Consistency and Completeness of Tableau
Consistency

• Every term proveable by the tableau method is a tautology.

• Build the tree for ~p, show that every branch is closed

• Then the starting term p is a tautology
Build a tableau by the rules

- If it is closed, then it must be a tautology

```haskell
tabTree [] tree = tree
tabTree (x:xs) tree =
    case discrim x of
        Lit p -> tabTree xs tree
        Alpha a b -> tabTree (a:b:xs)
            (extendTree (double a b) tree)
        Beta a b -> extendTree
            (Branch
                (tabTree (a:xs) (single a))
                (tabTree (b:xs) (single b)))
    tree
```
Branches in a Tableau Tree

• A tableau tree has a number of branches
• Let \( v \) be an assignment to all the variables mentioned anywhere in the tree.
• A branch is defined to be True under \( v \), if every term on the branch is True under \( v \).
• A tableau is true under \( v \), if some branch of its tree is true under \( v \).
Property of algorithm

- Note that in every case, the tree grows by extending the existing tree
- -- invariant: elements of the list are in
- -- the tree but not yet "used"

```haskell
tabTree [] tree = tree

tabTree (x:xs) tree =
  case discrim x of
    Lit p -> tabTree xs tree
    Alpha a b -> tabTree (a:b:xs)
      (extendTree (double a b) tree)
    Beta a b -> extendTree
      (Branch
        (tabTree (a:xs) (single a))
        (tabTree (b:xs) (single b)))
  tree
```
Strategy

• Show that if a tree $T$ is true, and it is extended by the rules of the algorithm, then the new tree is true too!

• Recall a tree is only extended by examining some node already in the tree.

• Thus that node must already be true!
Where is the tree extended?

• Recall the tree is true, so at least one of its paths is true, call this path A

• The tree is extended along some path, call it B.
  – If B is distinct from A, then the new node does not affect path A, and so the whole Tree is still True.
  – If B is the same path as A then we must consider the two cases that are possible. The Alpha and Beta cases
We know \((\text{Alpha } x \ y)\) is True, so by prop1 both \(x\) and \(y\) are true, so the new path on the extended tree is also True.
We know \((\text{Beta } x \ y)\) is True, so by prop1, either \(x\) or \(y\) are true, There are 2 new paths on the extended tree. One of which must be true so the tree remains True.
By induction on the number of steps

• If the initial tree node is True, then the tree returned will also be True.
• A closed tableau cannot be true (since every path has at least one conjugate pair), thus the original root node must be unsatisfiable.
• But the original node was \( \sim p \)
  
  \[
  \text{solveT } p =
  \]
  
  \[
  (\text{tabTree } \text{[NotP } p])
  \]
  
  \[
  \text{(single } (\text{NotP } p))
  \]
  
• So \( p \) must be a tautology.
Completeness

• Here we must show that every tautology has a closed tableau tree
• And that the algorithm will find it.
• This is about being sure we have enough rules to complete a closed tableau for every kind of formula.

• If X is a tautology, will every complete tableau for \( \sim X \) close?
Definition of complete path

• Consider a path in a tableau:  \( P = p_1 \ p_2 \ldots \ p_n \)
• We say \( P \) is complete, if for every \( p_i \),
  – if \( p_i \) is an (Alpha x y) then both x and y are in the path
  – If \( p_i \) is a (Beta x y), then either x is in \( P \) or y is in \( P \)
• \textit{completed}, if every path is either closed or complete
• The algorithm always constructs complete paths
**Strategy**

- Let $T$ be a tableau.
- If $T$ is an open completed Tableau
  - i.e. $T$ is completed, but at least one path is still open
- Then the root (or origin) of $T$ is satisfiable. I.e. we can extend the open path (in fact we can extend all the open paths) to keep the root satisfiable.
Theorem

- Let $P$ be an open complete path in $T$
- Let $S$ be the set of terms in the path $P$
- The set $S$ satisfies the 3 following conditions for every $(\text{Alpha, Beta term})$ in $S$.
  - No signed variable and its conjugate are in $S$
  - If $(\text{Alpha } x \ y)$ is in $S$, then $x$ in $S$ and $y$ in $S$
  - If $(\text{Beta } x \ y)$ is in $S$, then either $X$ in $S$ or $y$ in $S$
Hintikka Sets

• Any set obeying the 3 rules
  – No signed variable and its conjugate are in S
  – If (Alpha x y) in S, then x in S and y in S
  – If (Beta x y) is in S, then either X in S or y in S
• Is called a Hintikka set.
Hintikka’s lemma

• Let $S$ be a Hintikka set, then there exists and interpretation (assignment to its variables) in which every set in $S$ is True.

• Start by constructing the following assignment for every variable $v$ that appears in the set.
  1. If $v \in S$, then assign $v$ True
  2. If $\neg v \in S$ then assign $v$ False
  3. Otherwise give it any assignment you want (we will choose True for concreteness)
 Comments

• 1 and 2 are not inconsistent, because $S$ is a Hintikka set, and by definitions both $v$ and $\sim v$ cannot be in $S$

• We will now show that every $p$ in $S$ is true under this assignment

• We do this by induction over the structure of $p$
Case \( v \) or \( \sim v \)

- If the term is a variable or a negated variable then it is clearly True, since we designed the assignment \( v \) to be True in this case.
Other cases

• If \( p \) is \( \text{ImpliesP} \), \( \text{AndP} \), or \( \text{OrP} \), or a Negation of one of these, then it is either an \( (\text{Alpha} \ x \ y) \) or a \( (\text{Beta} \ x \ y) \)

• So by structural induction both \( x \) and \( y \) evaluate to True under the assignment \( v \)
(Alpha x y) Case

• Because S is a Hintikka set, then both x and y are in S, and by induction x and y evaluate to True under $\nu$

• So by the structure of discrim (there are three cases)
  – discrim (AndP x y) = Alpha x y
  – discrim (NotP (OrP x y)) = Alpha (NotP x) (NotP y)
  – discrim (NotP (ImpliesP x y)) = Alpha x (NotP y)

• (Alpha x y) must also evaluate to True by the definition of Hintikka set.
(Beta x y) Case

• Because S is a Hintikka set, then either x or y are in S, and by induction the one in S must evaluate to True under v

• So by the structure of discrim (there are three cases)
  – discrim (OrP x y) = Beta x y
  – discrim (ImpliesP x y) = Beta (NotP x) y
  – discrim (NotP (AndP x y)) = Beta (NotP x) (NotP y)

• (Beta x y) must also evaluate to True by the definition of Hintikka set.
Completeness Theorem

• If X is a tautology then every tableau rooted with \( \sim X \) must close.

• Suppose T is a complete tableau rooted at \( \sim X \).
  • If T is open, then by Hinitkka’s lemma we can find an assignment where \( \sim X \) is satisfiable, that means X cannot be a tautology since there is an assignment that makes \( \sim X \) True.
  • Thus if X is a tautology, then the tableau for X must close.