Semantic Tableaux proof method for predicate logic *

Graeme Taylor

January 16, 2004

The Semantic Tableaux method is a formalisation of proof in predicate logic which works by backward chaining: that is, proof by contradiction rather than deduction. In seeking to establish whether a sentence A is a logical consequence of a set of axioms Γ we instead assume that Γ holds yet A is false- then proceed systematically by a version of the tree method for logical decomposition and hope to find our derivation blocked by contradiction. These contradictions lead us to conclude the original assertion (Γ true but A false) is itself contradictory and hence if Γ is true A must also be (A rigorous illustration of completeness and correctness is given later in this writeup; but an appreciation of the method is required for that). Of course, we may fail to find contradiction in all cases- if we are left with an open branch, then that branch describes an interpretation in which Γ can hold but A is false, so A is not a logical consequence (which requires A true under all Interpretations).

A Semantic Tableau consists of a tree with nodes; using the semantic tableau rules we can extend a tree by adding nodes- in some instances these rules will introduce branches but for very simple proofs there may only be a single branch. The aim is to achieve all branches closed by contradiction- that is, for the branch to contain nodes which both assert and deny some formula B. Our initial nodes are the axioms Γ and the negation of A. It is clearer if nodes are given a number and are then referenced when used to assert new nodes, but this is not necessary if you are confident with the process. Note that you can wilfully misuse the system to never finish at a contradiction because you can uselessly generate new skolem constants forever and hence avoid applying a rule that would close the branch; we will describe the method as having been applied systematically if any operation which is possible on an open branch is eventually done (unless it has already been closed by an earlier application of a rule). Here then are the rules:

*First appeared on Everything2, at *http://www.everything2.com/index.pl?node_id=1513377*

Semantic Tableau rules

- If NOT A is denied, add a node asserting A
- If (A AND B) is asserted, add two nodes, one asserting A and the other asserting B
- If (A OR B) is denied, add two nodes, one denying A and the other denying B
- If (A IMPLIES B) is denied, add nodes to assert A and deny B
  This is the only way in which A IMPLIES B can be false
- If (A AND B) is denied, form two branches, one with a node denying A and the other with a node denying B
- If (A OR B) is asserted, form two branches, asserting A on one and asserting B on the other
- If (A IMPLIES B) is asserted, form two branches and add a node denying A on one and a node asserting B on the other
  This is a special case of OR, since A IMPLIES B is (NOT A) OR B; having this as a rule saves time
- If (A IFF B) is asserted, form two branches. On one of them add two nodes to deny each of A and B; on the other branch add two nodes to assert A and B
- If (A IFF B) is denied, form two branches, but now assert A and deny B on one; for the other we assert B and deny A
- If (FORALL X) A(X) is denied invent a new skolem constant c and deny A(c).
  There must be at least one counterexample for NOT FORALL to hold, we'll call it c. This rule can be used repeatedly- but you must invent a new skolem constant rather than use one previously generated by such a rule.
- If (EXISTS X) A(X) is asserted, again invent a skolem constant c and assert A(c), taking care as with the previous rule to avoid using an existing symbol
- If (FORALL X) A(X) is asserted, and t is any variable free term, assert A(t)
  This may be done for any number of variable free terms t, such as those generated by the above two rules.
- If (EXISTS X) A(X) is denied, and t is any variable free term, deny A(t)

Some advice on the order of rule application

Generally we want to avoid repetition of work on multiple branches. As rules other than the substitution rules (the last two listed above) need only be applied once on a branch, it makes sense to use those first; and to maximise the power of closing a branch by contradiction, we will use a rule that does not increase the number of branches in preference to one that does. Hence a general approach would be to use, in descending order of preference:

- Propositional rules that do not add to the number of branches
- Quantifier rules that create new skolem constants on the branch
- Propositional rules that increase the number of branches
- Substitution rules

Some worked examples

First, let us try to prove something which is true, namely that if (FORALL X)(A(X) $\rightarrow$ B(X)) and (EXISTS X)(A(X)) then we can find an X such that B(X) is true (the same X that made A true).

So we have two axioms Γ_1 = (FORALL X)(A(X) $\rightarrow$ B(X)) and Γ_2 = (EXISTS X)(A(X)) . We want to ascertain whether A= (EXISTS X)(B(X)) is a logical consequence of these two axioms, so we start our tree by listing the axioms and the negation of A, then try to proceed to contradiction:

```
1) (FORALL X)( A(X) --> B(X)) This is Gamma1
2) (EXISTS X)(A(X))           This is Gamma2
3) NOT (EXISTS X)(A(X))       This is NOT A
4) A(t)                       We create a new skolem constant by applying the rule to line 2
5) A(t) --> B(t)              Substitution of the constant t into line 1
6) NOT B(t)                   Substitution of the constant t into line 3
         /     \
        /       \

7) NOT A(t)      7) B(t)      We branch by applying a rule to line 5
```

Our left-hand branch is closed by contradiction as 4 asserts A(t) whilst 7 denys it; and our right-hand branch is closed by contradiction between 6 and 7 regarding B(t). Hence we have shown that Γ and NOT A leads inexorably to contradiction; we must conclude that to avoid such a fate, when Γ is true A must also be. So A is a logical consequence of Γ. ■

For an example where the method fails to obtain a contradiction (that is, there is a case in which Γ holds but A does not), consider the axiom $P \rightarrow P$ and let the sentence we wish to prove as a logical consequence be simply P. Then applying the method gives:

```
      1)   P --> P
      2)   NOT P
     /   \
    /     \
3) NOT P  3) P       from 1
```

The right-hand branch is closed by contradiction; we have both asserted and denied P. But on the left there is no contradiction, and the process has terminated systematically (i.e. we have no more rules to apply). The left-hand branch therefore describes a counterexample; namely any interpretation in which P is false. We can verify that this is correct ourselves- if P is false then $P \rightarrow P$ evaluates as true, but P evaluates as false, so P's truth is not a logical consequence of that of $P \rightarrow P$ in this case.

How powerful is the Semantic Tableau system?

So we have a system of rules which, it is claimed, will enable logical consequence to be determined. But can we be sure of the validity of the semantic tableau system for any set of axioms and sentences? It turns out that we can, by applying Gödel's theorems of completeness and correctness. There are two conditions we wish to show are true:

- Correctness- If the ST system claims that A is a logical consequence of a set of axioms, we want that to really be the case.
- Completeness- If A is a logical consequence of a set of axioms, we want the ST method to indicate this.

Writing |= to mean "is a logical consequence of" and |- to mean "can be proved in the Semantic Tableau system" these conditions become

$$\Gamma \mid= A \;\Leftrightarrow\; \Gamma \mid\!- A$$

Note that attempting to prove completeness is not at odds with Gödel's Incompleteness Theorem. In brief, that result can be stated as "there are true statements in the standard interpretation of first order arithmetic which cannot be proved from the axioms of first order arithmetic". The difference with the semantic tableau system is that we are looking for logical consequences, that is sentences that hold in all possible interpretations that include the axioms. So if a statement is sometimes

true, and sometimes not (As with the unprovably true statements in first order arithmetic, which are false statements in other first order languages) then the ST method cannot be used to prove it (as it isn't always true!).

Correctness

Suppose we have a satisfiable tableau T_0 -that is, there is an interpretation I and a branch α on the tableau such that all the sentences asserted on α are true in I, and all the sentences denied on α are false in I. Then a single application of the Semantic Tableau rules $T_0 \Rightarrow T_1$ gives a tableau T_1 which is also satisfiable. *(To see this, consider that applying a rule either alters α or it doesn't. If if didn't, we retain our satisfiable branch. If α has been changed, then the nodes added are logical consequences of earlier ones on α and so still hold under I; if we introduced branches at least one of them was true in I.*

Now, we can extend this lemma to a satisfaction theorem- if a single application takes a satisfiable tableau to a satisfiable one, then any number of such applications also preserves satisfiability (simply induct on the number of rules applied).

Now suppose we have been applying rules to a tableau T_0 which asserted Γ and NOT A; and obtained a tableau T_c which is closed on all branches by contradiction. This constitutes a proof in the ST system, i.e. we are in the position Γ |- A.

T_c cannot be satisfiable, as it terminates in contradiction. Hence T_0 was not satisfiable (as if it was, then T_c would have to be by our satisfaction theorem, and we know T_c isn't.)

So T_0 is not satisfiable, meaning there is no interpretation in which it is valid. Hence in all possible interpretations that have Γ we can't have NOT A as this would make T_0 satisfiable by that interpretation. But if we can't have NOT A, we'll have to take A. So whenever Γ holds, A holds. This is precisely $\Gamma \models$ A and so we have correctness for the ST system.

Completeness

We will prove this by contraposition - that any sentence which cannot be proved in the ST method from a set of axioms is not a logical consequence of those axioms, and thus any sentence which is a logical consequence must be provable.

So suppose A cannot be proved from Γ in the ST method. This means that starting with T_0, which asserts Γ and denies A, there is no closed T_c such that $T_0 \Rightarrow^* T_c$ (i.e., we cannot achieve a tableau closed by contradiction through any number of rule applications). Instead, we have a progression of tableau $T_0 \Rightarrow T_1 \Rightarrow T_2 \Rightarrow \ldots$

Let

$$T_w = \lim_{i \to \infty} T_i$$

(Note that derivations that don't end in contradiction can be of infinite length as described earlier). Then T_w has at least one open branch α ; this branch describes an interpretation I in which T_w is satisfied. But T_0 is a subset of any branch of T_w, and hence I is an interpretation in which T_0 is satisfied. But T_0 asserts Γ and denies A. So we have an interpretation in which A does not follow from Γ and thus we do not have $\Gamma \models A$.

So NOT $\Gamma \vdash A$ implies NOT $\Gamma \models A$. So by contraposition we have $\Gamma \models A$ implies $\Gamma \vdash A$ which is the requirement for completeness. ■

Sources and reference material

This work adapted from my own revision notes, lecture notes and the coursebook from the *CM20019-Formal Systems, Logic and Semantics* module offered by the computer science department of the University of Bath, England.