On the Relative Complexity of Labelled Modal Tableaux

Guido Governatori

School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, QLD 4072, Australia
email: guido@itee.uq.edu.au

Abstract

We investigate the relative complexity of two free-variable labelled modal tableaux (KEM and Single Step Tableaux, SST). We discuss the reasons why p-simulation is not a proper measure of the relative complexity of tableaux-like proof systems, and we propose an improved comparison scale (p-search-simulation). Finally we show that KEM p-search-simulates SST while SST cannot p-search-simulate KEM.

1 Introduction

In the last few years several comparisons (competitions) of theorem provers for modal logic have been held (cf. [2,16,18]) and experimental research has been carried out (cf. [13,12]). Despite the potential interest for eventual applications, we believe that this kind of research provided little or no insight on better theoretical architectures for modal theorem provers. Very often the overall performance is heavily influenced by external factors such as, for example, language specific optimisations of the implementation.

In this paper we perform a theoretical comparison of two labelled tableaux for modal logic. This means that we do not consider implementation issues, but only logical ones; in particular, we are not interested in the propositional features and in the interaction of modal operators and propositional connectives, but only in the modal characteristics.

To prove that a proof system $\mathcal{A}$ is essentially better than a proof system $\mathcal{B}$ we have to exhibit at least one formula (or a class of formulas) for which $\mathcal{A}$ is better than $\mathcal{B}$, and for all formulas $\mathcal{A}$ is not essentially worse than $\mathcal{B}$.[1] There are many distinct modal logics, and it is possible that the result in one logic does not apply to a different logic. Moreover $\mathcal{A}$ may cover some modal

[1] We shall give a precise definition of what "better" and "worse" mean in this context in Section 4.

logics which are not covered by $\mathcal{B}$ and the other way around. However, any general purpose modal theorem prover should cover the basic fifteen normal modal logics. Among them, some offer too simple modal structures while other lend themselves to specialised optimisation procedures (in particular the logics with a finite number of distinct modalities). In both cases, these logics do not provide the better scenario to really test the theoretical architecture behind a modal theorem prover. Therefore we have to identify a modal logic with the following properties:

(i) it is one of the basic fifteen normal modal logics;

(ii) the proof procedures are modular for both systems, that is, they are the combination of the proof procedures of the single components of the logic; and

(iii) there are no specialised proof procedures.

As we shall see the basic normal modal logic DB satisfies the criteria listed above to be a representative candidate to test the capability of a theorem prover for modal logic. In the rest of the paper we shall concentrate on this logic. Briefly, DB is the normal modal logic obtained from the basic modal logic K by adding the axiom D ($\Box A \to \Diamond A$), which characterises the class of serial frames, and the axiom B ($A \to \Box\Diamond A$), which characterises the class of symmetric frames. Thus DB is semantically characterised by the class of serial and symmetric frames. [2] Moreover, due to some well-known difficulties [7], symmetric logics lie outside most of the current modal theorem prover methods (e.g., Wallen's matrix proof method [23]), though they play an important role in non-monotonic reasoning [14], knowledge representation [20], and information systems [21].

The paper is organised as follows: in Section 2 we briefly describe the basic ideas of labelled tableaux and we introduce the systems at hand (Section 2.2 and Section 2.3). Then, in Section 3, we investigate the complexity of KEM unification mechanisms; this will enable us to compare theoretically the relative computational complexity of the two systems (Section 4).

2 Labelled Modal Tableaux

Since the seminal work by Fitch [6] labels have been widely used in modal logic to simulate possible world semantics in the proof theory to improve, simplify and speed up proofs. Usually the main function of labels is to import semantic structures in the object language. Accordingly, in semantic based proof methods (cf., among others, [19]), labels represent possible worlds and accessibility relations (using sequences of atomic labels) in Kripke models.

Semantic tableaux (cf., [22]) is one of the most common form of semantic

[2] Let $\langle W, R\rangle$ be a frame, where W is a set of possible worlds and R is a binary relation over W. A frame is serial iff $\forall x \in W \exists y \in W(xRy)$, and symmetric iff $\forall x, y \in W(xRy \Rightarrow yRx)$.

based proof procedures and, we believe, it offers the best format for the use of labels. The basic idea is to supplement the object language with a label language and a label algebra. The basic entities of labelled deductions are labelled formulas, i.e., expressions of the form $A : x$, where x is a label drawn form the label language, and A (the declarative unit) is a well-formed formula of the logic at hand (cf., [8]). Intuitively the meaning of a labelled formula such as $A : x$ is that the declarative unit (A) is true at the world(s) denoted be the label x.

In most systems (new) labelled formulas are generated from previous formulas using inference rules that closely resemble the semantic evaluation of the premises. Given the semantic conditions, it is indeed possible that the conclusion of a premise holds in a set of possible worlds instead of a single worlds; for example, just consider the semantic clause for $\Box A$ which requires A to be true in all worlds accessible from the world where $\Box A$ is evaluated. We have two alternative ways for representing such conclusions using labels:

(i) we can use ground labels and generate all possible/relevant instances of such worlds;

(ii) we can use a label with a free-variable.

In what follows we shall compare two labelled tableaux systems (KEM and SST) using labels with free-variables.

Before presenting the two systems we give some common notions. As is well known semantic tableaux calculus is a refutation proof method. Therefore a proof of A is a failed attempt to provide a model for $\neg A$. A tableaux for a formula A is a (binary) tree whose root is $A : i_0$ where i_0 is the initial label, and the nodes are derived from previous nodes according to the inference rules of the system. A branch is *closed* iff it contains a pair of complementary formulas (the notion of complementary formulas may vary from system to system), otherwise it is *open*; a tree is *closed* iff every branch in it is closed. Finally a proof of A is a closed tree with root $\neg A : i_0$. A tree is *complete* iff every rule that can be applied has been applied.

2.1 Label Formalism

In this section we present the KEM label formalism, which will be also used for SST. In fact most of the differences between the formalisms of the two systems are just notational ones, and the differences that are not notational are not relevant for the present investigation.

KEM has two basic kinds of atomic labels: variables and constants. Formally, let $\Phi_C = \{w_1, w_2, \dots\}$ and $\Phi_V = \{W_1, W_2, \dots\}$ be two arbitrary sets of *atomic labels*: the set of *constant world-symbols* (or simply *constants*) and the set of *variable world-symbols* (or simply *variables*). A *label* is then an element of the set of labels $\Im$ defined as follows:

Definition 2.1 $\Im = \bigcup_{1 \leq p} \Im_p$ where $\Im_p$ is:

$\Im_1 = \Phi_C \cup \Phi_V$
$\Im_2 = \Im_1 \times \Phi_C$
$\Im_{n+1} = \Im_1 \times \Im_n,\ n > 1$.

Thus, a label i is either a variable or a constant or a "structured" sequence of atomic labels. For a structured label $i = (k', k)$ we have the following cases: (i) k' is an atomic world-symbol and (ii) $k \in \Phi_C$ or $k = (m', m)$ where (m', m) is a label. As we have alluded to in the previous section, we may think of constant and variable world-symbols as denoting respectively worlds and sets of worlds in a standard Kripke setting. A label of the form (k', k) is called a "world-path". For instance, the label (W_1, w_1) represents a path from w_1 to the set W_1 of worlds accessible from w_1; $(w_2, (W_1, w_1))$ represents a path which takes us to a world w_2 accessible by any world accessible from w_1 (i.e., accessible by the sub-path (W_1, w_1)) according to the appropriate accessibility relation. Thus a label of the form (k', k) is "structurally" designed to record information about the accessibility relation when we move from a label (a world or a set of worlds) to another label. We define the length of a label i, $\ell(i)$, as the number of atomic labels in i. From now on we shall use $i, j, k, \ldots$ to denote arbitrary labels.

Definition 2.2 For a label $i = (j, k)$, we shall call j the *head* and k the *body* of i, and denote them by $h(i)$ and $b(i)$ respectively.

The notions of body and head are obviously recursive (they can be defined as projection functions), and allow us to identify any sub-label of a given label; thus, if $b(i)$ denotes the body of i, then $b(b(i))$ will denote the body of $b(i)$, $b(b(b(i)))$ will denote the body of $b(b(i))$, and so on. We call each of $b(i)$, $b(b(i))$, etc., a *segment* of i. Let $s(i)$ denote any segment of i (obviously, by definition every segment $s(i)$ of a label i is a label); $h(s(i))$ will denote the head of $s(i)$. With $s^n(i)$ we will denote the segment of i of length n, i.e., $s^n(i) = s(i)$ such that $\ell(s(i)) = n$. We shall use $h^n(i)$ as an abbreviation for $h(s^n(i))$.

Definition 2.3 For any label $i, \ell(i) \geq n$, we define the *countersegment-n* of i, as follows:

$$c^n(i) = h(i) \times (\cdots \times (h^k(i) \times (\cdots \times (h^{n+1}(i), w_0)))) \text{ for } n < k < \ell(i)$$

where w_0 is a dummy label, i.e., a label not appearing in i (the context in which such a notion is applied will tell us what w_0 stands for).

If $n = \ell(i)$ we have that $c^n(i) = w_0$, and $s^n(i) = i$.

Example 2.4 If $i = (w_4, (W_3, (w_3, (W_2, w_1))))$, then $\ell(i) = 5$, $h^3(i) = w_3$, $s^3(i) = (w_3, (W_2, w_1))$, and its countersegment-3 is $c^3(i) = (w_4, (W_3, w_0))$; intuitively $c^n(i)$, is what remains of i after deleting $s^n(i)$.

To clarify the notion of countersegment, which will be used frequently in this work, we present, in the following table the list of the segments of i in the left-hand column and the relative countersegments in the right-hand column.

$s^1(i) = w_1$	$c^1(i) = (w_4, (W_3, (w_3, (W_2, w_0))))$
$s^2(i) = (W_2, w_1)$	$c^2(i) = (w_4, (W_3, (w_3, w_0)))$
$s^3(i) = (w_3, (W_2, w_1))$	$c^3(i) = (w_4, (W_3, w_0))$
$s^4(i) = (W_3, (w_3, (W_2, w_1)))$	$c^4(i) = (w_4, w_0)$
$s^5(i) = i$	$c^5(i) = w_0$

The dummy label w_0 is considered as an atomic label, and it is used to encapsulated complex labels into an atomic one.

2.2 KEM

KEM (see [1,10,9]) is a labelled analytic proof system based on a combination of tableau and natural deduction inference rules which allows for a suitably restricted ("analytic") application of the cut rule and a specialised, yet modular, unification mechanism for the labels.

2.2.1 Unifications

In the course of proofs labels are manipulated in a way closely related to the semantics of the logic under analysis. Labels are compared and matched using a specialised logic dependent unification mechanism. The notion of two labels i and j being unifiable means that the intersection of their denotations is not empty and that we can "move" to such a set of worlds through the path corresponding to the result of the unification of the two labels.

The definition of the unification appropriate for the logic DB (or logic unification) is carried out in several steps with the help of several auxiliary notions of unification.

First we have to provide the foundation of our unification (σ-unification). The basic unification is defined, as usual, in terms of a substitution, then we use the basic unification to define the unifications corresponding to the various modal axioms (axiom unifications); in the same way a modal logic is obtained by combining several axioms we combine the axiom unifications in combined unification. Finally we apply, in a recursive way, the combined unification to define the unification for the logic (logic unification).

Before presenting the formal machinery for the various unifications we have to give the notation used for them. Let L be a modal logic, and $\mathsf{A}_1, \dots, \mathsf{A}_n$ be the axioms of L. With σ^{A_i} we denote the unification for the axiom A_i; with $\sigma^{\mathsf{A}_1 \dots \mathsf{A}_n}$ the unification obtained from the combination of the σ^{A_i}-unifications; and with σ_{L} the unification for the logic L. Given two labels i and j and a unification σ^* we shall use $[i, j]\sigma^*$ to denote both the results of the σ^*-unification of i and j, and the fact that i and j σ^*-unify.

We are now ready to introduce the unifications for DB. For each unification we will provide the formal definition as well as a procedure to compute it. We begin with the notion of substitution.

Definition 2.5 A substitution is a mapping $\rho : \Im_1 \to \Im_1$ such that

$$\rho(i) = \begin{cases} i & i \in \Phi_C \\ j & \text{otherwise} \end{cases}$$

Accordingly we have that two atomic ("world") labels i and j σ-unify iff there is a substitution ρ such that $\rho(i) = \rho(j)$. The notion of σ-unification (or label unification) is extended to the case of composite labels (path labels) as follows:

Definition 2.6 Let $i, j \in \Im$

$$\begin{aligned} [i,j]\sigma = k \text{ iff } \exists \rho : h(k) = \rho(h(i)) = \rho(h(j)) \text{ and} \\ b(k) = [b(i), b(j)]\sigma \end{aligned}$$

Clearly σ is symmetric, i.e., $[i,j]\sigma$ iff $[j,i]\sigma$. Moreover this definition offers a flexible and powerful mechanism: in [11] we have shown that different classes of modal logics (in particular classes of non-normal modal logics such as regular and monotonic modal logics) are determined by conditions on the underlying substitution but the axiom unifications can be left unchanged. At the same time it allows for an independent computation of the elements of the result of the unification, and variables can be freely renamed without affecting the result of a unification, namely:

Proposition 2.7 *Let i and i' be two labels such that $i' = i(W_n/W_m)$ (i.e., i' has been obtained from i by replacing a variable by a different variable). $\forall j, k \in \Im$ if $[i,j]\sigma$, then $[i',j]\sigma$; and if $[[i,j]\sigma, k]\sigma$ then $[[i',j]\sigma, k]\sigma$.*

Proof. Tedious and mundane by case inspection.

The above proposition justifies the following simple algorithms to compute the unification of two world symbols and the unification of two labels.

Algorithm 1 *world-symbol-unification*(w, w')

```
begin
  if $w \in \Phi_V$
    then $[w,w']\sigma = w'$
    elseif $w' \in \Phi_V$ or $w' = w$
        then $[w,w']\sigma = w$
        else fail
    endif
  endif
end
```

We extend the algorithm to cover the case of arbitrary labels.

Algorithm 2 *basic-unification*(i, j)

```
begin
  if $\ell(i) = \ell(j)$
    then world-symbol-unification$(h(i), h(j))$ and
           basic-unification$(b(i), b(j))$
    else fail
  endif
end
```

It is worth noting that the procedure described in Algorithm 2 characterises the unification for the axiom D, and thus it is also part of DB.

Definition 2.8 Let $i, j \in \Im$

$$[i, j]\sigma^{\mathsf{D}} = [i, j]\sigma$$

The next step involves the definition of the unification corresponding to the axiom B or σ^{B}-unification.

Definition 2.9 Let $i, j \in \Im$

$$[i, j]\sigma^{\mathsf{B}} = \begin{cases} [s^{\ell(i)-2n}(i), j]\sigma \text{ if } h(i) \in \Phi_V \text{ and} \\ \qquad [h(i), h(j)]\sigma = [h^{\ell(i)-2n}(i), h(j)]\sigma \\ [i, s^{\ell(j)-2n}(j)]\sigma \text{ if } h(j) \in \Phi_V \text{ and} \\ \qquad [h(i), h(j)]\sigma = [h(i), h^{\ell(j)-2n}(j)]\sigma \end{cases}$$

Where $1 \leq n \leq V$, and $V = \ell(i) - m$, with m such that $\forall x, m \leq x \leq \ell(i), h^x(i) \in \Phi_V$.

The key idea of σ^{B}-unification is to match world symbols laying an even number of steps apart. The number of steps is given by the number of consecutive variables occurring in the head of the labels. If the head of a label is a variable we can go back by two steps. In general we are allowed to go back two steps for each variable. Accordingly labels like

$$(W_1, (w_2, w_1)) \qquad\qquad w_1$$

provide a simple instance of this unification. Intuitively W_1 denotes the set of worlds accessible from w_2, but, since w_2 is accessible from w_1, so, by symmetry, w_1 is one of the world accessible from w_2.

Example 2.10 Let us consider the labels

$$i = (W_3, (W_2, (w_2, (W_1, w_1)))) \qquad\qquad j = (W_4, (w_3, w_1)) \tag{1}$$

The labels i and j σ^{B}-unify since i has two variables, so we have two options for going back: one steps from $b(i)$, or two steps from $b(b(i))$. In the first case we have to see whether $(w_2, (W_1, w_1)) = s^{\ell(i)-2n}(i)$ for $n = 1$ and j σ-unify. In the second case the label that have to σ-unify with j is $w_1 = s^{\ell(i)-2n}(i)$ for $n = 2$. But in this case the unification fails.

Algorithm 3 *b-axiom-unification*(i, j)

```
begin
  if $\ell(i) > \ell(j)$ and $h(i) \in \Phi_V$
    then $m := i$ and $k := j$
    else if $\ell(j) > \ell(i)$ and $h(j) \in \Phi_V$
      then $m := j$ and $k := i$
      else fail
    endif
  endif
  if $(\ell(m) - \ell(k)) \mod 2 = 0$ and
    foreach $n = \frac{\ell(m) - \ell(k)}{2}$ upto $\ell(m)$
        $h^n(m) \in \Phi_V$
    then basic-unification$(s^{\ell(k)}(m), k)$
    else fail
  endif
end
```

Before introducing the main unification, the unification for the logic DB we introduce the combined unification (or σ^{DB}-unification) and the corresponding procedure, meant to verify whether two labels either σ- or σ^{B}-unify.

Definition 2.11 Let $i, j \in \Im$

$$[i, j]\sigma^{\mathsf{DB}} = \begin{cases} [i, j]\sigma^{\mathsf{D}} \\ [i, j]\sigma^{\mathsf{B}} \end{cases}$$

Algorithm 4 *db-axioms-unification*(i, j)

```
begin
  basic-unification$(i, j)$ or
  b-axiom-unification$(i, j)$
end
```

Finally we are ready to give the main unification for DB (or σ_{DB}), the unification which will be used with the inference rules.

Definition 2.12 Let $i, j \in \Im$

$$[i, j]\sigma_{\mathsf{DB}} = \begin{cases} [i, j]\sigma^{\mathsf{DB}} & \text{or} \\ & \\ [c^n(i), c^m(j)]\sigma_{\mathsf{DB}} & \exists n, m : \ 1 \le n \le \ell(i) \text{ and} \\ & \qquad\qquad 1 \le m \le \ell(j) \end{cases}$$

Notice that σ_{BD} has a recursive definition and it can be computed using the following algorithm.

Algorithm 5 *db-unification*(i, j)

```
begin
  db-axioms-unification$(i, j)$ or
  for $n = 1$ upto $\ell(i) - 1$
     for $m = 1$ upto $\ell(j) - 1$
        if db-axioms-unification$(w_0, c^{\ell(j)-m}(j))$
          then db-unification$(i, s^{\ell(j)-m}(j))$
        endif or
        if db-axioms-unification$(c^{\ell(i)-n}(i), w_0)$
          then db-unification$(s^{\ell(i)-n}(i), j)$
        endif or
        if db-axioms-unification$(c^{\ell(i)-n}(i), c^{\ell(j)-m}(j))$
          then db-unification$(s^{\ell(i)-n}(i), s^{\ell(j)-m}(j))$
        endif
     endfor
  endfor
end
```

Example 2.13 Let us consider the labels

$$i = (W_2, (w_3, (W_1, (w_2, w_1)))) \qquad\qquad j = w_1 \tag{2}$$

It is easy to see that the above labels σ_{DB}-unify, with a recursive application of the unification. In fact we have that $[c^3(i), w_0]\sigma^{\mathsf{DB}}$, where $j = w_0 = [s^3(i), j]\sigma_{\mathsf{DB}}$: indeed, $[s^3(i), j]\sigma^{\mathsf{DB}}$.

Proposition 2.14

- $[i, j]\sigma^{\mathsf{D}}$ *iff basic-unification*(i, j)*;*
- $[i, j]\sigma^{\mathsf{B}}$ *iff b-axiom-unification*(i, j)*;*
- $[i, j]\sigma_{\mathsf{DB}}$ *iff db-unification*(i, j)*.*

Proof. Immediate, modulo renaming of variables, from the definitions of the unifications and the corresponding algorithms.

2.2.2 Inference Rules

For the presentation of the inference rules of KEM, and subsequently of SST we shall assume familiarity with Smullyan-Fitting α, β, ν, π unifying notation [7]. For the propositional part we exemplify only the rules for conjunction.

$$\frac{A \wedge B : i}{\begin{array}{c} A : i \\ B : i \end{array}} \qquad (\alpha\text{-rules})$$

The α-rules are just the familiar linear branch-expansion rules of the tableau method.

$$\frac{\begin{array}{c}\neg(A \wedge B) : i\\ A : j\end{array}}{\neg B : [i,j]\sigma_{\mathsf{DB}}} \qquad \frac{\begin{array}{c}\neg(A \wedge B) : i\\ B : j\end{array}}{\neg A : [i,j]\sigma_{\mathsf{DB}}} \tag{β-rules}$$

The β-rules are nothing but natural inference patterns such as Modus Ponens, Modus Tollens and Disjunctive syllogism generalised to the modal case. In order to apply such rules it is required that the labels of the premises unify and the label of the conclusion is the result of their unification.

$$\frac{\Diamond A : i}{A : (w_n, i)} \qquad \frac{\neg\Box A : i}{\neg A : (w_n, i)} \tag{π-rules}$$

where w_n is new, that is, it does not occur in the tree.

$$\frac{\Box A : i}{A : (W_n, i)} \qquad \frac{\neg\Diamond A : i}{\neg A : (W_n, i)} \tag{ν-rules}$$

where W_n is new.

ν- and π- rules allow us to expand labels according to the intended semantics, where, with “new” we mean that the label does not occur previously in the tree.

$$\frac{}{A : i \quad | \quad \neg A : i} \tag{PB}$$

PB (the “Principle of Bivalence”) represents the semantic counterpart of the cut rule of the sequent calculus (intuitive meaning: a formula A is either true or false in any given world). PB is a zero-premise inference rule, so in its unrestricted version can be applied whenever we like. However, we impose a restriction on its application. Then PB can be only applied w.r.t. immediate sub-formulas of unanalysed β-formulas, that is β formulas for which we have no immediate sub-formulas with the appropriate labels in the branch (tree).

$$\frac{\begin{array}{c}A : i\\ \neg A : j\end{array}}{\times} \tag{PNC}$$

if $[i,j]\sigma_{\mathsf{DB}}$.

The rule PNC (*Principle of Non-Contradiction*) states that two labelled formulas are σ_{DB}-complementary when the two formulas are complementary and their labels σ_{DB}-unify.

With $\vdash_{\mathrm{KEM}(\mathsf{DB})} A$ we mean that there is a close KEM-tree for $\neg A : w_1$; or, in other words, that KEM proves that A is a theorem of DB.

Theorem 2.15 $\vdash_{\mathrm{KEM}(\mathsf{DB})} A$ *iff* $\models_{\mathsf{DB}} A$.

Proof. For the proof and for detailed accounts of KEM see [1,10,9].

*2.3 Single Step Tableaux (*SST*)*

Single Step Tableaux [15] originate from and add modularity to Fitting's prefix tableaux [7]. The free-variable version we shall focus on here has been proposed by Beckert and Goré [3].

The basic idea of SST is that (modal) formulas are used to move the evaluation point to the "neighbourhood" of the labels they are associated with, that is, each time we are allowed to move only to one step apart. In other words the information that can be extracted from a formula is propagated only to the labels that the current label extends immediately or an are an immediate extension of the current label.

SST has the following inference rules. For the propositional part we give only the rules for $\wedge$.

$$\frac{A \wedge B : i}{\begin{array}{c} A : i \\ B : i \end{array}} \qquad (\alpha\text{-rules})$$

$$\frac{\neg(A \wedge B) : i}{\neg A : i \quad | \quad \neg B : i} \qquad (\beta\text{-rules})$$

$$\frac{\Diamond A : i}{A : (w_{\lceil \pi \rceil}, i)} \qquad \frac{\neg \Box A : i}{\neg A : (w_{\lceil \pi \rceil}, i)} \qquad (\pi\text{-rules})$$

where $\lceil \cdot \rceil$ is an arbitrary but fixed bijection from the set of formulas to N

$$\frac{\Box A : i}{A : (W_n, i)} \qquad \frac{\neg \Diamond A : i}{\neg A : (W_n, i)} \qquad (\nu_D\text{-rules})$$

$$\frac{\Box A : i}{A : b(i)} \qquad \frac{\neg \Diamond A : i}{\neg A : b(i)} \qquad (\nu_B\text{-rules})$$

The α-, π-, and ν_D-rules are common to KEM and SST and the β-rules are the usual branching rules of tableau methods. The ν_B-rules are the specific rules for symmetric logics. The intuition behind the latter is that symmetry allows us to travel backward in the accessibility relation. The main consequence of two sets of ν-rule is that every time we have a formula of type ν we have to introduce two new labelled formulas.

We say that two labelled formulas $A : i$ and $B : j$ are complementary in SST when $B = \neg A$ and there exists a substitution ρ which is a unifier of i and j.

With $\vdash_{\mathrm{SST(DB)}} A$ we mean that there is a close SST-tree for $\neg A : w_1$; or, in other words, that SST proves that A is a theorem of DB.

Theorem 2.16 $\vdash_{\mathrm{SST(DB)}} A$ *iff* $\models_{\mathsf{DB}} A$.

Proof. For the proof and for detailed accounts of SST see [3,17].

3 The Complexity of KEM Unifications

To provide a comparison of the two methods at hand first we have to study the complexity of the KEM unification procedure. We start by showing that the unification of two world symbols can be computed in constant time.

Lemma 3.1 *The σ-unification of two world symbols w and w' can be computed in constant time.*

Proof. It is immediate to see that the unification of two world symbols requires at most three steps, and thus it has constant complexity.

As we have seen the unification of two world symbols is just the first basic step of the unification. The next step is the σ-unification of two labels; in this case, we can prove that its complexity is linear in the length of the two labels.

Lemma 3.2 *The σ-unification of two labels i and j can be computed in linear time.*

Proof. All we have to do is to see whether the word symbols in the two labels stepwise unify.

Thus at the end we have to verify n unifications of world symbols, but from Lemma 3.1, we know that the unification of world symbols can be computed in constant time. Therefore the σ-unification of two labels can be computed in linear time.

The next unification we have to examine is the unification for the axiom B.

Lemma 3.3 *The σ^{B}-unification of two labels i and j can be computed in linear time.*

Proof. Here we have to count the number of consecutive variables in the head of the longest of the two labels; if such a number is appropriate (see Definition 2.9 and Algorithm 3), then we have to σ-unify the shortest label and a given segment of the longest; by Lemma 3.2, the σ-unification of two labels has linear complexity. Therefore the complexity of σ^{B} is linear.

Unfortunately we cannot prove such good complexity results for σ_{DB}; however, for special labels we can prove the following result.

Lemma 3.4 *The σ_{DB}-unification of two labels i and j such that $\ell(i) = 1$ can be computed in quadratic time.*

Proof. For a label j of length n there are n distinct segments and n distinct countersegments, namely

$$\begin{array}{rlcrl}
c^n(j) &= w_0 & \qquad & s^n(j) &= j \\
c^{n-1}(j) &= (h^n(j), w_0) & & s^{n-1}(j) &= b(j) \\
c^{n-2}(j) &= (h^n(j), (h^{n-1}(j), w_0)) & & s^{n-2}(j) &= b(b(j)) \\
&\vdots & & &\vdots
\end{array}$$

Now we have to see whether i either σ^{B}- or σ-unifies with the countersegments and whether i σ_{DB}-unifies with the segments. Thus we have to compute $2n$ linear unifications and n σ_{DB}-unifications. Let us examine the first of these, i.e., $[s^{n-1}(j), i]\sigma_{\mathsf{DB}}$. This time the length of $s^{n-1}(j)$ is $n-1$, and thus we have $n-1$ ways to split it in segments and countersegments. That is:[3]

$$\begin{array}{rlcrl}
c^{n-1}(c^{n-1}(j)) &= w_0 & \qquad & s^{n-1}(j) &= b(j) \\
c^{n-2}(c^{n-1}(j)) &= (h^{n-1}(j), w_0) & & s^{n-2}(j) &= b(b(j)) \\
&\vdots & & &\vdots
\end{array}$$

A close inspection shows that only the countersegments are different from the previous step. Therefore we can repeat this process for all the segments of j, and each time we can replace the σ_{DB} unification for the appropriate segment of length m, with $2m$ linear unifications. Hence, at the end, the number of linear unifications we have to compute is

$$2\sum_{n=1}^{n=\ell(j)} n = O(n^2)$$

which shows that the σ_{DB}-unification for the case at hand is quadratic.

4 KEM vs SST

So far the standard way to compare the relative complexity of two proof systems was given by the notion of p-simulation.

Definition 4.1 A proof system $\mathcal{A}$ p-simulates a proof system $\mathcal{B}$ iff there is a function g, computable in polynomial time, which maps derivations in $\mathcal{B}$ for any given formula ϕ, to derivations in $\mathcal{A}$ for ϕ (cf. [4]).

The main problem with p-simulation is that it considers only proofs, i.e., closed trees in tableaux terminology, and it says nothing about open trees. While this notion is fully appropriate for semi-decidable logics and non deterministic proof systems, it does not offer a good measure to compare tableaux-like proof-systems for decidable modal propositional logics. The main point

[3] Notice that for $m \leq n$ $s^m(s^n(i)) = s^m(i)$.

is that this notion does not contemplate proof-procedures. Modal tableaux proof-procedures, in effect, are systematic searches for models that make the initial formula true with respect to the initial world. In this perspective modal tableaux can show that a formula is not a theorem by showing that the negation of the formula is satisfiable. However, to show that a formula is satisfiable we have to complete its tree. In general, to complete a tree we have to explore the whole search space generated by the formula.

Therefore, to obviate the above problem, we propose a stepwise simulation. Here the main idea is that a proof system $\mathcal{A}$ stepwise simulates a proof system $\mathcal{B}$ iff $\mathcal{A}$ does not perform any inference steps for which no corresponding inference steps exist in $\mathcal{B}$.

Definition 4.2 A proof system $\mathcal{A}$ p-search-simulates a proof system $\mathcal{B}$ iff there is a polynomial function g such that for any formula ϕ, g maps derivations (trees) from ϕ in $\mathcal{A}$ to derivations (trees) from ϕ in $\mathcal{B}$ (cf. [5]).

Note that a stepwise simulation is independent of whether the considered derivations are proofs or not.

We are now ready to present the main result of the paper. To prove it we have to identify a formula (or a class of formulas) whose complete KEM-tree is polynomial while the complete SST-tree is exponential. Surprisingly the formula is extremely simple, namely:

$$p \to (\Box\Diamond)^n p \tag{3}$$

As we shall see (3) involves only one propositional linear step and there are no interaction between propositional connectives and modal operators. Therefore the discriminant is only the way the two proof systems deal with modalities.

Theorem 4.3 *The length of the complete proof of $p \to (\Box\Diamond)^n p$ in* KEM *is $O(n^2)$.*

Proof.

$$
\begin{array}{rl}
1. & \neg(p \to (\Box\Diamond)^n p) : w_1 \\
2. & p : w_1 \\
3. & \neg(\Box\Diamond)^n p : w_1 \\
4. & \neg\Diamond(\Box\Diamond)^{n-1} p : (w_2, w_1) \\
5. & \neg(\Box\Diamond)^{n-1} p : (W_1, (w_2, w_1)) \\
 & \vdots \\
2n+3. & \neg p : (W_n, (w_{n+1}, (\ldots, (W_1, (w_2, w_1)) \ldots)))
\end{array}
$$

The initial formula, i.e., $\neg(p \to (\Box\Diamond)^n p) : w_1$, is of type α, then we expand the tree with two nodes both labelled with w_1: the first of such nodes (2) consists of p which is atomic and does not need further investigations; the second node

(3) contains a formula of type π labelled with w_1. From (3) we obtain (4), which is of type ν. Applying the ν-rule on it, we get (5). We repeat the above steps $n-1$ times, for a total of $2n+3$ steps (nodes).

At this point we have two complementary formulas, the formulas in (2) and $(2n+3)$. We have to verify whether the two labels σ_{DB}-unify.

From Lemma 3.4 we know that the complexity of the σ_{DB}-unification at hand is quadratic. Therefore the complexity of the complete KEM-proof of $p \to (\Box\Diamond)^n p$ is $2n+3+O(n^2) = O(n^2)$.

Theorem 4.4 *The length of the complete proof of* $p \to (\Box\Diamond)^n p$ *in* SST *is* $O(2^{n+1})$.

Proof.

1. $\neg(p \to (\Box\Diamond)^n p) : w_1$
2. $p : w_1$
3. $\neg(\Box\Diamond)^n p : w_1$
4. $\neg\Diamond(\Box\Diamond)^{n-1} p : (w_2, w_1)$
5. $\neg(\Box\Diamond)^{n-1} p : (W_1, (w_2, w_1))$
6. $\neg(\Box\Diamond)^{n-1} p : w_1$
7. $\neg\Diamond(\Box\Diamond)^{n-2} p : (w_3, (W_1, (w_2, w_1)))$
8. $\neg\Diamond(\Box\Diamond)^{n-2} p : (w_3, w_1)$
9. $\neg(\Box\Diamond)^{n-2} p : (W_2, (w_3, (W_1, (w_2, w_1))))$
10. $\neg(\Box\Diamond)^{n-2} p : (w_2, w_1)$
11. $\neg(\Box\Diamond)^{n-2} p : (W_2, (w_3, w_1))$
12. $\neg(\Box\Diamond)^{n-2} p : w_1$

$\vdots$

The formula we start with ($\neg(p \to (\Box\Diamond)^n p) : w_1$) is of type α, and then we obtain two formulas $p : w_1$ and $\neg(\Box\Diamond)^n p : w_1$. At this point we have an atomic formula and a formula of type π. We apply the π-rule on it deriving $\neg\Diamond(\Box\Diamond)^{n-1} p : (w_2, w_1)$. Now we have a formula of type ν, and we have to apply both the ν-rule for D and B, thus we have to produce the formulas $\neg(\Box\Diamond)^{n-1} p : w_1$ and $\neg(\Box\Diamond)^{n-1} p : (W_1, (w_2, w_1))$. These last two formulas are of type π, and from them we obtain $\neg\Diamond(\Box\Diamond)^{n-2} p : (w_3, w_1)$ and $\neg\Diamond(\Box\Diamond)^{n-2} p : (w_3, (W_1, (w_2, w_1)))$; both formulas produce two new formulas. It is then clear that each formula of type ν produces two new formulas of less complexity, showing thus a geometrical progression; it is then immediate to see that the formula determining the number of steps is

$$\begin{aligned} 2\sum_{m=1}^{n} 2^{m-1} + 2^m &= 2\left(\frac{2^{(n-1)+1}-1}{2-1}\right) + 2^n \\ &= 2(2^n - 1) + 2^n \\ &= 2^{n+1} + 2^n - 2 \end{aligned}$$

thus the complexity of the complete proof of $p \to (\Box\Diamond)^n p$ in SST is $O(2^{n+1})$.

It is true that there are shorter proofs for (3) in SST. However, if we consider the formula

$$p \to (\Box\Diamond)^n q \tag{4}$$

which is not a theorem of DB, then the search space for it is $O(2^{n+1})$, since (4) has the same modal structure as (3). This is the reason why when we compare proof systems using p-search-simulation we have to consider exhaustive proof-search procedures and worst-case scenarios.

Theorem 4.5 SST *cannot p-search-simulate* KEM.

Proof. From Theorem 4.3 and Theorem 4.4 it follows that SST cannot p-simulate KEM since the complexity of $p \to (\Box\Diamond)^n p$ is $O(2^{n+1})$ for SST, while for KEM it is $O(n^2)$.

Let us now examine the question whether KEM p-search-simulates SST or whether the two systems cannot p-search-simulate each other. To show that a system $\mathcal{A}$ p-search-simulates a system $\mathcal{B}$ we have to define a polynomial procedure that transforms a tree for ϕ in $\mathcal{A}$ in a tree for ϕ in $\mathcal{B}$.

Lemma 4.6 *The rule ν_B is a derived rule in* KEM*, and it can be derived in polynomial time.*

Proof.

$$\begin{tikzcd}
& \nu : i \arrow[dl, dash] \arrow[dr, dash] & \\
\nu_0 : b(i) & & \overline{\nu}_0 : b(i) \arrow[d, dash] \\
& & \nu_0 : (W_n, i) \arrow[d, dash] \\
& & \times
\end{tikzcd}$$

We apply PB with respect to ν_0, and with label $b(i)$; in the right branch we apply the ν rule and we obtain $\nu_0 : (w_n, i)$, but $[b(i), (W_n, i)]\sigma_{\mathsf{DB}}$, and thus the branch is closed. In particular it is possible to show that the labels involved σ^{B}-unify, and we have seen (Lemma 3.3) that the σ^{B}-unification can be computed in linear time. Therefore the derivation of ν_B has linear complexity.

Lemma 4.6 allows us to define a proof-search in KEM where we use both the new derived ν-rule and the original ν-rules of KEM, and the unification is restricted to σ. It is immediate to see that this proof procedure corresponds to SST, and the components involved have linear complexity, we have thus proved the following theorem.

Theorem 4.7 KEM *p-search-simulates* SST.

5 Conclusions

In this paper we have compared two modal tableaux systems from a theoretical perspective, and we have investigated their relative complexity. We have shown that label unification algorithms could lead to theoretical speed-ups, therefore we believe that the study of relative complexity of modal proof systems is beneficial to the design of better modal theorem provers and better implementations.

References

[1] Artosi, A., P. Benassi, G. Governatori and A. Rotolo, *Shakespearian modal logic: A labelled treatment of modal identity*, in: M. Kracht, M. de Rijke, H. Wansing and M. Zakharyaschev, editors, *Advances in Modal Logic. Volume 1*, CSLI Publications, Stanford, 1998 pp. 1–21.

[2] Balsiger, P. and A. Heuerding, *Comparison of theorem provers for modal logics – introduction and summary*, in: H. C. M. de Swart, editor, *TABLEAUX'98*, number 1397 in LNCS (1998), pp. 25–26.

[3] Beckert, B. and R. Goré, *Free variable tableaux for propositional modal logics*, Studia Logica **69** (2001), pp. 59–96.

[4] Cook, S. A. and R. A. Reckhow, *The relative efficiency of propositional proof systems*, Journal of Symbolic Logic **44** (1979), pp. 36–50.

[5] de Nivelle, H., R. Schmidt and U. Hustadt, *Resolution-based methods for modal logics*, Logic Journal of IGPL **8** (2000), pp. 265–292.

[6] Fitch, F. B., *Tree proofs in modal logic*, Journal of Symbolic Logic **31** (1966), p. 152.

[7] Fitting, M., "Proof Methods for Modal and Intuitionistic Logics," Reidel, Dordrecht, 1983.

[8] Gabbay, D. M., "Labelled Deductive System," Oxford University Press, 1996.

[9] Gabbay, D. M. and G. Governatori, *Fibred modal tableaux*, in: D. Basin, M. D'Agostino, D. Gabbay, S. Matthews and L. Viganó, editors, *Labelled Deduction*, Applied Logic Series **17**, Kluwer, Dordrecht, 2000 pp. 163–194.

[10] Governatori, G., "Un modello formale per il ragionamento giuridico," Ph.D. thesis, CIRFID, University of Bologna, Bologna (1997).
URL `www.itee.uq.edu.au/~guido/Papers/_KEM/tesi.pdf`

[11] Governatori, G. and A. Luppi, *Labelled tableaux for non-normal modal logics*, in: E. Lamma and P. Mello, editors, *AI*IA 99: Advances in Artificial Intelligence*, LNAI **1792** (2000), pp. 119–130.
URL `www.itee.uq.edu.au/~guido/Papers/_KEM/aixia.pdf`

[12] Horrocks, I., P. F. Patel-Schneider and R. Sebastiani, *An analysis of empirical testing for modal decision procedures*, Logic Journal of IGPL **8** (2000), pp. 293–323.

[13] Hustadt, U. and R. Schmidt, *On evaluating decision procedures for modal logic*, in: *Proc. of the 15th Iternational Joint Conference on Artificial Intelligence (IJCAI'97)*, 1997, pp. 202–207.

[14] Marek, V., G. Shwarz and M. Truszczynski, *Modal non-monotonic logics: Ranges, characterization, computation*, Journal of the ACM **40** (1993), pp. 963–990.

[15] Massacci, F., *Strongly analytic tableaux for normal modal logic*, in: A. Bundy, editor, *CADE-12*, number 814 in LNAI (1994), pp. 723–737.

[16] Massacci, F., *Design and results of the tableaux-99 non-classical (modal) systems comparison*, in: N. V. Murray, editor, *TABLEAUX'99*, number 1617 in LNCS (1999), pp. 14–18.

[17] Massacci, F., *Single step tableaux for modal logic*, Journal of Automated Reasoning **24** (2000), pp. 319–364.

[18] Massacci, F. and F. M. Donini, *Design and results of tancs-2000 non-classical (modal) systems comparison*, in: R. Dyckhoff, editor, *TABLEAUX 2000*, number 1847 in LNCS (2000), pp. 52–56.

[19] Ohlbach, H. J., *Semantic based translation methods for modal logics*, Journal of Logic and Computation **1** (1991), pp. 691–746.

[20] Orłowska, E., *Logic for reasoning about knowledge*, Zeitschrift für mathematiske Logik und Grundlagen der Mathematik **35** (1989), pp. 559–572.

[21] Orłowska, E. and Z. Pawlak, *Representastion of nondeterministic information*, Theoretical Computer Science **29** (1984), pp. 27–39.

[22] Smullyan, R. M., "First-Order Logic," Springer-Verlag, Berlin, 1968.

[23] Wallen, L., "Automated Deduction in Nonclassical Logics," MIT Press, Cambridge Mass., 1990.