Notre Dame Journal of Formal Logic
Volume 25, Number 3, July 1984

Correcting the Tableau Procedure for $S4$

BANGS L. TAPSCOTT

The tableau procedure for normal modal logics, as given in Kripke [3], may be summarized as follows.[1] For simplicity, we assume (when appropriate) that the tableaux in question have vacant righthand (False) columns, and that the beginning formulae are jointly satisfiable. From an initial set of wffs, a tableau is generated by standard truth-functional procedures. Then, whenever it would not be superfluous to do so, the following two rules are applied: Each possibility wff, of the form Mp, is Advanced to begin a new tableau stipulated to be accessible from the old one, with the operand p as its initial formula. Necessity wffs of the form Lq are Executed by placing the operand q in each accessible tableau, and the process is continued.

The result is an array or tree of tableaux. When the procedure runs out of things to do, the tree terminates and a Kripke model for the original formulae may be read off, taking the set of tableaux in the array as the set of "possible worlds" and the access relation R between tableaux as the access relation between worlds.

$S4$, which defines R as reflexive and transitive, raises a special problem. There are $S4$-satisfiable formulae which do not lead to termination but rather yield an infinite tree. To manage these, Kripke provides the following expedient. If a tableau is a duplicate of one earlier in the construction, it is not subjected to the Advancement procedure. It thereby blocks that particular path through the tree. When all paths are blocked, the tree terminates.

Such a "blocked" tree will not yield a model by the same recipe used for other modal systems. Instead, Kripke gives us the following modification. First, assemble all duplicate tableaux into equivalence classes H, and let the set of H's represent the "possible worlds". Second, generate a derivative relation $\mathcal{R}$ among the H's, using the relation R among the tableaux, by the recipe:

$H_n \mathcal{R} H_m$ iff there are tableaux t_n and t_m in H_n and H_m, respectively, such that $t_n R t_m$.

Received April 15, 1983; revised June 13, 1983

The purpose of this paper is to correct shortcomings in the procedure just described.

When the procedure is applied to the four-part conjunction

$$A = (LM(Mp \mathbin{\&} Mq) \mathbin{\&} LM(r \mathbin{\&} M(Mp \mathbin{\&} Mq)) \\ \mathbin{\&} LM(s \mathbin{\&} M(Mp \mathbin{\&} Mq)) \mathbin{\&} Mq)$$

it yields the following array of nine tableaux, six of which are distinct. (Where an Advancement would be superfluous, it is so indicated.)

t_1	
A	
$LM(Mp \mathbin{\&} Mq)$	
$LM(r \mathbin{\&} M(Mp \mathbin{\&} Mq))$	
$LM(s \mathbin{\&} M(Mp \mathbin{\&} Mq))$	
Mq	to t_2
$M(Mp \mathbin{\&} Mq)$	(spfl--t_4)
$M(r \mathbin{\&} M(Mp \mathbin{\&} Mq))$	(spfl--t_3)
$M(s \mathbin{\&} M(Mp \mathbin{\&} Mq))$	(spfl--t_6)

t_2	
q	
$M(Mp \mathbin{\&} Mq)$	(spfl--t_4)
$M(r \mathbin{\&} M(Mp \mathbin{\&} Mq))$	to t_3
$M(s \mathbin{\&} M(Mp \mathbin{\&} Mq))$	(spfl--t_6)

t_3	
$(r \mathbin{\&} M(Mp \mathbin{\&} Mq))$	
r	
$M(Mp \mathbin{\&} Mq)$	to t_4
$M(r \mathbin{\&} M(Mp \mathbin{\&} Mq))$	(spfl--t_3)
$M(s \mathbin{\&} M(Mp \mathbin{\&} Mq))$	(spfl--t_6)

t_4	
$(Mp \mathbin{\&} Mq)$	
Mp	to t_5
Mq	to t_7
$M(Mp \mathbin{\&} Mq)$	(spfl--t_4)
$M(r \mathbin{\&} M(Mp \mathbin{\&} Mq))$	(spfl--t_8)
$M(s \mathbin{\&} M(Mp \mathbin{\&} Mq))$	(spfl--t_6)

t_5	
p	
$M(Mp \mathbin{\&} Mq)$	(spfl--t_9)
$M(r \mathbin{\&} M(Mp \mathbin{\&} Mq))$	(spfl--t_8)
$M(s \mathbin{\&} M(Mp \mathbin{\&} Mq))$	to t_6

t_6	
$(s \mathbin{\&} M(Mp \mathbin{\&} Mq))$	
s	
$M(Mp \mathbin{\&} Mq)$	to t_9
$M(r \mathbin{\&} M(Mp \mathbin{\&} Mq))$	to t_8
$M(s \mathbin{\&} M(Mp \mathbin{\&} Mq))$	(spfl--t_6)

$t_7 = t_2$
$t_8 = t_3$
$t_9 = t_4$

Representing each tableau in the array by its subscript numeral, a complete enumeration of the relation R in the array will be the following:

(1;1),	(1;2),	(1;3),	(1;4),	(1;5),	(1;6),	(1;7),	(1;8),	(1;9)
	(2;2),	(2;3),	(2;4),	(2;5),	(2;6),	(2;7),	(2;8),	(2;9)
		(3;3),	(3;4),	(3;5),	(3;6),	(3;7),	(3;8),	(3;9)
			(4;4),	(4;5),	(4;6),	(4;7),	(4;8),	(4;9)
				(5;5),	(5;6),		(5;8),	(5;9)
					(6;6),		(6;8),	(6;9)
						(7;7),	(8;8),	(9;9)

And the equivalence classes of tableaux generated by the array are:

$H_1 = \{t_1\}$
$H_2 = \{t_2, t_7\}$
$H_3 = \{t_3, t_8\}$

$H_4 = \{t_4, t_9\}$
$H_5 = \{t_5\}$
$H_6 = \{t_6\}$.

From these, by Kripke's recipe, we obtain the following derivative relation $\mathcal{R}$ over the set of equivalence classes:

$(H_1;H_1)$, $(H_1;H_2)$, $(H_1;H_3)$, $(H_1;H_4)$, $(H_1;H_5)$, $(H_1;H_6)$
$(H_2;H_2)$, $(H_2;H_3)$, $(H_2;H_4)$, $(H_2;H_5)$, $(H_2;H_6)$
$(H_3;H_3)$, $(H_3;H_4)$, $(H_3;H_5)$, $(H_3;H_6)$, and $(H_3;H_2)$ since $t_3 \ R \ t_7$
$(H_4;H_4)$, $(H_4;H_5)$, $(H_4;H_6)$, and $(H_4;H_2)$ since $t_4 \ R \ t_7$, $(H_4;H_3)$ since $t_4 \ R \ t_8$
$(H_5;H_5)$, $(H_5;H_6)$, and $(H_5;H_4)$ since $t_5 \ R \ t_9$, $(H_5;H_3)$ since $t_5 \ R \ t_8$
$(H_6;H_6)$, and $(H_6;H_3)$ since $t_6 \ R \ t_8$, $(H_6;H_4)$ since $t_6 \ R \ t_9$.

The relations R and $\mathcal{R}$, given above by enumeration, may be observed more clearly in the following diagrams. The first is a mapping of R, with the arrows understood as reflexive and transitive.

$$\begin{tikzcd}
 & & & & & & t_9 = t_4 \\
 & & & & t_5 \arrow[r] & t_6 \arrow[ur] \arrow[dr] & \\
t_1 \arrow[r] & t_2 \arrow[r] & t_3 \arrow[r] & t_4 \arrow[ur] \arrow[dr] & & & t_8 = t_3 \\
 & & & & t_7 = t_2 & &
\end{tikzcd}$$

The second is a mapping of $\mathcal{R}$, with the double-stemmed arrows understood as transitive and with reflexivity understood.

$$\begin{tikzcd}
H_1 \arrow[r, Rightarrow] & H_2 \arrow[r, Rightarrow] & H_3 \arrow[r, Rightarrow] \arrow[l, bend left, "(t_3;t_7)"] & H_4 \arrow[r, Rightarrow] \arrow[ll, bend right=50, "(t_4;t_7)"'] \arrow[l, bend left, "(t_4;t_8)"] & H_5 \arrow[r, Rightarrow] \arrow[ll, bend right=50, "(t_5;t_8)"'] \arrow[l, bend right, "(t_5;t_9)"'] & H_6 \arrow[ll, bend left=40, "(t_6;t_9)"] \arrow[lll, bend left=60, "(t_6;t_8)"]
\end{tikzcd}$$

Observe that the derivative relation $\mathcal{R}$ is not transitive. It contains $(H_6;H_3)$ *and* $(H_3;H_2)$, but not $(H_6;H_2)$; it contains $(H_6;H_4)$ and $(H_4;H_5)$, but not $(H_6;H_5)$; and it contains $(H_5;H_3)$ and $(H_3;H_2)$, but not $(H_5;H_2)$.

Since an $S4$ model requires transitivity of access between possible worlds, the procedure as presently formulated does not always yield $S4$ models for $S4$

satisfiable formulas. Thus it is not the counterproof procedure for $S4$ that it is advertised to be.

The trouble with the recipe for generating $\mathcal{R}$ is that it carries over to $\mathcal{R}$ only the transitivities already present in R. This may be seen by reformulating the recipe as follows:

> $H_n \mathcal{R} H_m$ iff for some t_j in some H_j (perhaps identical with H_n or H_m), and some t_n and t_m in H_n and H_m, respectively, $t_n R t_j$ and $t_j R t_m$.

The equivalence of this to the original recipe is easily shown. Suppose $H_n \mathcal{R} H_m$ by the old recipe. Then since R is reflexive, let $H_j = H_n$ and $t_j = t_n$. Then $H_n \mathcal{R} H_m$ by the new recipe. Suppose $H_n \mathcal{R} H_m$ by the new recipe. Since R is transitive, $t_n R t_j$ and $t_j R t_m$ entail that $t_n R t_m$; hence $H_n \mathcal{R} H_m$ by the old recipe.

The new recipe says, informally, that one equivalence class has access to another just in case a member of the one has access to a member of the other, either directly or via an intervening tableau t_j. However, as Kripke points out, the intuitive effect of forming the equivalence classes is to identify all equivalent tableaux with each other. With this "identification" in place, there appears to be no good reason for requiring a *single* member t_j of H_j to carry t_n to t_m. It will be equally appropriate to allow that $H_n \mathcal{R} H_m$ provided that t_n is R to some t_j and that some *equivalent* t_j' is R to t_m. That is, the original recipe may be replaced by

C1 $H_n \mathcal{R} H_m$ iff for some t_j and t_j' (perhaps identical with each other) in some H_j (perhaps identical with H_n or H_m), and some t_n and t_m in H_n and H_m, respectively, $t_n R t_j$ and $t_j' R t_m$.

This gives H_n transitive access through H_j to H_m, though not always via a single member of H_j.

C1 resolves the transitivity problem for the specimen formula. For example the gap between H_6 and H_2 is filled by letting $H_j = H_3$, $t_j = t_8$, and $t_j' = t_3$. The other gaps are filled similarly.

Nevertheless, C1 has no guarantee of being fully adequate to cover all cases. It is capable of filling any single gap in the transitivity chain; but there may be $S4$ satisfiable formulae leading to multiple sequential gaps under the original recipe, which cannot be filled by a single intermediary H_j but require several linked intermediary H's.[2] To accommodate this possibility, we may use the form of C1 to define a subordinate relation $\mathcal{Z}$:

D1 $H_n \mathcal{Z} H_m$ iff for some t_j and t_j' (perhaps identical with each other) in some H_j (perhaps identical with H_n or H_m), and some t_n and t_m in H_n and H_m, respectively, $t_n R t_j$ and $t_j' R t_m$.

The relation $\mathcal{R}$ may then be defined as the ancestral of $\mathcal{Z}$:

C2 $H_n \mathcal{R} H_m$ iff $H_n \mathcal{Z}^* H_m$.

Since every ancestral relation is transitive, C2 assures that $\mathcal{R}$ will contain no transitivity gaps.

However, closure of the transitivity gaps raises problems in another dimension of the tableau procedure. The method for reading off a model from a terminated array assigns truth-values as follows. A nonmodal formula is assigned T or F for a given H accordingly as it falls in the leftcolumn or the rightcolumn of the members of H. A possibility wff Mp is assigned T for H_j iff there is an H_k such that H_j $\mathcal{R}$ H_k and $V(p;H_k) = T$. And a necessity wff Lq is assigned T for H_j iff for every H_k such that H_j $\mathcal{R}$ H_k, $V(q;H_k) = T$.

Under the original recipe for constructing $\mathcal{R}$, whenever $V(Lq;t_j) = T$ then $V(Lq;H_j) = T$. That is, if q is in the leftcolumn of every tableau to which t_j has the relation R, then it belongs to the leftcolumn of every equivalence class to which H_j has the relation $\mathcal{R}$.

With transitivity gaps closed, this is no longer true. Closing the gaps has the effect of extending the access-path, thereby bringing new H's into the purview of the L-wff. It can then happen that an L-wff which ought to receive the value T will instead receive the value F since, though Executed into every accessible T, it will not have been Executed into every accessible H.

An illustration, consider the three-part conjunction

$$B = (M(Lp \,\&\, Mq) \,\&\, \mathrm{M}(Mp \,\&\, Lq) \,\&\, LMr) \ .$$

When subjected to the tableau procedure, this formula yields the following array of tableaux:

t_1		t_2		t_3	
B		$(Lp \,\&\, Mq)$		$(Mp \,\&\, Lq)$	
$M(Lp \,\&\, Mq)$	to t_2	Lp		Mp	to t_5
$M(Mp \,\&\, Lq)$	to t_3	Mq	to t_4	Lq	
LMr		Mr	(spfl--t_6)	Mr	(spfl--t_7)
Mr	(spfl--t_6)	p		q	
		t_4		t_5	
		q		p	
		p		q	
		Mr	to t_6	Mr	to t_7
		t_6		t_7	
		r		r	
		p		q	
		Mr	(spfl--t_6)	Mr	(spfl--t_7)

The relation R over this array is

$$\begin{array}{ccccccc}
(1;1), & (1;2), & (1;3), & (1;4), & (1;5), & (1;6), & (1;7) \\
 & (2;2), & & (2;4), & & (2;6) & \\
 & & (3;3), & & (3;5), & (3;6) & \\
 & & & (4;4), & & (4;6) & \\
 & & & & (5;5), & & (5;7) \\
 & & & & & (6;6), & (7;7)
\end{array}$$

or, diagrammatically:

$$\begin{tikzcd}
 & t_2 \arrow[r] & t_4 \arrow[r] & t_6 \\
t_1 \arrow[ur] \arrow[dr] & & & \\
 & t_3 \arrow[r] & t_5 \arrow[r] & t_7
\end{tikzcd}$$

The set of equivalence classes is

$H_1 = \{t_1\}$
$H_2 = \{t_2\}$
$H_3 = \{t_3\}$
$H_4 = \{t_4, t_5\}$
$H_6 = \{t_6\}$
$H_7 = \{t_7\}$,

and the derivative relation $\mathcal{R}$ (using C2) over this set is

$(H_1;H_1)$, $(H_1;H_2)$, $(H_1;H_3)$, $(H_1;H_4)$, $(H_1;H_6)$, $(H_1;H_7)$
$(H_2;H_2)$, $(H_2;H_4)$, $(H_2;H_6)$, and $(H_2;H_7)$ since t_2 R t_4 and t_5 R t_7
$(H_3;H_3)$, $(H_3;H_5)$, $(H_3;H_7)$, and $(H_3;H_6)$ since t_3 R t_5 and t_4 R t_6
$(H_4;H_4)$, $(H_4;H_6)$, $(H_4;H_7)$
$(H_6;H_6)$, $(H_7;H_7)$

or, diagrammatically:

$$\begin{tikzcd}
 & H_2 \arrow[dr] & & H_6 \\
H_1 \arrow[ur] \arrow[dr] & & H_4 \arrow[ur] \arrow[dr] & \\
 & H_3 \arrow[ur] & & H_7
\end{tikzcd}$$

The procedure thus applied is supposed to produce a model for B. But it does not. $V(Lp;H_2) = F$, since H_2 $\mathcal{R}$ H_7 and p does not have the value T at H_7; likewise $V(Lq;H_3) = F$, since H_3 $\mathcal{R}$ H_6 and q does not have the value T at H_6. Thus $V(B;H_1) = F$, rather than T. This happens in the present case because at H_7 no value at all is assigned to p, and at H_6 none is assigned to q. But I am confident that with ingenuity one could construct an example where they actually receive the value F.

The reason for this failure of the procedure is that membership in the equivalence classes is drawn from too broad a field. As it stands, the procedure identifies all equivalent tableaux with each other, no matter where they may fall within the web of R-access. Thus it identifies t_4, to which t_2 has access, with t_5 to which it does not have access, merely because they allocate the same truth-values to the same wffs. And that is not a good enough reason, since the two tableaux arise under consideration of *different* possibilities: the one leading to the even-numbered arm of the array, and the other leading to the odd-numbered arm.

An array of tableaux is a structure of access paths generated via the relation R. Its *nodes* are the tableaux containing leftcolumn M-wffs, which are Advanced to extend the path on which they occur. Its *followers* are the tableaux containing leftcolumn L-wffs, which are Executed (in $S4$) to each tableau farther down the path. The reason for the "blockage" mechanism is to terminate the otherwise endless paths that can crop up in $S4$ when followers generate new nodes. But this can be accomplished just as well by waiting to invoke "blockage" until a tableau is a duplicate of one earlier on its own access path. (Such dulications are inevitable on an endless $S4$ path, since tableaux are of finite length and formulas are only finitely complex.) Equivalence classes may then be restricted to duplicate tableaux lying on a common path.

This will resolve the current difficulty, since it can be shown that if a follower containing Lp has access to a tableau t_m, then p will appear in every tableau to which every equivalent of t_m has access, whether earlier or later on the path containing the follower. More formally, if Lp is in t_n and $t_n\ R\ t_m$, then for any t_j equivalent to t_m, if $t_j\ R\ t_n$ or $t_n\ R\ t_j$ then for every t_k such that $t_j\ R\ t_k$, p belongs to t_k.

There are only four ways for a formula to enter a tableau: as a "given" in the initial tableau (PREM); by Advancement from an R-earlier tableau in the array (ADV); by Execution from an L-wff in that tableau or an R-earlier one (EX); and by truth-functional reduction from local formulae (TF). If a wff w enters a tableau by TF, it will have an ultimate truth-functional source which entered the tableau not by TF but in one of the other ways. Call this source $S(w)$. Only the initial tableau in an array has formulae by PREM. A tableau receives at most one formula by ADV. The initial tableau in an array receives no formulae by ADV.

1. Let t_n be an arbitrary tableau containing an L-wff Lp.
2. Let t_m be such that $t_n\ R\ t_m$.
3. Let t_j be equivalent to t_m, and such that either $t_n\ R\ t_j$ or $t_j\ R\ t_n$.
4. Let t_k be such that $t_j\ R\ t_k$.
   TO SHOW: t_k contains p.

If $t_n\ R\ t_j$, then by the transitivity of R, $t_n\ R\ t_k$, hence by the Execution rule p is a member of t_k.

Proof for the other case is more elaborate. Suppose $t_j\ R\ t_n$. Then,

5. Let Mq be the formula Advanced (from t_t identical with or R-after t_n) to begin the tableau t_m.

Then t_m contains p (by EX), and q (by ADV). Since t_j is equivalent to t_m, it also contains both p and q. And t_j cannot be the initial tableau in the array, since t_m is equivalent to it and no subordinate tableau can be equivalent to the initial one. The initial tableau projects only the operanda of its members up the various paths; and since its membership is finite its operanda cannot be equal to its members. Therefore no formula entered t_j by PREM, and exactly one formula entered it by ADV.

Suppose p entered t_j by ADV. Then q entered it by EX or by TF. Suppose q entered t_j by EX. Then by transitivity of R and the Execute rule, q is also a member of t_t; hence Mq is not Advanced from t_t (it would be superfluous), which contradicts (5). Therefore q did not enter t_j by EX, but by TF.

Since q entered t_j by TF, $S(q)$ entered by ADV or by EX. It cannot have entered by ADV if p did, therefore it entered by EX. But then by transitivity of R, the Execute rule, and the truth-functional reduction rules, both $S(q)$ and q are members of t_t and, as above, Mq is not Advanced, which contradicts (5). Therefore, q did not enter t_j by TF either, and so did not enter t_j, which contradicts (3,5).

Therefore, p did not enter t_j by ADV, but by EX or TF. Suppose p entered t_j by TF. Then $S(p)$ entered by EX or by ADV. Suppose $S(p)$ entered by ADV. Then q entered by EX or TF. But in either of these cases, Mq is not advanced, which contradicts (5). Therefore $S(p)$ entered by EX. In that case, by transitivity of R and the Execute rule both $S(p)$ and p are members of t_k.

Finally, suppose p entered t_j by EX. Then by transitivity of R and the Execute rule, p is a member of t_k.

The modified restriction on blockage and equivalence-class membership may now be stated as:

M1 If t_n and t'_n are duplicate tableaux such that $t_n \ R \ t'_n$, then and only then

(i) t'_n is not to be subjected to the Advancement rule; and
(ii) t_n and t'_n belong to a common equivalence class H_n.

The proof just given shows that if H_n and H_m are formed via M1, then if Lp is in t_n in H_n and $H_n \ \mathcal{R} \ H_m$ via C2, then p is in every t_m in H_m. The problem of accessible H's devoid of p cannot arise. In any equivalence class formed under M1, the R-earliest member will have been Advanced, thus assuring that whenever Mp is in leftcolumn H_j, there is an $\mathcal{R}$-accessible H_k with p in its leftcolumns. Since C2 defines $\mathcal{R}$ as an ancestral, it will invariably be transitive, and will be reflexive because R is. With these important details in place, we may advert to Kripke for the remainder of the proof of the effectiveness of the $S4$ tableau procedure.

The discussion so far has been concerned solely with the Kripke tableau procedure. But it is important to recognize that the "nontransitivity" problem is not unique to that procedure. It can also show up in other $S4$ decision methods based upon, or having the same "logical geography" as, the tableau procedure. Specifically, any method which develops its model incrementally, one "world" at a time, with a mechanism for shutting off an infinitely repetitive $S4$ construction, and which generates the relation $\mathcal{R}$ in the model from the relation R in the construction, is susceptible to the problem. The formula A, which was deliberately developed to make the additions to R fall "between each other" in such a way as to interrupt transitivity, provides a convenient test. For example, when the diagram method of [2] is applied to the formula A, the resulting model fails of transitivity. I shall not discuss the modifications needed to correct that procedure.

The Lemmon-Scott "decision procedure", based upon filtrations, is not susceptible to the same problem since it does not base $\mathcal{R}$ upon R, but generates it *de novo* on the basis of what is and is not true at each world.[3]

NOTES

1. For the sake of simplicity, this summary differs from Kripke's exposition in one respect: Kripke's rules do not directly consider *M*-wffs. Instead of "Advancing a leftcolumn *M*-wff", the procedure is: if *NLNp* is in leftcolumn t_n, put *LNp* in rightcolumn t_m. Advance to a new tableau t_m with *Np* in its rightcolumn; then put *p* in leftcolumn t_m. The outcome is the same.

2. My attempts to discover an example of such a formula, or to prove that there are none, have been unsuccessful.

3. However, that "procedure" does not provide a humanly useful method for deciding particular cases. It is based on the theorem that every *S*4-satisfiable formula is finitely satisfiable, and says, in effect: Start generating the finite *S*4 models. Since there are only denumerably many of them, if your particular formula is satisfiable a model for it will show up in a finite time. See [1] for details.

REFERENCES

[1] Chellas, B., *Modal Logic: An Introduction*, Cambridge University Press, Cambridge, 1980.

[2] Hughes, G. E. and M. J. Cresswell, *An Introduction to Modal Logic*, Methuen and Co., Ltd., London, 1968.

[3] Kripke, S. A., "Semantical analysis of modal logic I: Normal modal propositional calculi," *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, Band 9 (1963), pp. 67-96 (esp. pp. 89-90).

[4] Lemmon, E. J. (in collaboration with D. Scott), *The "Lemmon Notes": An Introduction to Modal Logic*, ed. K. Segerberg, American Philosophical Quarterly Monograph Series, Basil Blackwell, Oxford, 1977.

Department of Philosophy
The University of Utah
Salt Lake City, Utah 84112