (e.g. FaCT++, RacerPro, Pellet).

We consider the simplified case for propositional logic and explain the required extensions for predicate logic. The extensions to description logics will be taught in class in the following weeks :-)

Semantic Tableaux Method for Propositional Logic

The proposed algorithm is labelled as *semantic* because it build a *model* of satisfiability for the set of formulae (clauses) which make up the *knowledge base* of a given logical reasoning system. This means that it is a constructive method, since at the end of the algorithm one obtains an assignment of values to propositions which satisfy each formula. Alternatively, if no such assignment can be found, then the algorithm is a complete one and will be able to say that the knowledge base is inconsistent.

The *Semantic Tableaux Method* constructs a *tree* of inference having specific conditions for building *trunks* and *branches.* The *leaves* of the tree are nodes, where, in the case of propositional logic, only *individual truth statements (propositions)* remain and the tree can no longer be expanded.

The basic idea is to incrementally build the model by looking at a formula, by decomposing it in a top/down fashion. The procedure exhaustively looks at all the possibilities, so that it can eventually prove that no model could be found for unsatisfiable formulae.

Satisfiability If we have to check whether a set of statements is *satisfiable* we build the tableaux and check whether there is *at least one branch* that does not *close.* If it exists, this branch gives us the interpretation that satisfies the statements. Otherwise the set is *unsatisfiable*.

Let us now define the rules by which a tableaux is constructed and give examples that showcase what it means for a branch to be *open* or *closed*.

Tableaux Construction Rules

Starting from a given formula, or set of formulae, the tableaux method constructs *trunks* or *branches* depending on the logical connectors that bind formulae together. The rules are given in the following figure:

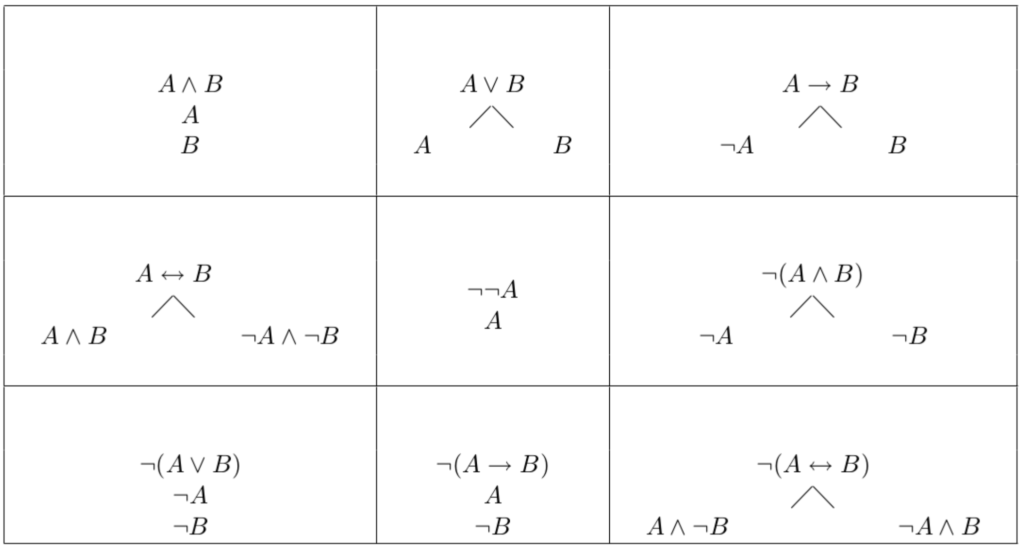
$A \wedge B$ / A / B
$A \vee B$ / A | B
$A \to B$ / $\neg A$ | B
$A \leftrightarrow B$ / $A \wedge B$ | $\neg A \wedge \neg B$
$\neg\neg A$ / A
$\neg(A \wedge B)$ / $\neg A$ | $\neg B$
$\neg(A \vee B)$ / $\neg A$ / $\neg B$
$\neg(A \to B)$ / A / $\neg B$
$\neg(A \leftrightarrow B)$ / $A \wedge \neg B$ | $\neg A \wedge B$

Figure 1: Tableaux method rules for propositional logic

Rule application heuristics When applying the rules to a node of the semantic tableaux containing formulae, precedence should be given to those rules that build *trunks* (e.g. the one for $A \wedge B$). Next, rules that introduce a negation inside parentheses (e.g. $\neg(A \wedge B), \neg(A \vee B)$). Finally, apply the rules which create branching (e.g. $A \vee B, \neg(A \wedge B)$).

Examples

Let us give two short examples of the application of the Tableaux method for propositional logic.

P1 Consider the case of wanting to prove the validity of the transitivity of logical consequence: e.g. from *premises* $p \to q$ and $q \to r$, one derives the conclusion that $p \to r$.

We can reduce proof to a matter of *satisfiability*, by applying the reduction-to-the-absurd principle. This means that we try to prove that the knowledge base consisting of:

$$\{p \to q, q \to r, \neg(p \to r)\}$$

i.e. the 2 premises and the *negated conclusion* is *insatisfiable*.

The construction of the Tableau follows as in figure . Explanations to be given by TA in class :-).

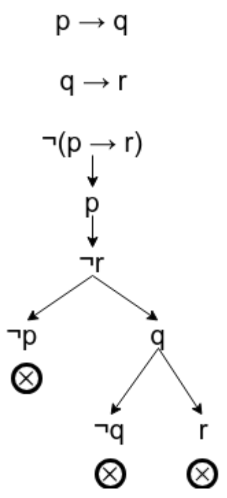

Figure 2: Applying tableau method to the knowledge base of P1. Observe that all branches are *closed*.

P2 Consider the following formula:

$$\neg((A_1 \to A_2) \land (A_3 \lor \neg(A_3 \lor \neg A_1)))$$

The task is to determine if it is satisfiable. Applying the Tableau Method rules the following tree ensues (cf. Fig):

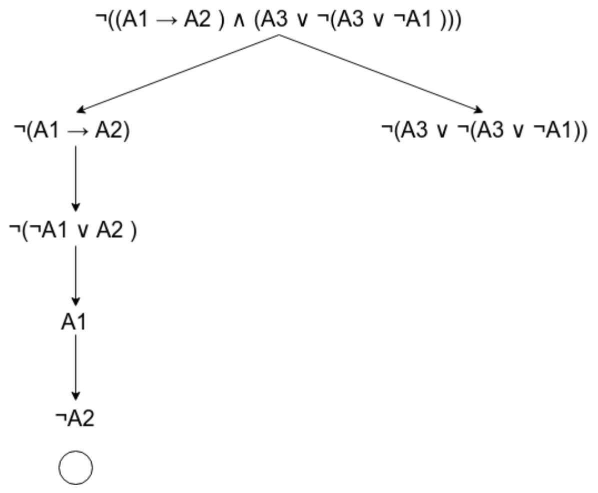

Figure 3: Applying tableau method to the knowledge base of P2. Observe that one branch is finished and *open*, which means one can find a satisfiability model for the entire formula just by having $A_1 = T$ and $A_2 = F$

Notice that in the case of this exercise, we only need expand the left branch until we finish the expansion (having just individual statements in the leaf nodes). Since there is no conflict along the branch (no contradictory statements), the branch remains open and the tree construction can stop, because it means we have found a model which satisfies the original formula.

Tasks

Let there be the following statements in natural language:

- Alfred or Beth or both will go to the party. ($A \lor B$)
- If Beth goes to the party then Clare will go unless Daniel goes. ($B \to (\neg D \leftrightarrow C)$)
- Daniel will go if Alfred does not go. ($\neg A \leftrightarrow D$)
- Therefore, Clare will go to the party. (C)

Task 1 Use the tableau method to prove that the above set of statements is consistent (i.e. there is a model of interpretation for propositions A, B, C and D such that all statements hold true). Write the ensuing tree on paper.

Task 2 Use the representation provided in the support code - or come up with your own in a programming language of choice :-) - to implement the tableau method for satisfiability proving. Your algorithm takes as input **a set of propositional logic formulae** and must apply the Tableaux Construction Rules. The algorithm should stop as soon as one branch of the tree is finished and open. It must then return the truth value assignments for the propositions in that branch. If all branches are *closed*, the algorithm must output that the knowledge base is *unsatisfiable*.

Task 3 Consider a simplified english rendering of the original natural language statements (that makes them easier to transcribe to propositional logic).

- Alfred go to party OR Beth go to party. ($A \lor B$)
- IF Beth go to party THEN Clare go to party IFF NOT Daniel go to party. ($B \to (\neg D \leftrightarrow C)$)
- Daniel go to party IFF NOT Alfred go to party. ($\neg A \leftrightarrow D$)
- Clare go to party. (C)

Convert the statements in the following problem set to propositional logic clauses and determine their satisfiability via the Semantic Tableau Method you have implemented.

Problem Set

P1

1. IF weather is sunny THEN park is beautiful.
2. IF park is beautiful THEN people walk dogs.
3. IF people walk dogs THEN park full of dogs.
4. Weather is sunny AND NOT park full of dogs.

P2

1. IF Paul likes apples THEN Paul buys apples.
2. IF Wendy likes apples THEN Wendy buys apples.
3. IF Susan likes apples THEN Susan buys apples.
4. IF Wendy buys apples THEN basket has apples.
5. Paul likes apples OR Wendy likes apples OR Susan likes apples.
6. Basket has apples.

P3

1. IF Yueh is blackmailed THEN Yueh pacts with Harkonen.
2. IF Yueh pacts with Harkonen THEN NOT Yueh is loyal.
3. Duke Atreides rewards Yueh IFF Yueh is loyal.
4. Yueh is blackmailed AND Duke Atreides rewards Yueh.

P4

1. Alfred takes car OR Alfred takes bus.
2. Car goes work IFF car has gas.
3. Alfred goes work IFF Alfred takes car AND car goes work.
4. Alfred goes work IFF Alfred takes bus AND bus goes work.
5. Alfred takes bus IFF NOT car goes work.
6. Alfred takes car IFF car has gas.
7. Bus goes work IFF NOT city has traffic.
8. NOT car has gas.
9. City has traffic.
10. Alfred goes work.