Semantic Tableaux Methods for Modal Logics That Include the B(rowerische) and G(each) Axioms

Francis Jeffry Pelletier
Depts. of Computing Science & Philosophy
Univ. Alberta
Edmonton, Alberta
Canada T6G 2E1

Introduction: Semantic tableaux are a method for determining validity of arguments in a certain class of logics. The method is perhaps not so well-known to the automated theorem proving community as resolution, but it has long been present in the logic community.[1] As its name suggests, the method uses semantic facts about the language in an attempt to generate a counter-model for the argument at hand (or to show that there is no such counter-model and that the argument is therefore valid). In this, the method bears a strong similarity to Resolution, but it does not require any conversion to a normal form. The method is also different in being an *analytic* system: each non-initial formula in any proof is a subformula of some earlier formula in the proof. Natural deduction is not analytic in this sense, nor is any Resolution method other than Unit Resolution.

There are various different formats for stating the method of semantic tableaux. Participants in this AAAI Symposium are doubtless familiar with the form given by Fitting (1988). The method he uses is quite similar to the method I wish to employ, and there should be no difficulty in transforming the one into the other. In this note, I am interested in extending my method to a certain class of modal logics. In this extension we will only consider propositional logics – a fact that lends some considerable simplification to the description of the tableaux method. (By not needing to describe quantification rules and their difficult integration into modal logics. See Garson 1988 for a survey of the difficulties).

I will first lay out the tableaux method that I prefer, restricting my attention to classical propositional logic. I then show how it is extended to a certain class of modal logics, the normal modal logics. The idea of such an extension is not particularly new; it is already mentioned in Fitting (1983, 1988), although naturally I believe that the particular modifications I make to the basic method makes the extension considerably easier to understand. After developing a method for the basic one of these logics I turn my attention to logics which contain the B(rowerische) axiom. Of these logics, Fitting (1983:192; 1988:203) says that the style of tableaux he presents does not lend itself to logics where the accessibility relation involves symmetry. And it is implied that it would be very difficult to construct reasonable tableaux methods for such logics. I show that the alteration I make avoids this difficulty and allows for a semi-analytic tableaux system. After this I consider modifications to the basic system so as to yield a tableaux method suitable for systems containing the G(each) axiom. This axiom is unusual in that it involves embedded modal operators in both its antecedent and its consequent, and this in turn raises interesting challenges for tableaux systems. Once again we see that a semi-analytic tableaux system will work.

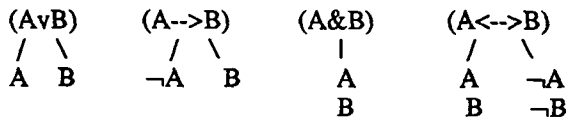
Classical Propositional Semantic Tableaux: Semantic tableaux, when done in the method I prefer, are in fact trees; and the method of making a tableau is in fact a method of tree construction. Like Resolution, it is a refutation method; and therefore it starts with a "negated conclusion" form of an argument to be tested. (Unlike Resolution, there is no further pre-processing of formulas, such as conversion to a normal form.) The root of the tree contains all the premises and the negation of the conclusion. The construction of the rest of the tree is governed by "branching rules", which operate (by decomposition) on formulas already in the tree.

For the propositional logic, formulas are defined in the usual way, using sentence letters (A,B,C....) and the connectives $\neg$, &, v, $\rightarrow$, $\leftrightarrow$. For describing the method, it is convenient to divide formulas into four classes: (i) literals (sentence letters and their negations), (ii) formulas which are double negations (i.e., start with two $\neg$-signs), (iii) formulas which have some binary connective (&,v,$\rightarrow$,$\leftrightarrow$) as their main connective, and (iv) formulas which are negations of type (iii) formulas. To each formula which is not a literal, exactly one of the following nine branching rules is applicable:

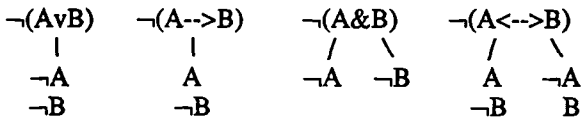
[1] The method goes back explicitly to Beth (1952,1958), and has been popularized by such elementary logic textbooks as Smullyan (1968), Jeffrey (1968), Bonevac (1987), Bergmann *et al* (1980). As has been noted by various researchers (e.g. Fitting 1983:4-6, Fitting 1988: 191), there is a strong analogy between the tableaux method and Gentzen's (1934/5) consecution method. (Although most people would wish to maintain a distinction between the two methods.)

[for type (ii) formulas]:

[for type (iii) formulas]:



[for type (iv) formulas]:

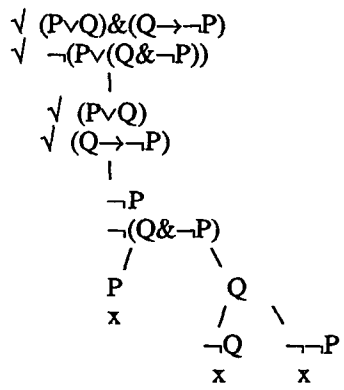


Since the construction of a tableau proof in this system is really the construction of a tree, to define a proof formally requires defining various notions relevant to trees -- such as 'dominates', 'branch', 'leaf', 'immediately dominates', and the like. For the present audience it is best just to allow one's background knowledge of trees come into play; and I will therefore just presuppose such notions. One terminological point: If a formula and its negation are on a branch, we say that the branch is *closed*; otherwise it is *open*.

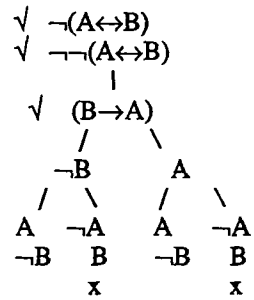
One begins the tree construction with the set of premises and negated conclusion at the root. One may close a branch as soon as one notices that it contains complementary formulas (i.e., any formula and also its negation on the same branch). Otherwise one chooses a complex formula (of type (ii), (iii), or (iv) above) from some node on an open branch and applies the relevant branching rule to it, placing the result at the bottom of all open branches dominated by that node. (Done by extending each such open branch in the manner indicated by the rule.) When this is done, the initial formula is "checked off" (by placing a '√' mark next to it), indicating that it needs to be considered no further. This lends itself to the following algorithm: Continue the following two steps until either all branches are closed or else there are no un-checked complex formulas in the tree. (i) determine whether any just-altered branches close; if so, close the branch by placing an 'x' at the bottom of the branch. (ii) choose an un-checked complex formula from an open branch, apply the relevant branching rule to it, and check off the formula.

The two halting conditions in the above algorithm correspond to whether the initial argument is valid or is invalid, respectively. A proof of the completeness and soundness of the algorithm depends on König's Lemma, and can be found in Jeffrey (1968:56). Here are two simple examples of a proof/disproof in this system.

Example 1: Given (P∨Q)&(P→¬Q), does it follow that (P∨(Q&¬P)) ?



Example 2: Given ¬(A↔B), does ¬(B→A) follow?



The first example is valid, as evidenced by all the branches closing, while the second example is invalid, as evidenced by branches remaining open yet all complex formulas being checked off. It is pretty clear in these examples which branches were the result of working with which previously-appearing formulas; other examples may not be so clear, and so some textbooks employ a bookkeeping method to record this information. It should be noted that a branching rule operates on a formula at a node in the tree, and the result of applying the rule is to extend every (open) branch in the tree that the node dominates by adding new node(s) as called for by the particular rule being used. The formula to which the rule has been applied is then checked off, to indicate that it never need be considered again. (Propositional tableaux systems are therefore decision procedures: they always terminate when all complex formulas have been checked off or when the tree is closed). In the classical propositional calculus it does not matter, logically speaking, in which order formulas are chosen in the proof construction algorithm (although it certainly matters for efficiency of tree construction). This aspect will change in the modal propositional calculus. It should also be noted that the this method of proof is *analytic*: no formula appears in the tree unless it is a subformula of some previously-appearing formula. Thus all formulas in any proof are subformulas of the initial set of negated-conclusion

formulas. In this, classical semantic tableaux stand in sharp contrast with Natural Deduction proof methods and with Resolution proof methods, which allow formulas to be generated that are not subformulas of previously-generated formulas. This aspect of classical tableaux systems will also change with the modal logics mentioned in the title of this article.

Intuitively speaking, branching a formula indicates that the formula is true if and only if at least one of these newly-created sub-branches is true (i.e., all the formulas on the new node of that branch are true). Hence, the initial root node is true exactly if at least one branch remains open -- and the initial argument is valid just in case no branch remains open. One can even read off a counterexample to an invalid argument (in terms of assignments to atomic sentences) by traversing an open branch and assigning True to (atomic) unchecked sentences which appear unnegated and False to those which appear negated.

Normal Modal Propositional Logics: Modal propositional logics are formed from classical propositional logic by adding two new (interdefinable) sentence operators: L ("necessarily") and M ("possibly"). In this section, 21 modal propositional systems are described. They form a subset of the so-called *normal modal logics*, and include most of the well-known modal systems. The technique used here to describe these systems is that of Chellas (1980). Starting with a fundamental system, K, various axioms are introduced (called such names as 'D', 'T', 'B', 'G', '4', '5') and more complex systems are named in accordance with which axioms are added to K. Thus, the modal system KD45 results from adding axioms D, 4, and 5 to system K. As it turns out, some axioms imply others, some combinations of axioms are equivalent to each other, and some combinations are known by other names. Attention will be called to these cases as they arise.

So, syntactically speaking, the various modal systems of interest are seen as being built upon K. System K can be axiomatized as follows:

1. Classical Propositional Logic (formulated with modus ponens as a rule of inference)
2. $\vdash_K (L\Phi \leftrightarrow \neg M\neg\Phi)$
3. $\vdash_K L(\Phi\rightarrow\Psi)\rightarrow(L\Phi\rightarrow L\Psi)$
4. if $\vdash_K\Phi$ then $\vdash_K L\Phi$

Typical theorems of system K are

$M(p\rightarrow q)\leftrightarrow(Lp\rightarrow Mq)$

$L(p\&q)\leftrightarrow(Lp\&Lq)$

But such familiar formulas as

$Lp\rightarrow p$

$Lp\rightarrow LLp$

$Lp\rightarrow Mp$

are not theorems of K. To build upon K, we consider the following six axioms:

D. $L\Phi\rightarrow M\Phi$

T. $L\Phi\rightarrow\Phi$

G. $ML\Phi\rightarrow LM\Phi$

B. $\Phi\rightarrow LM\Phi$

4. $L\Phi\rightarrow LL\Phi$

5. $M\Phi\rightarrow LM\Phi$

Starting with K (which adds 0 of these axioms), there are 64 different combinations of these six axioms, and so one might expect 64 different modal systems. However, there are certain implications between axioms and equivalences amongst groups of axioms. The relevant implications are:

T ==> D

B ==> G

5 ==> G

and there are the following equivalences

KB4 <==> KB5

KDB4<==>KTB4<==>KT45<==>KT5<==>KTB5

(and any other implications and equivalences entailed by these). This leaves us with the 21 modal systems pictured in Figure 1 (adapted from Chellas 1980 p.132) presented at the end of this article. The lines between systems indicate proper inclusion: all theorems of the weaker system are also theorems of the stronger system, but not conversely.

One interesting fact about these systems is that they have a semantics which can be described by a binary relation (called the "accessibility relation", R) on "normal possible worlds". The intuition is that a sentence is (not just simply true but rather) *true at a possible world*, and when the sentence has one of the modal operators as a main connective then it is true at a particular world just in case the demodalized formula is true at every (or some -- depending on the modal operator involved) accessible possible world. For example, $L\Phi$ is true at possible world w just in case Φ is true at every possible world that is accessible to w. In a normal modal logic, to say that a sentence is semantically valid is to say that it is true at every possible world in every model in which R obeys certain constraints. Obviously these definitions put a lot of weight on just what the relation R is.

It is well-known what the relevant requirements on R are for the modal systems under consideration here. In system K, R has no special requirements; therefore, a sentence Φ is valid in K (which we represent as $\models_K\Phi$) just in case for any relation R, Φ is true at every possible world (under that accessibility relation, R). (It is because there are no special requirements placed upon R that system K is "the minimal normal modal logic"). Other logics add conditions upon R, for example that it must be reflexive or that it must be connected. Each of the axioms to be considered gives rise to its own particular requirement on R, and thus a logic described as KD45 (for example) will impose upon R the requirements relevant to D and to 4 and to 5. And a formula Φ is valid in KD45 ($\models_{KD45}\Phi$) iff for any relation R which obeys the restrictions relevant to D, 4, and 5, Φ is true at every possible world (under that accessibility relation, R).

From the semantic point of view, a standard model M is a triple <W,R,P> where W is a (non-empty) set of indices ("possible worlds"), R is a binary relation on W, and P is a total function on $W \times L_A$ (where L_A is the set of atomic sentences of the language) into {true,false}. (That is, P says which atomic sentences are true/false at which possible worlds). A truth-functionally compound sentence Φ is true in the model at possible world α in just the "normal way", viz.:

if $\Phi = \neg\Psi$, then Φ is true at α iff Ψ is false at α

if $\Phi = (\Psi \& \Gamma)$, then Φ is true at α iff either Ψ is true at α or Γ is true at α

if $\Phi = (\Psi \rightarrow \Gamma)$, then Φ is true at α iff, if Ψ is true at α then Γ is true at α

and so on for any truth functional connective we happen to have. Modally compound sentences are true in the model at a possible world α in a way that makes use of the "accessibility relation", R:

if $\Phi = L\Psi$, then Φ is true at α iff Ψ is true at all β such that $R\alpha\beta$

if $\Phi = M\Psi$, then Φ is true at α iff Ψ is true at some β such that $R\alpha\beta$

A formula is *true in the model M* iff it is true at every possible world in M. A formula is *logically true in M* iff it is true at every possible world in each model <W,R,P'>, where P' might differ from P as to which atomic sentences are true at which possible worlds. A formula is *valid* iff it is logically true in every model <W,R',P>. That is, a sentence is valid iff for all R' and for all P' the sentence is true at every world in each <W,R',P'>.

The just-defined notion characterizes validity in system K. Each other system defines *validity* with respect to some restricted set of R'-models. For example, these R' relations might be required to be reflexive, or they might be required to be serial, etc. As it turns out, each of the axioms which we have considered adding to system K gives rise to its own requirement on what kind of relation R must be. The requirements are stated:

(d) [seriality] $(\forall x)(x \in W \rightarrow (\exists y)(y \in W \,\&\, Rxy))$

(t) [reflexivity] $(\forall x)(x \in W \rightarrow Rxx)$

(b) [symmetry] $(\forall x)(\forall y)(x,y \in W \rightarrow (Rxy \rightarrow Ryx))$

(g) [incestuality] $(\forall x)(\forall y)(\forall z)(x,y,z \in W \rightarrow (Rxy \,\&\, Rxz \rightarrow (\exists w)(w \in W \,\&\, Ryw \,\&\, Rzw)))$

(4) [transitivity] $(\forall x)(\forall y)(\forall z)(x,y,z \in W \rightarrow (Rxy \,\&\, Ryz \rightarrow Rxz))$

(5) [euclidean] $(\forall x)(\forall y)(\forall z)(x,y,z \in W \rightarrow (Rxy \,\&\, Rxz \rightarrow Ryz))$

In the modal system KD, for example, a formula Φ is valid just in case it is logically true in every model in which R obeys (d), that is, in which R is serial (= connected). Φ is valid in KD45 just in case it is logically true in every model in which R obeys (d), (4), and (5), that is, in which R is serial, transitive, and euclidean.

It can be noted that, just as in the case of the axioms, certain conditions imply others and various combinations are equivalent. In particular, (t) implies (d), (b) implies (g), and (5) implies (g). Furthermore, the combination (b) and (4) is equivalent to the combination (b) and (5). And the following combinations are all equivalent to one another: of (d), (b), (4); of (d), (b), (5); of (t), (4), (5); of (t), (b), (4); and of (t), (b), (5).

The material of the preceding paragraphs can be summarized by the chart in Fig. 2 (adapted from Chellas 1980, p.164) given at the end of this article. Let us abbreviate the notion of Φ being valid, with restrictions X placed upon R, as $\models_X \Phi$. Then Fig. 2 reports completeness and soundness theorems for all the modal logics we have discussed, saying

$\vdash_X \Phi$ iff $\models_X \Phi$

where X is come collection of 'K' plus any of the letters 'D', 'G', 'B', 'T', '4', '5' -- thus forming one of the syntactic (axiomatic) systems mentioned earlier -- and x is the corresponding collection chosen from amongst 'd', 'g', 'b', 't', '4', '5' given immediately above. These soundness and completeness theorems assure us that the results of any correct proof procedure for a normal modal system X can be mirrored by the results of any correct semantic evaluation procedure. And since each of these systems has the finite model property, it follows that there is a decision procedure for each of the logics.

Tableaux Methods for Propositional Modal Logics: In this section we give tableaux methods for any normal modal system containing B and G. Our strategy will be to give a method suitable for system K, and then to explain what new rules or modifications are called for by each of these axioms. Hence to produce a tableaux argument for KBG, for example, one proceeds as with system K but adds on the modifications relevant to (b) and (g). (The same method can be used to give tableaux methods for all the 21 systems, but that more general method is saved for a different occasion).

The rules considered above for classical logic are just that: classical. This means that they are relevant to any one world at a time, but are inapplicable when considering more than one world. For example, were formula A to be true in one world and formula $\neg$A to be true in another (even if it be an accessible world), that does not allow us to assert that any branch is or isn't closed. This suggests that we divide our rules into different sorts:

Type 1 rules ("World-Bound Rules"): This sort of rule contains the classical rules we have already introduced– rules that operate only *within* a specified possible world. In addition to the (classical) world bound rules we have already come across, we have two more new ones that involve the modal operators. These new Type 1 rules are common to all normal modal systems. After applying the rule, we check off the initial formula.

$\neg LA$	$\neg MA$
|	|
$M\neg A$	$L\neg A$

Type 2 Rules ("World-Creating Rules"): This type of rule is distinctive and new to modal logics. When evaluating a modalized formula, the semantic rules given above direct us to determine whether a certain subformula of the modalized formula is true in some other possible world. Type 2 rules detect when this situation occurs and directs us how to extend the semantic tree so as to encompass new worlds, and how these new worlds are to be initialized. This conception is so that we will use a double line (rather than a single line) to indicate how the proof tree is extended when these rules are applied. The rule can be put like this:

$$\begin{array}{c} MA_1 \\ | \\ MA_2 \\ | \\ . \\ . \\ | \\ MA_n \end{array}$$

$$A_1 \qquad A_2 \qquad A_3 \quad \ldots \quad A_n$$

The way to understand this rule is: given a group of M-formulas (formulas with 'M' as their main operator) all on the same (open) branch, then simultaneously start a collection of new branches in which each new double-line branch is initiated by a different one of these formulas (without their 'M' operator).

Type 3 Rules ("Interworld Copy Rules"): Given an open branch in a world which already has been extended by a double line (so that there are new worlds created off that branch), then if this branch contains the L-formulas $LB_1, LB_2, \ldots LB_m$, copy $B_1, B_2, \ldots B_m$ into each of the new worlds that the L-formula dominates.

Application of the Type 1, 2, and 3 rules:

The order of application of the rules in the classical logic made no logical difference–any order of application would yield the same final answer (valid/invalid) as any other. This is no longer the case, because of the introduction of differing possible worlds within the tree structure. Intuitively, the addition of a double-line branch to an already-existing branch amounts to constructing a new possible world accessible from the world in which we are currently working. Given this understanding of the double-line branch, it is clear that we need some restriction on how formulas above the double line can be interact with those below the double line. For instance, were the formula $\neg A$ to be found above the double-line branch (that is, in the earlier possible world), the appearance of A below the double-line branch (in the new possible world) should not allow us to cancel that branch. Our ruling will be that *no* formula above the double-line branch can interact with any below the double-line branch, except insofar as permitted to do so by Type 3 Interworld Copy Rules.

A side effect of the our decree that no formula in one world can interact with a formula of another world (except via Type 3 Rules) is that **all Type 1 Rules must be applied within a given world before any Type 2 Rule can be applied in that world.** If we do not have this requirement then it would be possible for a contradiction to be present in a world but to go unnoticed because the relevant Type 1 rule was not applied to the complex formula which would make the contradiction manifest. Our algorithm for modal logic therefore says that we apply **only** Type 1 rules in whatever world we happen to be investigating until either (a) all branches are closed in that world (and thus that world has been declared to be impossible), or (b) there is an open branch on which all formulas are either literals, or are checked off, or else have a modal operator as their main connective. In case (b), if there are any of the last-mentioned sort of formulas, then we can apply Type 2 rules.

To apply the Type 2 rule: each M-formula within a given world creates a new world under at the end of each open branch that it dominates, and initializes that world with the de-modalized formula (i.e., the formula without the main M-connective). Having done this with a given M-formula, it can be checked off (although for ease of use of some of the Type 3 rules there should be a different style of check-mark, since the Type 3 rules sometimes need to make reference to these formulas that have been decomposed by Type 2 rules). One continues to apply the Type 2 rules to every M-formula on an open branch in the tree at the given world. **Only after there are no more applications of Type 2 rules in a given world can one proceed to apply the Type 3 rules.** (Each of the new worlds introduced by an application of the World Creating Rules is independent of the other, and each of them is (as it were) self-contained, once the Type 3 Interworld Copy Rules are applied. This means that all of the branches below the double-lines can be investigated in parallel.)

After the Type 2 Rules have been applied in a given possible world, one can apply the Type 3 rules, copying the de-modalized formula into each of the already-existing possible worlds dominated by the L-formula. When a L-formula has had each of the relevant Type 3 rules applied to it [in some of the more complex logics a formula can satisfy the preconditions of more than one Type 3 rule, and each of them needs to be applied] then it can be checked off. (Although again we employ a different type of checkmark than with the Type 1 rules). After each of the L-formulas has been dealt with by all the Type 3 rules that are relevant to it, one is in a position to move to a different possible world and deal with the formulas that are in that world by constructing another tree–this time within the new possible world. In this new world, one again starts with

the Type 1 rules, then moves on to the Type 2 rules, etc.

The closing of trees, and of worlds:

Within each world, one uses the World-Bound Rules. If all branches of the tree that represents a possible world close, then the double-line *into* that world (and hence *out* of the previous world) closes – indicating that it was not a possible world after all. If any *one* of the double-line branches leading outward from a branch within some possible world closes, then that leaf is marked as closed. (Note that this requires only that one of the many double-line branches emanating from a leaf needs to close in order to close the leaf. Intuitively, from the point of view of the initial possible world, the branch asserts the existence of such-and-so possible worlds. If it turns out that one of them does not exist, then that initial branch is impossible – regardless of what occurs in the other possible worlds.)

The goal remains as in the classical logic: determine whether the tree that represents the initial world contains an open branch. If it does, then the argument is invalid; otherwise it is valid.

Tableaux Rules for B and G: The preceding set of rules describes system K. All and only the valid arguments of K will be declared so by the above tableaux method. And since the method operates only by decomposition of formulas into subformulas (it is analytic), the algorithm will detect this in a finite time. So it is a decision procedure. As mentioned at the outset, the point of this paper is to describe how to add tableaux rules for complex rules such as B and G.[2] We will see that the resulting method is not analytic. The tableaux rules for both B and G are new Type 3 rules.

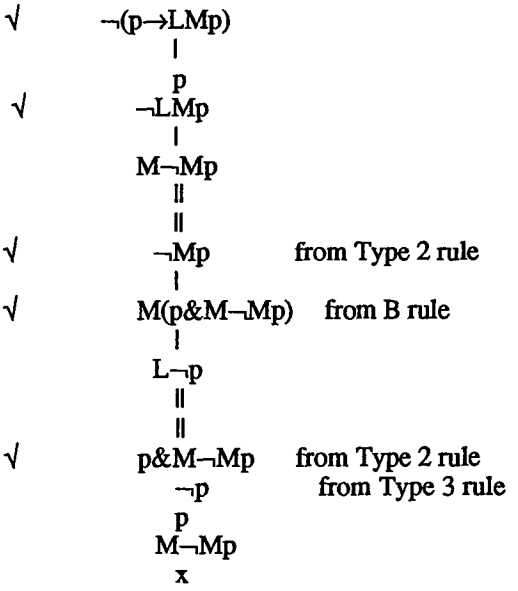
Tableaux Rule for B:

The syntactic axiom for B says that if a formula is true in a world, then it is necessary that it be possible. That is, if it is actually true then it is possibly true in every accessible world. The semantic condition on accessibility for B is symmetry. But in a tableaux setting this raises serious difficulties since it apparently implies that when we create a new world from a Type 2 rule, and then in that new world wish to create yet another new world, that the newly created world has to either be or somehow include the initial starting world...otherwise it would not be a symmetric accessibility relation.

This requirement can be satisfied in the following manner[3]: After applying all the rules as indicated for system K (thereby constructing new worlds by Type 2 rules and copying formulas into them by the Type 3 rules) also copy over the following formula into these new worlds w:

(B-formula): $M(A_1 \& A_2 \& .... \& A_n)$

where the A_i are all the formulas on the branch leading to world w that do not have the check-off marks relevant to the Type 1 rules. (So they may have been checked off by Type 2 or 3 rules, but not by Type 1 rules). This new rule will be applied to each of the worlds constructed in the earlier world. The effect of this will be to place the B-formula into each world j that is constructed from world i. But then it follows that when branches in world j are expanded, it is guaranteed that one of the worlds that can be reached will be the one initialized by the de-modalized B-formula. However, that is just world i – and thus we have simulated symmetry. As an example we will show that the B axiom is validated by this rule:



The contradiction of p and ¬p in the third world makes that world impossible, which gets "pushed back" into the second world, into the only branch of that world. This shows that the second world is impossible, and this gets "pushed back" into the only branch of the first world. Thus the first world is impossible and the

[2] A full version of this paper contains tableaux rules for each of the six axioms, and hence for the 21 different systems indicated in Figs. 1 and 2.

[3] The general theorem being appealed to here is known as the Bulldozer Theorem: any formula that is unsatisfiable in a (possible worlds) model where the accessibility relation is reflexive or symmetric or transitive is also unsatisfiable in an irreflexive, asymmetric, and non-transitive model. See Segerberg (1971: 80ff).

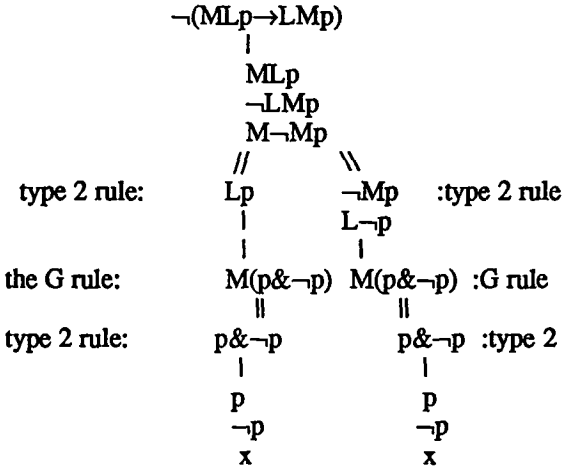
argument is valid: (p→LMp) is validated by the B tableaux method. Note that the B-formula is not necessarily a subformula of any formula in the tree up to the point of its introduction. So this tableaux method for systems containing the B axiom is not analytic. But since the B-formula can be algorithmetically constructed from the formulas in the tree, we might call the system *semi-analytic*.

Tableaux Rule for G:

The syntactic axiom for G says that if a formula is possibly necessary, then it is necessarily possible. That is, if it is necessary in any accessible world, then it must be possible in every accessible world. The semantic rule for G is incestuality: if world 1 leads to worlds 2 and 3, then there is a world 4 that both 2 and 3 lead into. The effect of this relation can be gotten by the following method. We consider any tree which has been constructed so as to have new worlds $w_1, w_2, \ldots w_n$ already in existence at the end of some branch. In these new worlds we apply all the relevant Type 1 rules. But now before applying whatever Type 2 rules might apply in these worlds, we apply this new sort of Type 3 rule:

> If LA_i, LB_i... are formulas in an open branch of wi, and LA_j, LB_j... are formulas in an open branch of w_j, then add $M(A_i\&B_i\&\ldots\&A_j\&B_j\&\ldots)$ to the end of those open branch in each of w_i and w_j. Do this for all open branches in each of the pairs of worlds of $w_1, w_2, \ldots w_n$.

Note that each pair of worlds in $w_1, w_2, \ldots w_n$ are now guaranteed to have a common descendant, and that this descendant will have true all the formulas that were necessarily true in its parents. So we have simulated incestuality. A simple example is to prove the G axiom:



The closure of at least one of the two worlds at the bottom of the tree will "push back" the impossibility of that world into the preceding world. Thus each of the previous worlds (the ones which are initialized with Lp and with ¬Mp) will get an x at their bottom. And since this is the only branch in each of these worlds, they too are each impossible, and this "pushes back" the impossibility into the initial world. (As mentioned, it is only necessary that one of the two double-line branches exiting from the initial world be impossible.) This shows that the only branch in the initial world closes, and thus that the argument is valid...the unnegated formula is a theorem in G.

Once again we note that this method is not analytic because the formulas constructed by the G rule is not necessarily subformulas of anything in the tree up to that point. Yet since the relevant formulas can be algorithmically constructed from the tree, we can call it a *semi-analytic* system of proof.

Acknowledgements: The research reported here was supported in part by the Canadian NSERC grant OPG 5525. The author is also a member of the Institute for Robotics and Intelligent Systems and wishes to acknowledge the support of the Networks of Centres of Excellence Program by NSERC, the Government of Canada, and the participation of PRECARN Associates.

References:

Bergmann, M., J. Moor, J. Nelson (1980) *The Logic Book* (McGraw-Hill).

Beth, F. (1952) *Symbolic Logic.* (North-Holland).

Beth, F. (1958) "On Machines which Prove Theorems", in Siekmann & Wrightson pp. 79-90.

Bonevac, D. (1987) *Deduction* (Mayfield Pub. Co.).

Chellas, B. (1980) *Modal Logic* (Cambridge Univ. Press).

Fitting, M. (1983) *Proof Methods for Modal and Intuitionistic Logics* (Reidel).

Fitting, M. (1988) "First Order Modal Tableaux" *Journal of Automated Reasoning* pp. 191-213.

Garson, M. (1988) "Quantified Modal Logic" in D. Gabbay & F. Guenthner *Handbook of Philosophical Logic Vol 2.*

Gentzen, G. (1934/5) "Investigations into Logical Deduction". Translation printed in M. Szabo *The Collected Papers of Gerhard Gentzen* (North-Holland) 1969, pp. 68-131.

Jeffrey, R. (1968) *Formal Logic: Its Scope and Limits* (McGraw-Hill).

Segerberg, K. (1971) *An Essay on Classical Modal Logic.* (Uppsala: Filosofiska Studier).

Siekmann,J. & Wrightson,G. (1983) *The Automation of Reasoning, Vol. 1.* Springer-Verlag.

Smullyan, R. (1968) *First Order Logic.* (Springer-Verlag).

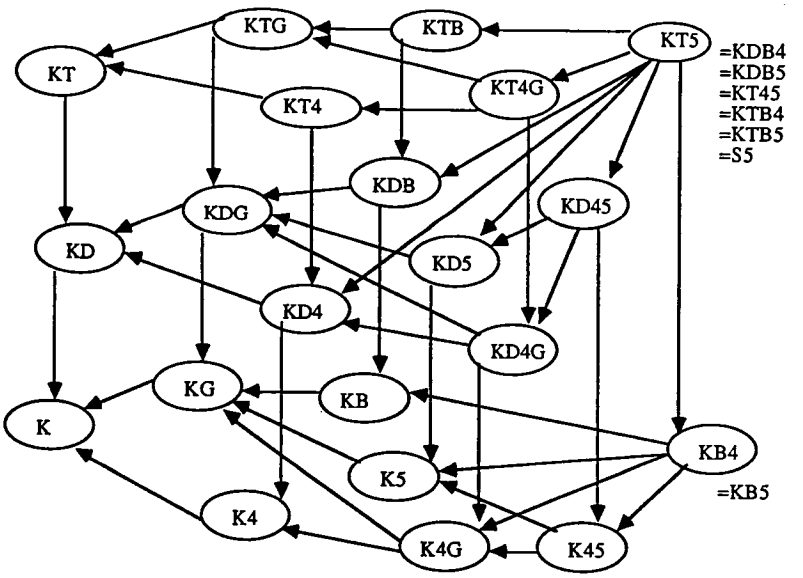


Figure 1: 21 normal modal systems. Arrows indicate proper inclusion.

	conditions on R					
system	serial	reflexive	incestual	transitive	symmetric	euclidean
K						
KD	•					
KT	(•)	•				
KG			•			
K4				•		
KB			(•)		•	
K5			(•)			•
K4G			•	•		
K45			(•)	•		•
KB4			(•)	(•)	•	•
			(•)	•	•	(•)
KDG	•		•			
KD4	•			•		
KDB	•		(•)		•	
KD5	•		(•)			•
KD4G	•		•	•		
KD45	•		(•)	•		•
KTG	(•)	•	•			
KT4	(•)	•		•		
KTB	(•)	•	(•)		•	
KT4G	(•)	•	•	•		
KT5	(•)	•	(•)	(•)	(•)	•
	(•)	•	(•)	•	•	(•)
	•	(•)	(•)	•	•	(•)
	•	(•)	(•)	(•)	•	•

TABLE 1: The semantic requirements on R for 21 normal modal systems. • = a required property, (•) = property entailed by required properties. KB4 and KT5 are shown with different (equivalent) combinations of required vs. entailed properties.