*Journal of Automated Reasoning* 7: 489–510, 1991.



# TABLEAUX: A General Theorem Prover for Modal Logics[1]

LAURENT CATACH
*LITP – Paris 6 University, IBM Paris Scientific Center, France*

(Received: 27 June 1988; accepted: 8 April 1991)

**Abstract.** We present a general theorem proving system for propositional modal logics, called TABLEAUX. The main feature of the system is its generality, since it provides an unified environment for various kinds of modal operators and for a wide class of modal logics, including usual temporal, epistemic or dynamic logics. We survey the modal languages covered by TABLEAUX, which range from the basic one L(□, ◇) through a complex multimodal language including several families of operators with their transitive-closure and converse. The decision procedure we use is basically a semantic tableaux method, but with slight modifications compared to the traditional one. We emphasize the advantages of such semantical proof methods for modal logics, since we believe that the models construction they provide represents perhaps the most attractive feature of these logics for possible applications in computer science and AI. The system has been implemented in Prolog, and appears to be of reasonable efficiency for most current examples. Experimental results are given in the paper, with two lists of test examples.

**Key words.** Modal (temporal, dynamic, epistemic) logics, theorem proving, decision procedures, tableaux method, experimental results.

## 1. Introduction

TABLEAUX is an automated theorem proving system for a wide class of modal logics, based on an adapted version of the traditional *semantic tableaux method* for these logics [20, 21, 30, 49]. This method also works for many extensions of traditional modal logic, such as temporal logics [3, 4, 15, 58], dynamic logics [29, 43], or epistemic logics [27, 31], which have become of some importance in computer science and AI.

In fact, the tableaux method is frequently used within the theoretical presentation of a modal system; this is due to some of its inherent advantages:

- it is a proof method closely related to the semantics of the modal operators,
- it provides a completeness theorem and a decision procedure, and may also be used to prove complexity results about the logic (e.g. the Fischer–Ladner filtrations method for dynamic logic [22]),
- the technique has a general scheme, so it can be adapted from one logical system to another,
- it works for proving both the *validity* and the *satisfiability* of formulas,
- it is *constructive*, i.e. it explicitly builds a model (resp. a counter-model) when a formula is found to be satisfiable (resp. non-valid).

The basic tableaux method is simple, but faces important problems of complexity; this is certainly the reason why few real implementations have been built [25, 42, 44, 51, 53]. Other methods have been explored to achieve automated deduction for these logics (see Section 5), but complexity is always crucial, especially for modal languages including transitive-closure operators.

Our motivations for developing TABLEAUX and the purposes of such a system are the following:

- To build a *general* theorem prover for many modal logics, and to use it as a *tool* for studying these logics. Of course, such a system is interesting for the logician, but we believe that it can also be very helpful for new developments in computer science and especially for experimenting with ideas, exploring possible applications and creating prototypes.
- To be able, in that theorem prover, to choose a modal system in the most *declarative* way, i.e. by defining the language of modal operators and by giving semantic or axiomatic characterizations of the logic.
- Most automated deduction methods recently proposed for modal logics are syntactic, in the sense that they do not rely on the *explicit* construction of models. On the contrary, we believe that it is precisely this feature of modal logics which is perhaps the most attractive, if modal logics are to be of any practical interest in computer science and AI. Thus, one of our motivations was to use a *semantic* deduction method.
- As shown by TABLEAUX, starting with a basic modal language and the tableaux method, one can obtain many of the modal systems developed in the literature, from the aspect of both the expressiveness (operators of temporal, epistemic, dynamic, etc. logics) and of the behavior (simulation [40], model checking [10], epistemic puzzles, etc.); thus, an important feature of TABLEAUX is its provision of a unified version of various existing modal systems and applications.
- To show that it is possible to implement the tableaux method with good efficiency for current examples. Inasmuch as complexity is indeed a handicap in the general case, it also depends very much on the systems and on the formulas considered.
- Starting from the 'classical' tableaux method, to discover where problems arise in an implementation, and to see whether new methods could be defined (with better efficiency). In fact, we believe that such prototypes, written in a powerful language such as Prolog, constitute necessary preliminaries before the realization of more efficient theorem provers, based for example on low-level languages.
- One main motivation for developing TABLEAUX was also to study expressive features of modal languages, and especially those of *general multimodal systems*, as studied in [7, 8]. This topic is briefly presented in Section 2.7.

The modal operators and systems covered by TABLEAUX are described in Section 2. This section does not contain new results about expressiveness or axiomatisations (though it might suggest interesting extensions, such as the multimodal languages we consider in Section 2.7), but it is necessary for us to give a complete and

unified presentation of the language covered by our system. As the reader will see, TABLEAUX provides a general environment for using many of the modal, temporal, dynamic, epistemic, etc. systems developed in the literature, and also for studying new ones.

The proof method used in TABLEAUX, which is described in Section 3, actually combines some features of the various proof methods developed for modal logics. Basically, it is a tableaux method since it is based on an *explicit construction of a counter-model*; the tableaux method we use is inspired from both the classical one [20, 30, 52] and the *prefixed tableaux systems* described in [21]; but the formalism of our tableaux rules is close to the system of (resolution) production rules developed in [1]. Also, *resolution* [18] is used as much as possible (i.e. as a heuristic), in order to reduce the size of the model and to shorten the proofs. Finally, the output result given by TABLEAUX (i.e. the counter-model) can be interpreted as the *automaton* associated with the input formula [16, 55]. In fact, we believe that such an approach, combining several distinct proof methods, could lead to new, interesting (and efficient) decision procedures.

The system TABLEAUX is implemented in VM/Prolog [47], and runs on IBM-370 machines. It has been successfully tested with many examples, taken from various developments dealing with modal logics. Sample experimental results are given in Section 4, and a complete list of tests for 31 formulas in 16 distinct modal systems is given at the end of the paper. Our present version, though fast on the example we tried, could be subsequently improved, especially by restricting the program to a particular logical system.

We do not recall complexity results for the various modal system we consider and we refer to, for example, References [15, 31, 36, 50 and 29] for more details on this topic.

## 2. Modal Systems

We now describe the modal operators and the calculi that TABLEAUX can handle. Note that all modal operators described in this section can be used with arbitrary *names* within TABLEAUX. Only propositional languages are considered, but with graded levels of expressiveness, depending on the modal operators which are used. All these operators are traditional in modal, temporal and epistemic logics, so only a brief survey will be given in this section; however, it provides a unified vision of the developments in the domain and might be useful for the reader who is unfamiliar with modal logic. We refer the reader to the general bibliography for axiomatic characteristizations.

In the following, $\Phi$ denotes a set of propositions variables $p$, $q$, $r$ . . . and $\wedge$, $\vee$, $\supset$, $\neg$, $\equiv$ denote the boolean connectives. Arbitrary formulas will be noted $\alpha$, $\beta$ . . . and the symbols $\top$ and $\bot$ denote the constants 'true' and 'false'. Two modal operators $O$ and $O'$ are dual if the axiom scheme $\vdash(O\neg\alpha \equiv \neg O'\alpha)$ holds.

2.1. BASIC MODAL LOGIC

Modal logic extends the language of classical logic by introducing two basic modal operators, denoted $\Box$ ('necessarily') and $\Diamond$ ('possibly'), which are unary. The smaller system of propositional modal logic is $K$, the axiomatisation of which is:

- axioms and inference rules for the propositional calculus, including Modus Ponens,
- axiom $K$. $\Box(\alpha \supset \beta) \supset (\Box\alpha \supset \Box\beta)$
- inference rule RN. if $\vdash \alpha$ then $\vdash \Box\alpha$ (rule of Necessitation)

Models and semantics for $K$ can be described in terms of *possible worlds* structures [5, 11, 30]. A *Kripke model* for modal logic is a triple $M = \langle W, R, V \rangle$, where $W$ is a non-empty set (of 'possible worlds'), $R$ is a binary relation over $W$ (called the 'accessibility relation') and $V: W \times \Phi \to \{0, 1\}$ is an evaluation function which gives in each world $w \in W$ the truth values $V(w, p)$ of the propositional variables $p \in \Phi$. A (Kripke) *frame* is simply a pair $\langle W, R \rangle$ as above. We write $(M, w) \models \alpha$ for '$\alpha$ is true at world $w$ in the model $M$', and this notion of truth is defined inductively on the complexity of $\alpha$ ($\neg$ and $\vee$ are taken as primitives):

- if $p \in \Phi$, $(M, w) \models p$ iff $V(w, p) = 1$
- $(M, w) \models \neg\alpha$ iff $(M, w) \not\models \alpha$
- $(M, w) \models (\alpha \vee \beta)$ iff $(M, w) \models \alpha$ or $(M, w) \models \beta$
- $(M, w) \models \Box\alpha$ iff $(M, w') \models \alpha$ for all $w'$ such that $R(w, w')$
- $(M, w) \models \Diamond\alpha$ iff $(M, w') \models \alpha$ for some $w'$ such that $R(w, w')$

This definition of $\models$ gives the semantic interpretation of modal sentences built with operators $\Box$ and $\Diamond$. A formula $\alpha$ is said to be *satisfiable* in a model $M$ if there exists a world $w$ such that $(M, w) \models \alpha$, and *unsatisfiable* otherwise. Also, $\alpha$ is said to be *valid* with respect to a class of models $C$ if $\neg\alpha$ is unsatisfiable in every model of $C$, which means that $(M, w) \models \alpha$ for every model $M = \langle W, R, V \rangle \in C$ and for every world $w \in W$.

A logical system $L$ is said to be *determined* by a class of models $C$ if, for every formula $\alpha$, '$\alpha$ is a theorem of $L$' if and only if '$\alpha$ is valid in $C$'. The system $K$ is determined by the class of all models [11, 30].

The system $K$ can easily be extended in the following way: if $M = \langle W, R, V \rangle$ is a Kripke model and $R^{-1}$ denotes the *converse* relation of $R$ (i.e. $R^{-1}(w, w') = R(w', w)$), two new dual operators $\Box^{-1}$ and $\Diamond^{-1}$ can be introduced with the following semantics:

- $(M, w) \models \Box^{-1}\alpha$ iff $(M, w') \models \alpha$ for all $w'$ such that $R^{-1}(w, w')$
- $(M, w) \models \Diamond^{-1}\alpha$ iff $(M, w') \models \alpha$ for some $w'$ such that $R^{-1}(w, w')$

The system $K_t$ denotes the extension of $K$ obtained by adding the 'converse' operators $\Box^{-1}$ and $\Diamond^{-1}$ together with the following axioms:

- $\alpha \supset \Box\Diamond^{-1}\alpha$
- $\alpha \supset \Box^{-1}\Diamond\alpha$

The system $K_t$ is known as the basic system of temporal logic [49], where the modal operators $\Box$, $\Diamond$, $\Box^{-1}$, $\Diamond^{-1}$ are usually denoted $G$, $F$, $H$ and $P$, respectively. The system $K_t$ is (like $K$) determined by the class of all models.

Many $K$ (or $K_t$) systems can be obtained by adding new axioms. In the favourable cases, the resulting systems are determined by classes of Kripke models where the accessibility relation $R$ possesses some particular properties. In this sense, some axioms of modal logic are said to *correspond* to a property of $R$ in the Kripke semantics.

Some well-known correspondences of this type are given in the chart below (notation is from [11]:

| | axiom scheme | property of R |
|---|---|---|
| 1 | **D.** $\Box p \supset \Diamond p$ | seriality: $(\forall w)(\exists w')$, $R(w, w')$ |
| 2 | **T.** $\Box p \supset p$ | reflexivity: $(\forall w)$, $R(w, w)$ |
| 3 | **B** $p \supset \Box\Diamond p$ | symmetry. $R(w_1, w_2) \Rightarrow R(w_2, w_1)$ |
| 4 | **4** $\Box p \supset \Box\Box p$ | transitivity. $R(w_1, w_2)$ & $R(w_2, w_3) \Rightarrow R(w_1, w_3)$ |
| 5 | **5.** $\Diamond p \supset \Box\Diamond p$ | euclideanity: $R(w_1, w_2)$ & $R(w_1, w_3) \Rightarrow R(w_2, w_3)$ |
| 6 | **U.** $\Box(\Box p \supset p)$ | quasi-reflexivity: $R(w_1, w_2) \Rightarrow R(w_2, w_2)$ |
| 7 | **F** $\Diamond p \supset \Box p$ | quasi-functionality. $R(w_1, w_2)$ & $R(w_1, w_3)$ $\Rightarrow (w_2 = w_3)$ |
| 8 | **H.** $(\Diamond p \wedge \Diamond q) \supset \Diamond(p \wedge q)$ $\vee \Diamond(\Diamond p \wedge q)$ $\vee \Diamond(p \wedge \Diamond q)$ | forwards-linearity. $R(w_1, w_2)$ & $R(w_1, w_3)$ $\Rightarrow R(w_2, w_3)$ or $(w_2 = w_3)$ or $R(w_3, w_2)$ |
| 9 | **W.** $\Box(\Box p \supset p) \supset \Box p$ | no-infinite-chains. there is no infinite sequence $(w_0, w_1 \cdots w_n \cdots)$ such that $R(w_n, w_{n+1})$ for all $n \geqslant 0$ |

Other properties can be defined in terms of the ones given above, such as $R$ being a pre-order relation ((2) and (4)), a similarity relation ((2) and (3)), an equivalence relation ((2), (3) and (4)), a functional relation ((1) and (7)), a backwards-linear relation ($R^{-1}$ is forwards-linear), a linear relation (both backwards and forwards linear), a backwards-serial (or infinite towards the past) relation ($R^{-1}$ is serial). Note that the class of frames $\langle W, R\rangle$ for which $R$ is functional determines the extension of $K$ with characteristic axiom $\Box\alpha \equiv \Diamond\alpha$. In this case, $\Box$ (or $\Diamond$) is usually noted as O, the 'next' operator, and we get a system with another type of linearity (see also 2.2).

Combining one or more properties of $R$, one can build many systems of modal logics. For example, properties (1)–(5) above generate *fifteen* distinct normal $K$-systems, among which the traditional systems $T$ (axiom **T**), $B$ (axioms **T**, **B**), $S4$ (axioms **T**, **4**), $S5$ (axioms **T**, **B**, **4**), and their deontic versions $KD$, $KDB$, $KD4$ and $KD5$ (see [11]). Other traditional $K$ or $K_t$ systems are indicated below:

- $S4.3 = S4 + \{\mathbf{H}\}$
- $Tr = K + \{\Box p \equiv \Diamond p\}$
- $K_c = K_t + \{\mathbf{4}\}^2$
- $K_h = K_t + \{\mathbf{4}$, backwards-linearity$\}$ = the system of 'branching time'
- $K_1 = K_h + \{\mathbf{H}\}$ = the system of 'linear time' (Cochiarella system)

- $K_l^{\infty +} = K_l + \{\mathbf{D}\}$ = the system of 'linear time' with infinite future
- $K_s = K_l^{\infty \pm} = K_l + \{$seriality, infinite towards the past$\}$ = the system of 'linear time' with infinite future and past (D. Scott)
- $G = K4 + \{$no-infinite-chains$\}$ = the Gödel system $G$ (see [21])

The above systems are determined by the classes of models in which the accessibility relation $R$ has the corresponding properties. For example, $T$ is determined by the class of reflexive models, $S5$ is determined by the class of models where $R$ is an equivalence relation, etc. We refer to [5, 11, 21, 30 and 49] for more details about the construction and determination of modal systems.

In TABLEAUX any combination of the semantic properties of the accessibility relation $R$ described above is allowed; as mentioned before, this yields many possible $K$ or $K_t$ systems.

### 2.2. TEMPORAL LOGICS

Basic systems, such as $K_t$, $K_b$, etc., have already been mentioned above.

#### *Branching Time and Path Operators*

If $M = \langle W, R, V \rangle$ is a model, an $R$-path is a sequence of worlds $c = (w_0, w_1, \ldots, w_n, \ldots)$ which is a maximal linearly $R$-ordered subset of $W$, i.e. $R(w_n, w_{n+1})$ for all $n \geqslant 0$ and either $c$ is infinite or its last element has no $R$-successor. If $w \in W$, a $w/R$-path is an $R$-path starting at $w$, i.e. with $w_0 = w$.

TABLEAUX includes *CTL* (Computation Tree Logic), as defined in [9]. *CTL* is obtained from basic modal logic by adding the path operators $\forall\Box$, $\exists\Box$, $\exists\Diamond$, $\forall\Diamond$[3] and 'until' operators $\forall U$ and $\exists U$, whose semantics are defined in Kripke models in terms of quantification over $R$-paths as follows:

- if $c = (w_0, w_1, \ldots, w_n, \ldots)$ is a path, $(M, c) \vDash \alpha$ iff $(M, w_n) \vDash \alpha$ for all $n \geqslant 0$
- $(M, w) \vDash (\forall\Box)\alpha$ iff $(M, c) \vDash \alpha$ for all $w/R$-paths $c$
- $(M, w) \vDash (\exists\Box)\alpha$ iff there exists a $w/R$-path $c$ such that $(M, c) \vDash \alpha$
- $(M, w) \vDash \alpha\,(\forall U)\,\beta$ iff for all $w/R$-paths $c = (w_0, w_1, \ldots)$ there exists $n \geqslant 0$ such that $(M, w_n) \vDash \beta$ and $(M, w_i) \vDash \alpha$ for all $i < n$
- $(M, w) \vDash \alpha\,(\exists U)\,\beta$ iff there exists a $w/R$-path $c = (w_0, w_1, \ldots)$ as above

and where $(\exists\Diamond)(\forall\Diamond)$ are defined as the dual operators of $(\forall\Box)(\exists\Box)$, respectively. Note that as, in a world $w$, $\Box$ and $\Diamond$ talk about the worlds which are accessible from $w$, the above operators talk about the paths of worlds starting at $w$. For example, $(\forall\Diamond)p$ is true at $w$ if, for every path $c$ starting at $w$, $p$ is true at least one world on $c$ ($p$ is *inevitable*).

The weaker system *UB* of [4] is obtained by omitting the 'until' operators. Also, in the language with $\forall U$ and $\exists U$ only, $\forall\Box$ and $\exists\Box$ (and therefore their dual $\exists\Diamond$ and $\forall\Diamond$) can actually be defined by $(\forall\Box)\alpha \equiv_{\text{Def}} (\top\,(\forall U)\alpha)$ and $(\exists\Diamond)\alpha \equiv_{\text{Def}} (\top(\exists U)\alpha)$.

Path operators are now traditional in the temporal logic of branching time, and many systems use them [4, 9, 10, 15, 17]. We also refer to the bibliography for a description of complete axiomatizations.

Variants of *CTL* or *UB* can easily be obtained by imposing semantic properties on the basic accessibility relation $R$. Generally, it is assumed that $R$ is serial (so paths are always infinte); in the case where $R^{-1}$ is quasi-functional we obtain a branching time logic dealing with trees [6, 26]. Also, many of these variants can be obtained by defining new operators in terms of existing ones (see also 2.4). We omit the details here.

A more general class of operators, which can arbitrarily combine quantification over worlds, has been proposed in [17]. We do not present the corresponding system *CTL** here, since this 'full branching time logic' does not have a tableaux method decision procedure.

*Linear Time*

In the case where $R$ is taken to be functional in *CTL*, quantification over paths collapses and so do path operators. Usual notations are then $\circ$ for $\Box = \Diamond$, $\Box$ for $\forall\Box = \exists\Box$, $\Diamond$ for $\forall\Diamond = \exists\Diamond$ and $U$ for $\forall U = \exists U$. The resulting system, based on $L(\{\circ, \Box, \Diamond, U\})$ is linear-time temporal logic *PTL* [57, 58].[4] As for *CTL*, there are different kinds of linear time logics, depending on the properties of $R$. They are covered by TABLEAUX.

In [57], *PTL* is extended to express any property dealing with linear paths (using right-linear grammars) and a wide class of temporal operators is introduced. Our present version of TABLEAUX does not cover this system *ETL*, but this could be easily done, since *ETL* has a tableaux method decision procedure; however, we think the interest in such operators depends very much on the intended applications.

*Past Path Operators*

Path operators corresponding to $R^{-1}$ can be introduced, written $\forall\Box^{-1}$, $\exists\Box^{-1}$, $\forall\Diamond^{-1}$, $\exists\Diamond^{-1}$ (or equivalently $\forall H$, $\exists H$, $\forall P$, $\exists P$). $U^{-1}$ is written $S$ (for 'since'), so we have the operators $\forall S$ and $\exists S$. The particular case of linear time can also be considered, with operators $\circ$, $\Box$, $\Diamond$, $U$, $\circ^{-1}$, $\Box^{-1}$, $\Diamond^{-1}$ and $S$. In this case, the modal systems induced by $\Box$ and $\Diamond$ are closely related to the systems $K_h$ or $K_l$ (or their extensions) mentioned in Section 2.1.

*Modal Logic of Time Intervals*

A modal logic of time intervals has been proposed in [32]. Basically, it is a multimodal system (see 2.7 below), with six primitive modal operators $\langle B\rangle$, $\langle E\rangle$, $\langle A\rangle$ and their

converses $\langle B' \rangle$, $\langle E' \rangle$ and $\langle A' \rangle$, from which many other operators can be defined. Worlds are interpreted as intervals, so for example $\langle B \rangle \alpha$ is true at an interval if $\alpha$ is true at some subinterval with the same beginning. This logic is also studied in [54], where a complete axiomatization is given; but no tableaux decision procedure seems to exist as yet. However, TABLEAUX provides a nice framework for studying these logics (see 2.7).

### 2.3. EPISTEMIC LOGICS

Epistemic logics use families of modal operators of the same type, denoted as $K_1$, $K_2$, . . . , $K_n$, where $K_i\alpha$ is intended to mean 'agent $i$ knows that/believes that $\alpha$ holds'. TABLEAUX handles epistemic systems $K_{(m)}$, $T_{(m)}$, $S4_{(m)}$, $S5_{(m)}$ or $KD45_{(m)}$ proposed in [31], which are just multi-dimensional versions of the normal $K$-systems $K$, $T$, $S4$, $S5$ and $KD45$.

As described in [31], interesting extensions are obtained by introducing *common knowledge* (or *common belief*) operators, denoted by $E$ and $C$. Note that $C$ is very similar to $\forall\Box$ (see 2.2), since it is related to $E$ exactly as $\forall\Box$ is related to $\Box$ (i.e. by a transitive-closure correspondence). Also, these operators can be defined for arbitrary subgroups of agents, as in [19]. TABLEAUX also covers epistemic systems incorporating both knowledge and belief operators [37], as well as their respective common operators for groups or subgroups of agents (see 2.7 below).

### 2.4. DEFINITIONS

In TABLEAUX, any operator definition is available, and introducing new operators (not necessarily unary) in terms of the existing ones is often very useful. For example, in $K_t$, we can define a new operator $\Delta$ by $\Delta\alpha \equiv_{\text{Def}} (\Box^{-1}\alpha \wedge \alpha \wedge \Box\alpha)$, which reads '$\alpha$ is, has always been and will always be true' (see also example (3) in Section 4). Similarly, the $E$ epistemic operator can simply be defined as $E\alpha \equiv_{\text{Def}} (K_1\alpha \wedge K_2\alpha \wedge \cdots \wedge K_n\alpha)$. Also, many semantic or axiomatic variants of modal systems can be handled using definitions, i.e. by translating one system into the other (this method is systematically used in [42]).

### 2.5. DYNAMIC LOGIC

Dynamic logic [22, 29], which has been introduced as a symbolic language to reason, state and prove properties about *programs*, also introduces a family of modal operators, as in epistemic logic; these operators are written $[a]$ instead of $\Box_a$, where $a$ denotes a program. The main feature of dynamic logic is the introduction of program operations ';' (composition), '$\cup$' (union) '$*$' (iteration) and '?' (tests), so that if $a$, $b$ are programs, so are $(a;b)$, $(a \cup b)$ and $a^*$, and if $\alpha$ is a formula then $?\alpha$ is a program.

TABLEAUX covers the dynamic systems *PDL*, *DPDL* and *CPDL* ([29]). Handling the ';' '$\cup$' and '?' operations is easily done, using the following definitions:

$$[a;b]\alpha \equiv_{\text{Def}} [a][b]\alpha$$

$$[a \cup b]\alpha \equiv_{\text{Def}} ([a]\alpha \wedge [b]\alpha)$$

$$[?\alpha]\beta \equiv_{\text{Def}} (\alpha \supset \beta)$$

For iteration, we observe that $[a^*]$ is related to $[a]$ exactly as $(\forall\Box)$ is related to $\Box$ in branching-time logic, i.e. again by a transitive-closure correspondence; therefore, handling the '$*$' operation is similar to the $\forall\Box$ operator case (and to the *C* operator case too). Finally, the *converse* operation of *CPDL* is treated as in Section 2.1 for the $\Box^{-1}$ operator. To obtain *DPDL*, we simply add the restriction that the binary relations associated with atomic programs are serial.

## 2.6 OTHER LOGICS

Though the philosophical motivations are different, *deontic logic* (see [28] II.11) is syntactically very close to basic modal logic, and many deontic systems (e.g. [11] Ch.6) can therefore be studied within TABLEAUX. Also, *conditional logic* shares many features with modal logic. As an example, TABLEAUX covers the normal conditional logics studied in [25], since the binary modal operator of conditional implication $\Rightarrow$ used in [25] seems to be definable in terms of the standard $\Box$ and $\Diamond$ by

$$\alpha \Rightarrow \beta \equiv \neg\Diamond\alpha \vee \Diamond(\alpha \wedge \beta \wedge \Box(\alpha \supset \beta))$$

Many other interpretations of modal logics have been proposed. The reader is referred to [28] for more information.

## 2.7 MULTIMODAL LOGICS

The language of multimodal logics [7, 8] is that of dynamic logic, but with the additional feature that $[a]$, $[b]$, etc. are used to denote *arbitrary* modal operators (with possibly completely different meanings), that these operators are allowed to belong to modal systems of different type (see below) and may also induce *interactions* between each other. Such systems are motivated by the simultaneous study of several modal aspects, e.g. *knowledge and time*, *knowledge and belief*, etc., and also by systems like interval modal logic (see 2.2). We refer to [8] for more information about these topics.

TABLEAUX handles all multimodal systems which are *simple*, in the terminology of [7], i.e. without interactions between distinct operators $[a]$, $[b]$, . . . , and in the case where all these operators fall within the scope of the systems described in Section 2.1. For example, we can simultaneously use an *S*4-operator $\Box_1$, a *KD*45-operator $\Box_2$, *n* *K*-operators $\Box_3^1 \cdots \Box_3^n$, etc. Dynamic logics, and also epistemic logics, are

multimodal systems which are *simple* and also *homogeneous*, i.e. such that the subsystems associated with [*a*], [*b*] . . . are all identical (*KD*, *T*, *S*4, etc).

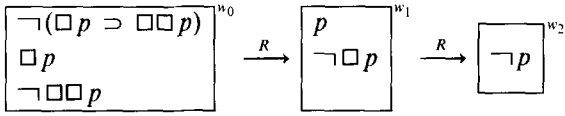
In [7], a class of *interaction axioms* (axioms involving distinct modal operators) is proposed and clearly this ability to specify interrelations is one main feature of multimodal logics. This study is also developed in [8]. The ability to handle such interaction axioms has not yet been incorporated in TABLEAUX, since theoretical investigations about the semantic tableaux method for multimodal logics are not fully examined. However, TABLEAUX appears to be a valuable tool for studying these systems, and the ability to use arbitrary *simple* multimodal systems already makes it very general, compared to other deductive systems for modal logics.

## 3. The Tableaux Method

### 3.1. OVERVIEW OF THE METHOD

Let $L$ be a modal system characterized by a class $C$ of Kripke models and let $\alpha$ be a formula in $L$. If $\alpha$ is not a theorem of $L$, then $\neg\alpha$ must be satisfiable in some $C$-model (called a *counter-model* for $\alpha$); conversely, if $\alpha$ is a theorem of $L$, then $\neg\alpha$ must be $C$-unsatisfiable, so that any attempt to build a $C$-model for $\neg\alpha$ should *necessarily* lead to a contradictory model. Therefore, the decision procedure for $L$ consists in systematically trying to construct $C$-models satisfying a given formula, or, more generally, a set $\Sigma$ of formulas.

Thus, the tableaux method works by necessary conditions: starting with a world $w_0$ in which it is assumed that all formulas of $\Sigma$ hold, a model is progressively constructed 'around' $w_0$ using *tableau rules*, which reflect the semantic laws of the satisfaction relation '$\models$' that must be fulfilled. Proofs are written in *tree form*, because *alternative tablaux* may be introduced by disjunctive formulas. A simple example of tableaux construction is given below, corresponding to the test for validity for the formula **4**. $\Box p \supset \Box\Box p$ (**4** is the axiom scheme expressing transitivity in Kripke models, see 2.1) in the system $K$, for which no special condition holds for $R$:




The above tableau represents a counter-model for the formula **4**, which is therefore not valid in $K$. If we now consider the same construction with $R$ being transitive (i.e. in the $K4$ system), then $R(w_0, w_2)$ must hold; hence, $\Box p$ true at $w_0$ implies $p$ true at $w_2$, which gives $w_2$ inconsistent and therefore yields a contradictory model. Since this contradiction appears necessarily during the construction, $\neg$**4** is unsatisfiable in transitive Kripke models and **4** itself is a valid formula (and therefore a theorem) of $K4$.

As we can see, the tableaux method is *constructive* (i.e. it always gives a result), it works both for testing *validity* or *satisfiability* of formulas (since $\alpha$ is satisfiable iff $\neg\alpha$ is not valid) and *modular* (i.e. the construction adapts to different modal systems with

slight modifications). Also, we remark that a modal formula $\alpha$ determines implicitly (via the semantics of the modal operators appearing in it) a class of models (the class of models making $\alpha$ satisfiable) and that the tableau construction makes explicit the general form of these models [16].

This aspect is of particular interest for some applications; for example in dynamic logic, modal operators are intended to represent programs, so that the models are actually interpreted as *possible executions* of these programs [40]. The tableau construction can therefore provide some kind of *simulation* process; see [2, 23, 40] for further details.

### 3.2 INTERNAL REPRESENTATION IN TABLEAUX

In the traditional approach, a tableau is a directed graph, whose nodes are labeled by sets of formulas [25, 30, 42, 51], so that the internal representation has the form ⟨world⟩ : ⟨list of formulas⟩. Another method is the *prefixed tableaux system*, described in [21], which uses the converse representation ⟨prefix of worlds⟩ : ⟨formula⟩, where the prefix indicates the path in the graph leading to the world where the formula is considered. Note that this latter representation has also been (independently) developed in [13] and [33] (see also Section 5).

In TABLEAUX, an intermediate representation is used: we manipulate *labeled formulas*, which are items of the form $f = \langle w : \alpha : a \rangle$, where $\alpha$ is an ordinary formula, $w$ denotes the world in which $\alpha$ holds and $a \in \{0, 1\}$ is a label indicating whether $f$ has been already examined or not. Labels ensure that tableau rules are applied only once to a formula. A *tableau* $T$ is a pair $\langle \Gamma, \mathbf{R} \rangle$, where $\Gamma$ is a set of labeled formulas and $\mathbf{R}$ is a set of relations $R(w, w')$ between worlds. Note that $\mathbf{R}$ may contain several distinct relations $R$, so the tableau may represent a graph with different types of edges; therefore, this representation is suitable for arbitrary multimodal systems (see 2.7). If $w \in W$, we denote by $\Gamma_w$ the subset of $\Gamma$ containing the labeled formulas $\langle w : \alpha : a \rangle$ and we define the cut of $R$ by $w$ to be $R(w) = \{w' \in W / R(w, w')\}$, i.e. the set of $R$-successors of $W$. When testing satisfiability for a set $\Sigma$ of ordinary formulas we initialize a tableau $T$ with $\Gamma = \{\langle w_0 : \alpha : 0 \rangle / \alpha \in \Sigma\}$ and $\mathbf{R} = \{\ \}$, and we successively apply *rules* to transform $T$.

In the following, the full modal language described in Part 2 is taken into account. Note that no special form is required for input formulas, but (internally) we retain $\neg$ and $\vee$ as Boolean connectives only, and we identify $\neg\Box$ with $\Diamond\neg$ and $\neg\Diamond$ with $\Box\neg$. Since (simple) multimodal languages are allowed (see 2.7), we may have distinct modal operators with possibly different properties, so we write $\Box_R \Diamond_R$ $(\forall \Box_R)$ $(\exists \Box_R)$ $(\forall \Diamond_R)$ $(\exists \Diamond_R)$ $(\forall U_R)$ $(\exists U_R)$ for the modal operators whose semantics are determined by the relation $R$. With the notations of dynamic logic, $\Box_R \Diamond_R (\forall \Box_R)$ and $(\exists \Diamond_R)$ correspond to the operators $[a]$, $\langle a \rangle$, $[a^*]$ and $\langle a^* \rangle$, respectively.

A tableau $T$ is said to be *closed* if $\Gamma$ contains an *unsatisfiable* labeled formula; *inconsistent* labeled formulas $\langle w : \bot : a \rangle$ are of course unsatisfiable, but other cases of unsatisfiability (for path formulas) will be examined in Section 3.5 below.

3.3. TABLEAU RULES

Tableau rules are production rules which apply to labeled formulas and which preserve satisfiability of tableaux, so that if a rule applies to a tableau $T$ to give a new tableau $T'$ then $T$ is closed iff $T'$ is. For most tableau rules, $T'$ is obtained from $T$ by changing the set $\Gamma$ into a new set $\Gamma'$, i.e. by a rule of the form $f_1 \cdots f_k \to g_1 \cdots g_l$, but some rules also modify **R** (when new worlds are introduced), and other create alternative tableaux.

A first class of tableau rules are pure rewriting rules, which consist only in *replacing* a set of ordinary formulas $\alpha_1 \cdots \alpha_k$ by another set of ordinary formulas $\beta_1 \cdots \beta_l$ in labeled formulas; this means that the labeled formulas $f_i$ containing the $\alpha_i$ are discarded from $\Gamma$. Since this transformation is world-independent and does not change the value of the label $a$, such a rule will be written $\alpha_1 \cdots \alpha_k \to \beta_1 \cdots \beta_l$ instead of $\langle w:\alpha_1:a\rangle \cdots \langle w:\alpha_k:a\rangle \to \langle w:\beta_1:a\rangle \cdots \langle w:\beta_l:a\rangle$. In these rewriting rules, we do not keep track of the left-side labeled formulas $f_i$ where the $\alpha_i$ occur.

A list of rewriting rules, which are *simplification* rules, is given below:

$\neg\neg\alpha \to \alpha$

$\alpha, \neg\alpha \to \bot$ (in which case $T$ is closed)

$O\alpha, O'\neg\alpha \to \bot$ if $O$ and $O'$ are dual operators,

$\alpha_1 \vee \cdots \vee \alpha_k \to \top$ if one of the $\alpha_i$ is $\top$ or is a conjunction containing $\top$

$\alpha, (\alpha \vee \beta) \to \alpha$ (this discards the formula $\alpha \vee \beta$)

$\alpha, (\beta \vee \beta') \to \alpha, \beta'$ if $\beta$ is $\neg\alpha$ or if $\beta$ is a conjunction containing $\neg\alpha$

These last two rules are the classical *resolution* rules, but applied within a world only (as in [1] or [35]). As explained below, the $\vee$-formulas introduce alternatives in the construction (and consequently duplications of tableaux), so it is important to reduce these disjunctions as much as possible when they appear. These resolution rules are consequently very useful for shortening the proofs.

The simplification rules given above are not systematically applied by a permanent exhaustive exploration of $\Gamma$, but each time a *new* (labeled) formula $\langle w:\alpha:a\rangle$ is to be added to $\Gamma$, i.e. during the computation of $\{\langle w:\alpha:a\rangle\} \cup \Gamma$. This guarantees that the set $\Gamma$ is maintained saturated (by the rewriting rules) and improves efficiency.

A second class of tableau rules are *transformation* rules, which have exactly the same format as rewriting rules except that labels change from 0 to 1, so that we keep track of the formulas $\alpha_i$ used. More precisely, we write $\alpha_1 \cdots \alpha_k \mapsto \beta_1 \cdots \beta_l$ a transformation rule, which is equivalent to a rewriting rule: $\langle w:\alpha_1:0\rangle \cdots \langle w:\alpha_k:0\rangle \to \langle w:\alpha_1:1\rangle \cdots \langle w:\alpha_k:1\rangle, \langle w:\beta_1:0\rangle \cdots \langle w:\beta_l:0\rangle$. In most cases we will have $k = 1$.

A list of transformation rules, derived from the axiomatization of the path operators of *CTL* (see [9]), is given below:

- $(\forall \Box_R)\alpha \mapsto \alpha,\ \Box_R(\forall \Box_R)\alpha$
- $(\forall \Diamond_R)\alpha \mapsto \alpha \vee \Box_R(\forall \Diamond_R)\alpha$
- $(\exists \Box_R)\alpha \mapsto \alpha,\ \Diamond_R(\exists \Box_R)\alpha$
- $(\exists \Diamond_R)\alpha \mapsto \alpha \vee \Diamond_R(\exists \Diamond_R)\alpha$
- $\alpha(\forall U_R)\beta \mapsto \beta \vee (\alpha \wedge \Box_R(\alpha(\forall U_R)\beta))$
- $\alpha(\exists U_R)\beta \mapsto \beta \vee (\alpha \wedge \Diamond_R(\alpha(\exists U_R)\beta))$

Transformation rules are systematically used to handle *definitions* (cf. 2.4); for example, we have a transformation rule $E\alpha \vdash K_1\alpha, \ldots, K_n\alpha$ for the common epistemic operator $E$ (cf. 2.3 and 2.4).[5] Rewriting or transformation rules are also used to introduce heuristics, in order to shorten the proofs; for example, we have the simplification rule $\Box_R\alpha,\ \Box_R \neg \alpha \rightarrow \bot$ if $R$ is taken to be functional.

Note that classical literals do not produce any formulas, so their labels can be systematically set to 1. Also, the above tableau rules make the set $\Gamma$ contain only labeled formulas $\langle w : \alpha : a \rangle$ for which $\alpha$ has $\neg$, $\vee$, $\Box_R$ or $\Diamond_R$ as its main connective. This corresponds to the usual 'present state' + 'rest of the sequence' decomposition used in linear-time logic [23, 58].

We now examine the three fundamental rules of TABLEAUX; they differ from rewriting and transformation rules since they deal with formulas belonging to different worlds, and since they may introduce modifications in the set **R** of relations. However, they are close to transformation rules because labels change from 0 to 1, new formulas get a label 0, and we keep track of the examined formulas. We take $W = \{w_0, w_1, \ldots, w_n, \ldots\}$ to be the set of possible worlds.

(r1) $\langle w_n : \Box_R\alpha : 0 \rangle \rightarrow \langle w_n : \Box_R\alpha : 1 \rangle,\ \{\langle w_p : \alpha : 0 \rangle / w_p \in R(w_n)\}$

(r2) $\langle w_n : \Diamond_R\alpha : 0 \rangle \rightarrow \langle w_n : \Diamond_R\alpha : 1 \rangle,\ \langle w_m : \alpha : 0 \rangle,\ \{\langle w_m : \beta : 0 \rangle / \langle w_n : \Box_R\beta : a \rangle \in \Gamma\}$

where $m$ is a new index for worlds ($m = 1 + \max\{p / \langle w_p : \beta' : b \rangle \in \Gamma\}$), and the relation $R(w_n, w_m)$ has been added to **R**,

(r3) if $f = \langle w_n : (\alpha_1 \vee \alpha_2 \vee \cdots \vee \alpha_k) : 0 \rangle$ is a labeled formula in $\Gamma$ ($k \geq 2$), then split the tableau $T = \langle \Gamma, \mathbf{R} \rangle$ into two tableaux $T_1 = \langle \Gamma_1, \mathbf{R} \rangle$ and $T_2 = \langle \Gamma_2, \mathbf{R} \rangle$ with

$$\Gamma_1 = \Gamma' \cup \{\langle w_n : \alpha_1 : 0 \rangle\},$$

$$\Gamma_2 = \Gamma' \cup \{\langle w_n : (\alpha_2 \vee \cdots \alpha_k) : 0 \rangle\},$$

where $\Gamma'$ is a copy of $\Gamma$ with the label of $f$ changed from 0 to 1.

Rule (r1) is a propagation rule for $\Box_R$, rule (r2) is a rule for creating new worlds and rule (r3), which is *recursive*, is the rule for alternatives, creating or-branches in the proof construction. In (r3), the tableau $T$ is closed iff both $T_1$ and $T_2$ are closed, so if $T_1$ is chosen first and turns out to be not closed, it is not necessary to develop $T_2$ (since $T$ is already not closed). Note that (r3) applies to 'real' $\vee$-formulas only,

i.e. those which cannot be reduced by a simplification/resolution rule. Note also that it is of some importance, for efficiency considerations, to control the order of application of these rules.

Some of the semantic properties of $R$ are completely taken into account during the computation of cuts (i.e. of the sets $R(w_n)$), namely quasi-reflexivity, reflexivity, transitivity and euclideanity. For the remaining cases, further rules are needed:

- if $R$ is *symmetric* and a new relation $R(w, w')$ is added to **R**, then $R(w', w)$ is added too.
- if $R$ is *serial*, the following additional rule (r'1) is needed:
  If we encounter a formula $\langle w_n : \Box_R \alpha : 0 \rangle$ and if (i) $R(w_n)$ is empty and (ii) $\Gamma$ contains no $\Diamond$-formula of the form $\langle w_n : \Diamond_R \beta : 0 \rangle$, then add a new world $w_m$ (as defined in (r2)) to $R(w_n)$ before applying rule (r1).
- If $R$ is *quasi-functional*, then $R(w_n)$ has at most one element, so rule (r2) must apply only if $R(w_n)$ is empty; otherwise, if $R(w_n)$ already contains an element $w_m$, then rule (r2) is weakened into a simple propagation of $\alpha$ into world $w_m$:

$$\langle w_n : \Diamond_R \alpha : 0 \rangle \rightarrow \langle w_n : \Diamond_R \alpha : 1 \rangle, \langle w_m : \alpha : 0 \rangle$$

- If $R$ is *forwards-linear*, rule (r2) must again be applied only if $R(w_n)$ is empty; otherwise, we must proceed as follows: suppose we have a $\Diamond$-formula $\langle w_n : \Diamond_R \alpha : 0 \rangle$ in $\Gamma$, and let $w_u$ be a world which is *minimal* for $R$ in $R(w_n)$, i.e. $w_u \in R(w_n)$ is such that for all worlds $w_p \neq w_u$ in $R(w_n)$ we have $R(w_u, w_p)$.[6] Since rule (r2) usually introduces a new world $w_m$ in $R(w_n)$, and since $R$ is forwards-linear, three cases must be considered: (i) $w_m$ is to be inserted *between* $w_n$ and $w_u$, (ii) $w_m = w_u$ and (iii) $w_m$ is to be inserted *beyond* $w_u$. In case (i) $w_m$ is indeed created and becomes a world closer to $w_n$ than $w_u$ and we impose $R(w_n) \subseteq R(w_m)$; in case (ii) the formula $\alpha$ is propagated into $w_u$ as in the quasi-functionality case above; in case (iii) world $w_u$ remains a 'minimal' world in $R(w_n)$, and $\Diamond_R \alpha$ is propagated into $w_u$ (so that its treatment is postponed; this rule is recursive). No world $w_m$ is created in cases (ii) and (iii).

  Thus, when $R$ is forwards-linear, we have a new rule (r'2) for $\Diamond$-formulas in case where $R(w_n)$ is not empty. This rule creates three alternatives, corresponding to cases (i) (ii) and (iii) described above; therefore, it requires some computation.
- In case of the Gödel system $G$, $R$ is transitive and has the property of having no infinite $R$-chains. To handle this latter property, a slight modification of rule (r2) is needed ([21]): if $w_m$ is the new world introduced by $\langle w_n : \Diamond_R \alpha : 0 \rangle$, then also add $\langle w_m : \Box_R \neg \alpha : 0 \rangle$ in the right-side of rule (r2). This simple rule guarantees that $\Diamond_R \alpha$ will not start (at $w_n$) an infinite $R$-chain of worlds containing $\alpha$.
- All dual properties for the converse relation $R^{-1}$ are treated as described above, changing $R$ into $R^{-1}$. This includes the case of $R$ being backwards-linear.

As can be seen, one (or possibly several) tableaux rule(s) correspond(s) to a given semantical property of $R$. These rules are *encoded* in TABLEAUX as procedures, and if we were to extend the system with new semantic properties (and new modal

systems), we would have to exhibit and incorporate the corresponding procedures in the system. This is certainly one limit to the modularity of the tableaux method approach, though the problem is the same for other proof methods. Fortunately, the semantic properties considered above, which are covered by TABLEAUX, seem to be the most useful ones.

### 3.4. UNWINDING

Transitive-closure operators may generate infinite tableaux, by repeated applications of rule (r2). For example, the tableau construction for testing the (non-valid) formula $(\forall \Diamond_R)p$ yields the infinite sequence shown in Figure a:

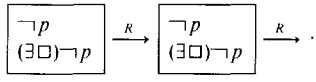



Fig. a

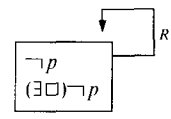



Fig. b.

Thus, the following rule is needed: each time we are about to create a new world $w$, labeled by a set of formula $\Gamma_w$, we check if there is an existing world $w'$ such that $\Gamma_w \subseteq \Gamma_{w'}$ (this inclusion does not take labels into account). If such a $w'$ exists, then we identify $w$ with $w'$, i.e. we do not create $w$ and we attach to $w'$ all the edges that would have been attached to $w$ instead. Since the total number of distinct sub-formulas which may appear in the tableau construction must be finite (because they are sub-formulas of the input formula(s)), the unwinding test guarantees that tableaux are always finite.[7]

In the previous example, the unwinding test succeeds and yields the finite tableau shown in Figure b above, which is a counter-model (representing an infinite sequence of '$\neg p$') for the input formula $(\forall \Diamond_R)p$. Note that the unwinding test is needed whenever transitivity is considered (which is the case for path operators), and is also known as the *rule of repeating chains* [30]. In fact, further complications appear for this unwinding test when using converse operators (dealing with the past), we omit details here.

### 3.5. SATISFIABILITY OF PATH FORMULAS

For modal systems without path operators, an *unsatisfiable* (labeled) formula is simply an inconsistent one (see 3.2), and this definition, along with the unwinding test described above, guarantees the completeness of the tableaux method. In case where path operators are considered, additional rules must be given to check unsatisfiability; we give the $(\forall \Box_R)$ and $(\exists \Box_R)$ cases below:

- a labeled formula $\langle w_n : (\forall \Box_R)\alpha : a \rangle$ is unsatisfiable iff there exists an $R$-path $c$ starting at $w_n$ and a world $w_p$ on $c$ such that $\langle w_p : \neg\alpha : b \rangle$ is in $\Gamma$.

- a labeled formula $\langle w_n : (\exists \Box_R)\alpha : a \rangle$ is unsatisfiable iff for every $R$-path $c$ starting at $w_n$ there exist a world $w_p$ on $c$ such that $\langle w_p : \neg\alpha : b \rangle$ is in $\Gamma$.

These rules are direct consequences of the semantic definitions of the operators $(\forall \Box_R)$ and $(\exists \Box_R)$ mentioned in Section 2.2. Similar rules for 'until' operators must also be considered. We refer the reader to [4] or [58] for more detail. Note that these rules, which require some computation, are to be applied only when the construction of the tableau has been finished and when no basic inconsistency (i.e. opposite formulas) has been found.

### 3.6 FINAL REMARKS

- The system of tableaux rules we have described avoids the problem of exploring a graph, as in the classical tableaux method, i.e. manipulating nodes (labeled by sets of formulas), finding non-cyclic paths in a graph, jumping from node to node, etc. [51]. Our tableau rules are only *production rules*: existing formulas create new formulas, and the process halts when an inconsistency is encountered.
- The generality of the system is certainly a handicap, because multiple tests concerning the semantic properties of $R$ are involved at every step of the tableau construction. Restricting the method to particular modal systems would lead to more efficient computations.
- As far as efficiency is concerned, the difficult points are *alternatives* (or-branches), *unwinding tests* and detection of unsatisfiable formulas. In TABLEAUX, as mentioned before, alternatives are reduced by using reductions of disjunctions (i.e. resolution), especially during computations of unions $\{f\} \cup \Gamma$ when a new labeled formula $f$ is to be added to $\Gamma$.

## 4. Examples and Experimental Results

TABLEAUX is implemented in VM/Prolog [47] and runs on IBM-370 machines. Its many full-screen facilities make it easy to use. The system has been tried on most of the axioms or theorems one can find in the various areas involving modal logics, and appears to be of reasonable complexity for current examples. Some of them, expressing valid formulas, are indicated in the chart opposite.

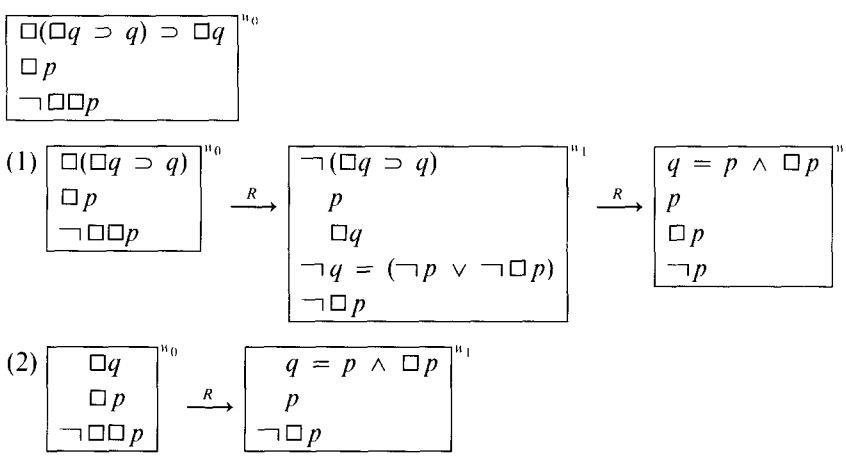
'Or-B' means 'Or-Branches', so the last column indicates the number of alternative tableaux created. Times are given in milliseconds.[8]

Example (1) shows that the Gödel axiom $\Box(\Box\alpha \supset \alpha) \supset \Box\alpha$ implies transitivity ([21] p. 248). Example (2) is axiom **D2** of ([30] p. 261), which holds in $S4.3$. Example (3) shows that, in the system $K$ of branching time logic, the Aristotelian/Megarian modality $\Box$ (already mentioned in Section 2.4. as $\Delta$) belongs to an extension of the Brouwersche system $B$ (reflexitivity and symmetry) containing the axiom $\Box\Box\alpha \supset \Box\Box\Box\alpha$ ([49] p. 129). Example (4) is McCarthy's famous three-wise-men puzzle of epistemic logic, where $p_i$ means 'The $i$th wise man has a white hat', $K_1$, $K_2$,

| | Formula | System | Time | Or-B |
|---|---|---|---|---|
| 1 | $(\Box(\Box q \supset q) \supset \Box q) \supset (\Box p \supset \Box\Box p)$ where $q = (p \wedge \Box p)$ | $K$ | 110 | 2 |
| 2 | $\Box(\Box p \supset \Box q) \vee \Box(\Box q \supset \Box p)$ | $S4.3$ | 230 | 5 |
| 3 | $\Box\Box p \supset \Box\Box\Box p$ where $\Box p = (Hp \wedge p \wedge Gp)$ | $K_h$ | 630 | 8 |
| 4 | $(p_1 \wedge p_2 \wedge p_3$ $\wedge C(p_1 \vee p_2 \vee p_3)$ $\wedge C(p_i \supset (K_j p_i \wedge K_k p_i))\ \ i \neq j, i \neq k$ $\wedge C(\neg p_i \supset (K_j \neg p_i \wedge K_k \neg p_i))\ \ i \neq j, i \neq k$ $\wedge C\neg K_1 p_1$ $\wedge C\neg K_2 p_2) \supset K_3 p_3$ | $K$ | 900 | 0 |
| 5 | $\Box(p \supset \Diamond \bigcirc p) \supset (\Diamond p \supset \Box \Diamond p)$ | $PTL$ | 1730 | 10 |
| 6 | $LUA(p, q) \wedge LUA(q, r) \supset LUA(p \vee q, r)$ | $PTL$ | 22 100 | 66 |

$K_3$ are the knowledge (or belief) operators and $C$ is the common knowledge (or belief) operator. Note that system $K$ is sufficient to solve the problem. Example (5) is a valid formula in $PTL$ (see 2.2), where $\Box$ is the transitive (but not reflexive) closure operator of $\bigcirc$. Example (6) is taken from [44], where $LUA$ is defined by $LUA(\alpha, \beta) \equiv_{\text{Def}} LU(\alpha, \alpha \wedge \beta)$ and $LU$ is defined by $LU(\alpha, \beta) \equiv_{\text{Def}} U(\neg\alpha, U(\alpha \wedge \neg\beta, \beta))$ in $PTL$.

To illustrate the output result of TABLEAUX, example (1) yields the following tableau construction (where $q$ is still an abbreviation for $p \wedge \Box p$):




where (1) and (2) indicates the or-branches generated by the $\vee$-formula $\Box(\Box q \supset q) \supset \Box q$ in world $w_0$. Note that the tableau is closed because both branches are closed: branch (1) is closed because $w_2$ is and branch (2) is closed because $w_1$ is.

A complete list of test examples is also given in the chart on the next page. In the language of basic modal logic (operators $\Box$ and $\Diamond$), 31 formulas have been considered,

| N° | Name | Formula | K | K4 | G | KH | K4H | KD | KD4 | KDB | KD5 | KDQ | KT | KT4 | S4.3 | KTB | S5 | KD4$ | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **K** | $\Box(p \supset q) \supset (\Box p \supset \Box q)$ | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 630 |
| 2 | **D** | $\Box p \supset \Diamond p$ | | | | | | x | x | x | x | x | x | x | x | x | x | x | 400 |
| 3 | **T** | $\Box p \supset p$ | | | | | | | | | | | x | x | x | x | x | | 300 |
| 4 | **4** | $\Box p \supset \Box\Box p$ | | x | x | | x | | x | | | | | x | x | | x | x | 630 |
| 5 | **B** | $p \supset \Box\Diamond p$ | | | | | | | | x | | | | | | x | x | | 500 |
| 6 | **5** | $\Diamond p \supset \Box\Diamond p$ | | | | | | | | | x | | | | | | x | x | 750 |
| 7 | **R** | $(\Box p \wedge \Box q) \equiv \Box(p \wedge q)$ | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 2520 |
| 8 | **W** | $\Box(\Box p \supset p) \supset \Box p$ | | | x | | | | | | | | | | | | | | 630 |
| 9 | **X** | $\Box(\Diamond p \supset q) \supset (p \supset \Box q)$ | | | | | | | | x | | | | | | x | x | | 1070 |
| 10 | | $\Diamond\Box(\Diamond p \supset \Box\Diamond p)$ | | | | | | | x | | x | | | x | x | | x | x | 1620 |
| 11 | | $\Box(p \supset \Box(q \supset r)) \supset \Diamond(q \supset (\Box p \supset \Diamond r))$ | | | | | | | x | | x | | x | x | x | x | x | x | 1300 |
| 12 | | $\Box(p \vee \Diamond q) \supset (\Box p \vee \Diamond q)$ | | x | x | | x | | x | | | | | x | x | | x | x | 1000 |
| 13 | | $\Diamond p \wedge \Diamond q \supset \Diamond(p \wedge \Diamond q) \wedge \Diamond(\Diamond p \wedge q)$ | | | | | | | | | x | | | | | | x | x | 2340 |
| 14 | **H** | $\Diamond p \wedge \Diamond q \supset \Diamond(p \wedge \Diamond q) \vee \Diamond(\Diamond p \wedge q) \vee \Diamond(p \wedge q)$ | | | | x | x | | | | x | x | | | x | | x | x | 2260 |
| 15 | **Q** | $\Diamond p \supset \Box p$ | | | | | | | | | | x | | | | | | | 560 |
| 16 | | $\Box\Diamond p \supset \Diamond\Box p$ | | | | | | | | | | x | | | | | | | 1500 |
| 17 | **G** | $\Diamond\Box p \supset \Box\Diamond p$ | | | | | | | | x | x | x | | | x | x | x | x | 960 |
| 18 | **D2** | $\Box(\Box p \supset \Box q) \vee \Box(\Box q \supset \Box p)$ | | | | | x | | | | x | x | | | x | | x | x | 1720 |
| 19 | **Grz** | $\Box(\Box(p \supset \Box p) \supset p) \supset p$ | | | | | | | | | | | | | | | | | 870 |
| 20 | **Dum** | $\Box(\Box(p \supset \Box p) \supset p) \supset (\Diamond\Box p \supset p)$ | | | | | | | | x | | | | | | x | x | | 1630 |
| 21 | **R1** | $(p \wedge \Diamond\Box p) \supset \Box p$ | | | | | | | | | x | | | | | | x | x | 800 |
| 22 | **Den** | $\Diamond p \supset \Diamond\Diamond p$ | | | | | | | | | x | | x | x | x | x | x | x | 570 |
| 23 | **FMP** | $(\Box p \wedge \neg\Box\Box p) \supset \Diamond(\Box\Box p \wedge \neg\Box\Box\Box p)$ | | x | x | | x | | x | | | | | x | x | | x | x | 1830 |
| 24 | | $\Box\Box p \equiv \Box p$ | | | | | | | | | | | | x | x | | x | x | 1350 |
| 25 | | $\Diamond\Box p \equiv \Box p$ | | | | | | | | | | | | | | | x | x | 1380 |
| 26 | | $\Box\Diamond\Box\Diamond p \equiv \Box\Diamond p$ | | x | x | | x | | x | x | | | | x | x | x | x | x | 3400 |
| 27 | | $\Box\Box\Box p \equiv \Box\Box p$ | | | | | | | | | x | | | x | x | | x | x | 1820 |
| 28 | | $\Box\Diamond\Box p \equiv \Box\Box p$ | | | | | | | | | x | | | | | | x | x | 2500 |
| 29 | | $\Diamond\Box\Box p \equiv \Diamond\Box p$ | | x | x | | x | | x | | x | | | x | | | x | x | 2920 |
| 30 | | $\Diamond\Diamond\Box p \equiv \Diamond\Box p$ | | | | | | | x | | x | | | x | | | x | x | 2550 |
| 31 | | $\Box\Box p \equiv \Diamond\Box p$ | | | | | | | | | x | x | | | | | x | x | 1730 |
| | | Total time / Logic | 1680 | 2220 | 2150 | 1820 | 2420 | 2430 | 3800 | 2400 | 2730 | 2310 | 2600 | 3120 | 4300 | 2850 | 3100 | 3550 | ≃42 s |

most of them being very classical (see 2.1). Each formula has been tested for validity in 16 distinct normal *K*-systems of modal logic (note that *KH* denotes the system $K$ + axiom **H**, *K4H* is *KH* + axiom **4**, and *KDQ* is $K$ + axioms **D** and **F**; other systems use standard notations (see [11])). A cross at the intersection of the formula and the modal system indicates that the formula is valid in this system.

For each formula, the total computational time (in milliseconds) required by TABLEAUX to test validity in all the 16 systems is indicated in the last column of the chart. Also, for each system, the total computational time required to test the 31 formulas is indicated at the bottom of the chart. Finally, the total time required for testing the 31 formulas in the 16 systems (496 tests) is about 42 seconds. Of course, most formulas are very simple, so the above chart gives a more precise idea of the real complexity of the system.

## 5. Related Work

An implementation of the tableaux method is presented in [48] and [51], but only for few modal systems and without experimental results. A decision procedure for discrete linear time propositional temporal logic is presented in [44]; a tableaux method is used and an implementation is described, with good experimental results. A theorem prover based on tableaux for normal conditional logics is presented in [25], but it seems that TABLEAUX covers these systems (see 2.6).

The system HLM described in [42] is certainly the most closely related to ours; the general framework is almost identical, though some non-modal systems are also covered by HLM. Furthermore, HLM uses an elegant method for translating various modal systems into the framework of dynamic logic. However, HLM still uses the traditional graph representation for tableaux and only pre-defined modal systems (and notations) are permitted. Thus, in allowing arbitrary (simple) multimodal systems (see 2.7), TABLEAUX appears to be more general.

Other semantic proof methods are automata techniques [17, 39, 55], which are well suited to complex temporal systems, or the 'Model checker' approach [10, 12]. Note that the latter can easily be derived from our tableaux method, as done in [42]. We also mention the KRIPKE system [53], which is based on Gentzen-style proof procedures and oriented towards relevant logics.

Other methods for automated deduction have been proposed for modal logics, but generally with syntactical features only (i.e. with no kind of *models construction*, as in TABLEAUX); furthermore, few implementations seem to have been achieved. Extension of classical clausal-form resolution techniques [14, 18], or non-clausal resolution [1] have been defined; both are purely syntactic methods and are also associated with important complexity and control problems. A sequent-based proof method for first-order modal logics is proposed in [33] and [34] using indexed formulas; a similar idea is proposed in [13], though the latter method seems to be more general. The extension of Bibel's connection method to modal logic is presented in [56], and the technique seems very powerful (though only a few traditional modal systems are

considered); furthermore, it has been implemented. Attempts to define modal and temporal executable languages, in a logic programming perspective, can be found in [2], [23] and [34].

## 6. Conclusion

In our opinion, the most interesting features of modal logics are the *expressivity* and *modularity* of their languages and also their *possible-worlds semantics*, which is both general and intuitive. From this point of view, TABLEAUX provides a unified environment for studying and using these formalisms, since various types of modal languages and systems are allowed, and since automated deduction is achieved using the tableaux method, which is precisely oriented towards models construction. Furthermore, the method appears to be of reasonable efficiency for practical use on current examples.

Working with TABLEAUX suggests many possible topics for further research dealing with modal logics, such as expressiveness features, semantic characterizations of axioms or other automated proof methods for modal logics. For us, it has been a valuable tool for studying multimodal logics [8]. More generally, as mentioned in the introduction, we believe that prototypes like TABLEAUX are always very helpful for experimenting with new ideas.

## Notes

[1] A preliminary version of this paper appeared in the Proceedings of the International Computer Science Conference (ICSC'88), Hong-Kong, December 19–21, 1988.

[2] Note that $R$ and $R^{-1}$ are simultaneously reflexive, symmetric or transitive.

[3] Alternative notations are $\forall X$ and $\exists X$ (or $\forall \bigcirc$ and $\exists \bigcirc$) for $\Box$ and $\Diamond$ and $\forall G$, $\exists G$, $\exists F$, $\forall F$ for $\forall \Box$, $\exists \Box$, $\exists \Diamond$ and $\forall \Diamond$.

[4] This system is called *DUX* in [42], with operators $X$, $F$, $G$, $U$.

[5] Transformation rules may also be used to handle rational operators (i.e. temporal operators defined by a right-linear grammar) as defined in [57]. These rules are similar to the decomposition rules introduced by the author for these operators.

[6] World $w_u$ is to be seen as a 'closer' world to $w_n$. Note that $w_u$ exists because $R$ being forwards-linear implies that $R$ induces a linear relation on cuts; but $w_u$ may not be unique.

[7] In this test, the set $\Gamma_w$ has to be saturated by the simplification rules first; for example, disjunctions must be reduced as much as possible. This augments the chances of finding a candidate $w'$ such that $\Gamma_w \subseteq \Gamma_{w'}$.

[8] Note that, using a Prolog compiler, these times should be divided by at least a factor 2.

## References

1. Abadi, M. and Manna, Z., 'Modal theorem proving', *Lecture Notes in Computer Science* Vol. 230 (1986).
2. Abadi, M. and Manna, Z., 'Temporal logic programming', *Proc. IEEE Symposium on Logic Programming* (Sept. 1987), San Francisco, California, pp. 4–16.
3. Van Benthem, J., *The Logic of Time*, D. Reidel, Dordrecht (1980).
4. Ben-Ari, M., Manna, Z. and Pnueli, A., 'The temporal logic of branching time', *Proc. 8th Annual ACM Symposium on Principles of Programming Languages* (1981), pp 164–176.
5. Bull, R. and Segerberg, K., 'Basic modal logic' in [28], pp. 1–88.

6. Burgess, J P , 'Decidability for branching time', *Studia Logica* **39** (2/3), (1980).
7. Catach, L., 'Normal multimodal logics', *Proc. AAAI'88*, pp. 491–495.
8 Catach, L., 'Les logiques multimodales' ('Multimodal logics'), Doctoral Thesis, IBM Paris Scientific Center and L I.T.P., Paris 6 University (1989).
9. Clarke, E M and Emerson, E. A., 'Design and synthesis of synchronization skeletons using branching time temporal logic', *Lecture Notes in Computer Science* Vol. 131 (1982), pp. 52–71.
10. Clarke, E M., Emerson, E. A., and Sistla, A. P., 'Automatic verification of finite-state concurrent systems using temporal logic specifications', *ACM Transactions on Programming Languages and Systems*, **8**(2) (1986)
11. Chellas, B F., *Modal Logic An Introduction*, Cambridge University Press (1980).
12 Cavalli, A. and Horn, F , 'Evaluation des spécifications formelles à l'aide des automates finis et de la logique temporelle', Report L.I.T.P. No. 86–74, Paris VI/Paris VII University, France (1986)
13. Enjalbert, P and Auffray, Y., 'Démonstration de théorèmes en logique modale: un point de vue équationnel', *European Workshop on Logical Methods in Artificial Intelligence* (JELIA'88), Roscoff, France (1988).
14 Enjalbert, P. and Fariñas, L., 'Modal Resolution in clausal form', Report No. RG 14–86, Greco de Programmation, Université de Bordeaux I, France (1986)
15 Emerson, E. A and Halpern, J. Y. 'Decision procedures and expressiveness in the temporal logic of branching time', *Journal of Computer and System Sciences* **30**, 1–24 (1985)
16. Emerson, E. A., 'Automata, tableaux and temporal logics', *Lecture Notes in Computer Science* Vol 193 (1985), pp. 79–87.
17 Emerson, E A. and Sistla, A. P., 'Deciding full branching time logic', *Information and Control* **61**, 175–201 (1984).
18. Fariñas Del Cerro, L., 'Resolution modal logic', *Logique et Analyse* 110–111 (1985)
19 Fischer, M J and Immerman, N , 'Interpreting logics of knowledge in propositional dynamic logic with converse', *Information Processing Letters* **25** 175–181 (1987)
20. Fitting, M , 'Tableaux methods of proof for modal logics', *Notre-Dame Journal of Formal Logic* **13** 237–247 (1972)
21 Fitting, M., 'Proof methods for modal and intuitionistic logics', Reidel, *Synthese Library* Vol. 169 (1983)
22. Fischer, M. J. and Ladner, R E., 'Propositional dynamic logic of regular programs, *Journal of Computer and System Sciences* **18** 194–211 (1979).
23. Fujita, M. *et al.* 'Tokyo: logic programming language based on temporal logic', *Proc. 3th International Conference on Logic Programming*, London pp. 695–709 (1986)
24. Gabbay, D. M., 'Modal and temporal logic programming', in *Temporal Logics*, A. Galton (ed.), Academic Press (1988).
25. Groeneboer, C and Delgande, J. P , 'Tableau-based theorem proving in normal conditional logics', *Proc. AAAI'88*, pp. 171–176.
26. Gurevich, Y. and Shelah, S., 'The decision problem for Branching Time Logic', *Journal of Symbolic Logic* **50**(3) (1985).
27. Halpern, J Y , 'Reasoning about knowledge. an overview', *Proc. Conference on Theoretical Aspects of Reasoning about Knowledge* (J. Y Halpern (ed.), Morgan Kauffmann (1986)
28. 'Extensions of classical logic', *Handbook of Philosophical Logic* Vol II, D Gabbay and F Guenthner (eds ), D. Reidel Publishing Company (1984).
29 Harel, D , 'Dynamic logic', in [28], pp. 497–604.
30. Hughes, G E and Cresswell, M. J., *An Introduction to Modal Logic*, Methuen, London (1968).
31 Halpern, J. Y and Moses, Y., 'A guide to the modal logics of knowledge and belief', *Proc IJCAI* pp. 480–490 (1985).
32. Halpern, J Y and Shoham, Y , 'A propositional modal logic of time intervals', *Proc IEEE Symposium on Logic in Computer Science* Vol. 1 pp. 279–292 (1986).
33. Jackson, P and Reichgelt, H., 'A general proof method for first-order modal logic', *Proc IJCAI* pp 942–944 (1987).
34. Jackson, P. and Reichgelt, H. 'A general proof method for modal predicate logic without the Barcan formula', *Proc. AAAI*88, pp. 177–181 (1988).
35. Konolige, K., 'Resolution and quantified epistemic logics', *Lectures Notes in Computer Science* Vol 230 pp 199–208 (1986).

36. Ladner, R. E., 'The computational complexity of provability in systems of modal propositional logic', *SIAM J. Computing* **6**(3) 467–480 (1977).
37. Lehmann, D. and Kraus, S., 'Knowledge, belief and time', *Lecture Notes in Computer Science*, Vol. 226 pp. 186–195 (1986).
38. Lafon, E. and Schwind, C. B , 'A theorem prover for action performance', *Proc. ECAI'88* (1988).
39. Michel, M., 'Computation of temporal operators', *Logique et Analyse* **110/111** (1985).
40. Moszkowski, B., *Executing temporal logic of programs*, Cambridge University Press (1986).
41. Manna, Z. and Wolper, P , 'Synthesis of communicating processes from temporal logic specifications', Report No. STAN-CS-81-872, Stanford University, Dept. of Computer Science (1981)
42. Niemela, I. and Tuominen, H., 'Helsinki logic machine· a system for logical expertise', Technical report Series B (No 1), Helsinki University of Technology, Digital Systems Laboratory (1987).
43 Parikh, R., 'Propositional dynamic logics of programs· a survey', *Lecture Notes in Computer Science* Vol 125 pp. 102–144 (1981).
44. Plaisted, D. A., 'A decision procedure for combinations of propositional temporal logic and other specialized theories', *J. Automated Reasoning* **2**, 171–190 (1986).
45. Pnueli, A., 'The temporal logic of programs', *Proc. 18th IEEE Symposium on Foundations of Computer Science* pp. 46–57 (1977).
46. Pratt, V. R., 'Models of program logics', *Proc. 20th IEEE Symposium on Foundations of Computer Science* (1978).
47. IBM, VM/Programming in Logic (VM/Prolog), Program Offering 5785-ABH.
48. Roche, Y., 'Implémentation d'un démonstrateur de théorèmes pour les logiques modales et temporelles en Prolog', Report GIA-GRTC, Luminy University, Marseille, France (1985).
49. Rescher, N. and Urquhart, A., *Temporal Logic*, Springer-Verlag (1971).
50. Sistla, A. P and Clarke, E. M., 'The complexity of Propositional linear temporal logics', *J. ACM* **32**(3) (1985).
51. Schwind, C., 'Un démonstrateur de théorèmes pour des logiques modales et temporelles en Prolog', *Proc. 5th. Congrès Reconnaissance des Formes et Intelligence Artificielle* (RFIA), France pp. 897–914 (1985).
52. Smullyan, R. M., *First-Order Logic*, Springer-Verlag (1968).
53. Thistlewaite, P. B., McRobbie, M. A., and Meyer, R. K., 'The KRIPKE automated theorem proving system', *Lecture Notes in Computer Science* Vol. 230 pp. 705–706 (1986).
54. Venema, Y., 'Expressiveness and completeness of an interval tense logic', Report, Institute for Language, Logic and Information, University of Amsterdam (1988).
55. Yardi, M. Y. and Wolper, P., 'Automata theoretic techniques for modal logics of programs', *Proc ACM Symposium on Theory of Computing* (1984).
56. Wallen, L. A., Matrix proof methods for modal logics', *Proc. IJCAI* pp. 917–923 (1987).
57. Wolper, P., 'Temporal logic can be more expressive', *Information and Control* **56** (1983).
58. Wolper, P., 'The tableaux method for temporal logic: an overview', *Logique et Analyse* **110–111** (1985)