A benchmark method for the propositional modal logics K, KT, S4

Peter Balsiger, Alain Heuerding*, Stefan Schwendimann*
Institut für Informatik und angewandte Mathematik,
Universität Bern, Switzerland

December 21, 1999

Abstract

A lot of methods have been proposed — and sometimes implemented — for proof search in the propositional modal logics K, KT, and S4.

It is difficult to compare the usefulness of these methods in practice, since in most cases no or only a few execution times have been published. We try to improve this unsatisfactory situation by presenting a set of benchmark formulas. Note that we do not just list formulas, but give a method that allows to compare different provers today and in the future.

As a starting point we give the results we obtained when we applied this benchmark method to the Logics Workbench (LWB).

We hope that the discussion of postulates concerning ATP benchmark helps to obtain improved benchmark methods for other logics, too.

1 Introduction

A lot of methods have been proposed — and sometimes implemented — for proof search in the propositional modal logics K, KT, and S4. An incomplete list: tableaux and sequent calculi, embeddings in other logics, connection method, inverse method, logical frameworks. (There are far too many publications on this subject to list them here.)

It is difficult to compare the usefulness of these methods in practice, since in most publications no or only a few execution times are listed. There are some exceptions like [2] (31 formulas for K, KT, S4) and [4] (38 formulas for S4). However, these formulas are already today too simple to serve as benchmarks. For example it takes about 0.002 seconds to solve the hardest problem of [2] for K, KT, S4 with the Logics Workbench (LWB). The formulas of [4] are harder, but also there it takes only about 0.035 seconds to solve the hardest formula with the LWB.

This situation is very unsatisfactory. Therefore we decided to collect a set of benchmark problems for the propositional modal logics K, KT, and S4.

In classical predicate logic, the often-cited collection of Pelletier [10] has been replaced by the TPTP library [12]. Although this library is considerably large, it is still common to choose a dozen of these formulas out of this library and publish the execution times for these formulas. Therefore we do not just give a list of formulas, but present a method that makes it possible to compare different provers. (Also the producers of TPTP plan a so-called benchmark suite for the next version that should allow the computation of a performance index for automated theorem provers for classical predicate logic; cp. [12].)

*Work supported by the Swiss National Science Foundation, 21-43197.95.

The selection of the hard benchmark formulas was guided by the following postulates:

1. Provable as well as not provable formulas.
2. Formulas of various structures.
3. Some of the benchmark formulas are hard enough for forth-coming provers.
4. For each formula the result is already known today.
5. Simple 'tricks' do not help to solve the problems.
6. Applying the benchmark test to a prover takes not too much time.
7. The results can be summarized.

These postulates lead us to the exclusive use of scalable formulas. We test for which parameters n a certain prover can decide whether the scalable formula $A(n)$ is provable or not in less than 100 seconds. On purpose, this test does not take the machine speed into account. Because it is nearly impossible to determine a scaling factor to relate different machines, especially for distributed system, it is equally impossible to include it into the benchmarks. Furthermore, most of the benchmark formulas are built in such a way, that computation time increases exponentiallly with the scaling factor. Thus the speed of the machine will change the result for normally no more than one up or down. The results of a comparsion of modal provers held at the TABLEAUX '98 [1], showed that the type of machine used did not influence the overall results of a prover at all.
Clearly, these postulates and the benchmark formulas described below cannot measure any mathematical or logical properties of a prover. They are just used to measure the practical speed of the prover, i.e. the efficiency a user can expect when using the prover for general problems. Although this does ignore many aspects of provers, like user friendliness or special fitness for certain problems, it can be used to compare the general efficiency of provers.
The drawback of scalable formulas is their regular structure. If 'by chance' a prover 'recognizes' this structure, it can perform extremely well for a certain class of formulas, but still work very bad for slightly different formulas. We try to overcome this problem by hiding their structure with superfluous subformulas, and by presenting a sufficient number of different classes.
Collecting and constructing scalable formulas – always keeping the seven postulates in mind – is a very time-consuming task. This is not a problem in the approach chosen in [5], where formulas (in some sort of nested conjunctive normal form) are constructed using a random generator. However all these formulas have a similar structure (cp. postulate 2), and the correct result is not known beforehand (cp. postulate 4).
At the end we apply the benchmark method on the decision procedures integrated in the LWB. At the TABLEAUX '98 conference, the benchmarks were successfully used in a comparison of several modal provers [3].
Of course it is possible to implement much faster decision procedures for K, KT, and S4, but hopefully these results serve as a starting point.
For convenience we offer the possibility to fetch some formulas and the programs that produce these formulas via WWW. However note that this paper contains all the information that is required in order to use the benchmark method. This is very important: in a journal the formulas are accessible for many years, whereas experience shows that it is difficult to ensure the maintenance of files in ftp or WWW site over a long period. We think that a major part of the success of the collection of Pelletier was the publication of the formulas in this journal.

2 Postulates

These postulates guided us when we developed our benchmark method. Note that they are widely applicable, not just for the logics we consider in this paper.

Provable as well as not provable formulas.

Often only provable formulas are considered as benchmark formulas. We think that not provable formulas are as interesting as provable ones.
Moreover, algorithms like the Greedy procedure in classical propositional logic ([11]) can recognize many not provable formulas in a very short time, but they cannot show that a formula is provable. A benchmark method must also be applicable for such semi-decision procedures.

Formulas of various structures
It is clear that four or five formulas are not enough to obtain representative results about the performance of a prover. However, also a large number of formulas does not guarantee the quality of the benchmark set. For example it is tempting to use the embedding of IPC in S4 in order to obtain a list of benchmark formulas for S4 from a list of benchmark formulas for IPC, but all the resulting formulas look similar. An even more dubious method it the use of just one class of formulas, e.g. the pigeonhole formulas.
Things look of course different if one has a certain application in mind and develops a tuned prover for this application, but here we want to measure the overall performance.

Some of the benchmark formulas are hard enough for forth-coming provers.
It is not enough if the benchmark test is hard enough for today's provers. The test should still be applicable for much faster computers and improved search methods in the future. Thus the test must contain formulas that are far too hard for existing provers.

For each formula the result is already known today.
The correct result, i.e. 'provable' or 'not provable', must be known already today. Therefore it must be possible to check the provability resp. non-provability of the formulas with logical methods, and not just with theorem provers. Random formulas do not fulfill this postulate.

Simple 'tricks' do not help.
The addition of a simple trick to the prover should not influence the results.
Assume for example that a benchmark set for K contains formulas without $\Box$ and without $\Diamond$. If a prover checks at the beginning whether modal operators occur in the formula, then he can apply a fast decision procedure for classical propositional logic. Therefore we disguised e.g. the pigeonhole formula with some 'superfluous' $\Box$ and $\Diamond$. (If a prover recognizes this disguise, or if he sees during the proof search that a subproblem is purely classical, then we think that this is no longer a simple trick.)
Of course it is impossible to foresee all such tricks when constructing benchmark formulas, but at least one should try.

Execution of the benchmark test takes not too much time.
Nobody wants to spend a lot of time to measure the performance of his prover. In particular, the required time should not depend on the prover or on the computer. Therefore some limits must be used, e.g. 'if the prover cannot solve the problem in n seconds, then stop'.

Results can be summarized.
A list of hundreds of numbers is not a satisfactory result, since it makes a comparison of several provers almost impossible. Therefore it must be possible to summarize the results. On the other hand one single number is probably not satisfactory (remember the semi-decision procedures).

3 Benchmark method for K, KT, and S4

3.1 Formulas

For each logic L, the formulas are divided into 9 classes of provable and 9 classes of not provable formulas. The formulas in each class are parametrized by a number in $\mathbb{N}$. We call the jth formula in the ith class of provable resp. not provable formulas $F_{\mathrm{p},i,j}$ resp. $F_{\mathrm{n},i,j}$. Thus $L \vdash F_{\mathrm{p},i,j}$ and $L \nvdash F_{\mathrm{n},i,j}$ for all $i \in \{1,\ldots,9\}$, $j \in \mathbb{N}$.

Let $t(C)$ be the time in seconds the prover takes to decide whether or not the formula C is provable, i.e. $t : \mathrm{Fml} \to \mathbb{N} \cup \infty$. We compute for each $i \in \{1, \ldots, 9\}$ the numbers $n_{\mathrm{p},i} := \max\{j \mid t(F_{\mathrm{p},i,j}) < 100\ \mathit{seconds}\}$ and $n_{\mathrm{n},i} := \max\{j \mid t(F_{\mathrm{n},i,j}) < 100\ \mathit{seconds}\}$. Thus $F_{\mathrm{p},i,n_{\mathrm{p},i}+1}$ is the first formula in the ith class of provable formulas such that the prover cannot decide its provability in less than 100 seconds ($F_{\mathrm{n},i,n_{\mathrm{n},i}+1}$ analogous).
The limit of 100 seconds may be too low to be able to separate two provers with only a minimal difference in the execution times. Furthermore, a prover producing bad results with small problems might still be faster for complex problems, because of a slower increase in execution times. By carefully selecting the appropriate time limit according to the provers to be tested, it should be possible to avoid the problem. Furthermore, the general user of a prover normally is interested in problems the prover can solver in quite a short time, anyway.

3.2 Timing

It is not possible to describe a timing procedure that is sensible for all provers (think e.g. of non-deterministic and parallel provers). Therefore it is important that everybody who applies the benchmark method describes exactly how the timing took place. See section 6 for an example.
If possible, then observe the following conditions in order to make the comparison of different provers easier.

- The timing starts after the start-up of the prover.
- No conversions, e.g. in negation normal form, before the timing starts.
- The timing includes the construction of the data structure that contains the formula. This is especially important if you use some sort of preprocessing.
- Make your prover accessible in some way, in order to give people the possibility to check your results.

3.3 Presentation of the results

It is important that others can check your results. We propose to include the following information (see section 6 for an example):

1. Name of the prover, a reference to further information, a short description of the methods used by the prover.
2. Availability of the prover.
3. Short list of features your prover offers besides checking the provability of formulas: Examples: Is the prover tuned for a certain application? Has the correctness of the prover been verified? Can one use one prover for many logics? Is a proof resp. a counter-model generated?
4. Hardware that was used for the benchmark test.
5. An example of the way you timed your prover.
6. Results, i.e. the numbers $n_{\mathrm{p},i}$ and $n_{\mathrm{n},i}$ for all $i \in \{1, \ldots, 9\}$.

3.4 What about the postulates ?

Our method largely satisfies the seven postulates:

1. There are provable as well as not provable formulas.
2. The formulas have various properties (number of variables, modal depth, ...) and origins.
3. Each class contains arbitrarily hard formulas.
4. It is clear which formulas are provable.
5. We tried ...

6. Applying our benchmark method is rather tiresome if one does not automate it, but it can be done in a reasonable time.
7. For each logic the result consists of a list of 18 numbers.

The main problems are the postulate 5, and — at least for semi-decision procedures — the relatively small number of classes.

4 Notation

Logics: CPC stands for classical propositional logic. K, KT, S4 are the modal propositional logics we consider. In order to avoid misunderstanding we list the corresponding properties of the accessibility relation in Kripke semantics. K: neither reflexive nor transitive, KT: reflexive, but not transitive, S4: reflexive and transitive.

Variables and formulas: $\mathrm{Var} = \{p_0, p_1, p_2, \ldots\}$ is the set of the variables. We inductively define the set of the formulas Fml: $\mathsf{true}, \mathsf{false} \in \mathrm{Fml}$; $P \in \mathrm{Var} \Rightarrow P \in \mathrm{Fml}$; $A \in \mathrm{Fml} \Rightarrow \Box A, \Diamond A, \neg A \in \mathrm{Fml}$; $A, B \in \mathrm{Fml} \Rightarrow A \wedge B, A \vee B, A \to B, A \leftrightarrow B \in \mathrm{Fml}$. $A \equiv C$ means that the two formulas A and C are syntactically equal.

Conventions: We use meta-variables (sometimes with subscripts) as follows: A, C for formulas, n for elements of $\{1, 2, \ldots\}$.
$\vee$ and $\wedge$ are left-associative. $\neg$, $\Box$, $\Diamond$ have the highest priority, followed by $\wedge$, $\vee$, $\to$, and $\leftrightarrow$. We omit the outermost brackets. Example: $p_0 \vee p_1 \vee p_2 \to (\neg p_2 \wedge p_1) \leftrightarrow p_0$ stands for the formula $((((p_0 \vee p_1) \vee p_2) \to ((\neg p_2) \wedge p_1)) \leftrightarrow p_0)$.

Iterations: $\Box^0 A$ stands for the formula A, $\Box^n A$ stands for the formula $\Box\Box^{n-1}A$. $\Diamond^0 A$ stands for the formula A, $\Diamond^n A$ stands for the formula $\Diamond\Diamond^{n-1}A$. $\bigwedge_{i=n_1,\ldots,n_2}(A_i)$ stands for the formula $A_{n_1} \wedge A_{n_1+1} \wedge \ldots \wedge A_{n_2}$ (if $n_1 \leq n_2$) resp. for the formula true (if $n_1 > n_2$). $\bigvee_{i=n_1,\ldots,n_2}(A_i)$ stands for the formula $A_{n_1} \vee A_{n_1+1} \vee \ldots \vee A_{n_2}$ (if $n_1 \leq n_2$) resp. for the formula false (if $n_1 > n_2$).

Substitution: If $P_1, \ldots, P_n$ are variables, then $A\{C_1/P_1, \ldots, C_n/P_n\}$ is the formula A with simultaneously substituted P_1 by C_1, $\ldots$, P_n by C_n. Example: $(p_0 \vee \Box(p_1 \to p_0))\{p_2/p_0, \Diamond p_1 \leftrightarrow p_0/p_1\} \equiv p_2 \vee \Box((\Diamond p_1 \leftrightarrow p_0) \to p_2)$.

Lists: $[x_1, x_2, \ldots, x_n]$ is the list with the elements x_1, x_2, $\ldots$, x_n. Note that $[1,2]$, $[2,1]$, $[1,1,2]$ are three different lists. $\bigvee_{y \in [x_1,\ldots,x_n]}(A_x)$ stands for $A_{x_1} \vee \ldots \vee A_{x_n}$.

Standard functions: The function nnf (negation normal form) for formulas without equivalence symbols, div and mod are defined as usual. Examples: $\mathrm{nnf}(\neg p_0 \to \neg(\Box p_1 \vee \mathsf{true})) \equiv p_0 \vee \Diamond\neg p_1 \wedge \mathsf{false}$, $14 \mathbin{\mathrm{div}} 3 = 4$, $14 \bmod 3 = 2$.

Standard formulas:
D $:\equiv \Box p_0 \to \Diamond p_0$
$\mathrm{D}_2 :\equiv \Diamond\mathsf{true}$
B $:\equiv p_0 \to \Box\Diamond p_0$
T $:\equiv \Box p_0 \to p_0$
A4 $:\equiv \Box p_0 \to \Box\Box p_0$
A5 $:\equiv \neg\Box p_0 \to \Box\neg\Box p_0$
H $:\equiv \Box(p_0 \vee p_1) \wedge \Box(\Box p_0 \vee p_1) \wedge \Box(p_0 \vee \Box p_1) \to \Box p_0 \vee \Box p_1$
L $:\equiv \Box(p_0 \wedge \Box p_0 \to p_1) \vee \Box(p_1 \wedge \Box p_1 \to p_0)$
$\mathrm{L}^+ :\equiv \Box(\Box p_0 \to p_1) \vee \Box(\Box p_1 \to p_0)$
Grz $:\equiv \Box(\Box(p_0 \to \Box p_0) \to p_0) \to p_0$

$\mathrm{Grz}_1 :\equiv \Box(\Box(p_0 \to \Box p_0) \to p_0) \to \Box p_0$
$\mathrm{Dum} :\equiv \Box(\Box(p_0 \to \Box p_0) \to p_0) \to (\Diamond\Box p_0 \to p_0)$
$\mathrm{Dum}_1 :\equiv \Box(\Box(p_0 \to \Box p_0) \to p_0) \to (\Diamond\Box p_0 \to \Box p_0)$
$\mathrm{Dum}_4 :\equiv \Box(\Box(p_0 \to \Box p_0) \to p_0) \to (\Diamond\Box p_0 \to p_0 \vee \Box p_0)$

5 Formulas

5.1 Presentation of formulas

For each class of formulas we list in the following section:

1. Idea: Why is the formula provable resp. not provable?
2. Hiding: How is the formula 'hidden' in order to make the problem harder to solve?
3. An inductive definition of the formulas.

5.2 Formulas for K

k_branch_p

Idea: The branching formula as defined in [6], plus a negation symbol in front and the additional subformula $\neg\Box^n p_{n \text{ div } 3+1}$ in order to make the formula provable. We assume $n < 100$.

$k\_branch\_p(n) :\equiv \neg(p_{100} \wedge \neg p_{101} \wedge \bigwedge_{i=0,\ldots,n}(\Box^i(bdepth(n) \wedge det(n) \wedge branching(n)))) \vee \neg\Box^n p_{n \text{ div } 3+1}$
$bdepth(n) :\equiv \bigwedge_{i=1,\ldots,n+1}(p_{100+i} \to p_{99+i})$
$det(n) :\equiv \bigwedge_{i=0,\ldots,n}(p_{100+i} \to (p_i \to \Box(p_{100+i} \to p_i)) \wedge (\neg p_i \to \Box(p_{100+i} \to \neg p_i)))$
$branching(n) :\equiv$
$\bigwedge_{i=0,\ldots,n-1}(p_{100+i} \wedge \neg p_{101+i} \to \Diamond(p_{101+i} \wedge \neg p_{102+i} \wedge p_{i+1}) \wedge \Diamond(p_{101+i} \wedge \neg p_{102+i} \wedge \neg p_{i+1}))$

k_branch_n

Idea: The branching formula as defined in [6].

$k\_branch\_n(n) :\equiv \neg(p_{100} \wedge \neg p_{101} \wedge \bigwedge_{i=0,\ldots,n}(\Box^i(bdepth(n) \wedge det(n) \wedge branching(n))))$
bdepth, *det*, *branching* as in *k_branch_p*.

k_d4_p

Idea: $\mathsf{K} \vdash \mathrm{D} \wedge \mathrm{A4} \wedge \mathrm{B}\{\neg p_0/p_0\} \to \mathrm{T}$.
Hiding: The left hand side occurs with 1 to n $\Box$, the right hand side occurs just with n $\Box$ in front. $\Box^n\mathrm{T}$ is repeated n times. Additionally some superfluous instances, and the whole formula in negation normal form.

$k\_d4\_p(n) :\equiv \mathrm{nnf}(\bigvee_{i=1,\ldots,n}(\Box^n\mathrm{T} \vee \neg\Box^i\mathrm{D}_2 \vee \neg\Box^i\mathrm{A4} \vee \neg\Box^i\mathrm{A4}\{\Diamond p_0/p_0\} \vee \neg\Box^i\mathrm{B} \vee \neg\Box^i\mathrm{B}\{\neg p_0/p_0\}))$

k_d4_n

Idea: A5 is not provable in K plus any instances of D, A4, and T.
Hiding: As for *k_d4_p*.

$k\_d4\_n(n) :\equiv \mathrm{nnf}(\bigvee_{i=1,\ldots,n}(\Box^n(\Box p_0 \vee \Box\Diamond\neg p_0) \vee \neg\Box^i\mathrm{D}_2 \vee \neg\Box^i\mathrm{A4} \vee \neg\Box^i\mathrm{A4}\{\Diamond p_0/p_0\} \vee \neg\Box^i\mathrm{D}$
$\vee \neg\Box^i\mathrm{A4}\{\Diamond p_0 \to p_0/p_0\} \vee \neg\Box^i\mathrm{A4}\{\Box p_0 \to p_0/p_0\}))$

k_dum_p

Idea: $\mathsf{K} \vdash \mathrm{A4}\{\Box(p_0 \to \Box p_0) \to p_0/p_0\} \wedge \Box\mathrm{A4} \wedge \mathrm{Dum} \wedge \mathrm{Dum}\{p_0 \to \Box p_0/p_0\} \to \mathrm{Dum}_1$.
Hiding: Some of the formulas on the left hand side of the implication occur with 1 to $n-1$ $\Box$ in front, the right hand side occurs just once with n div $2+1$ $\Box$ in front.

$k\_dum\_p(n) :\equiv \bigwedge_{i=1,\ldots,n \text{ div } 2}(\Box^i(\Box\mathrm{A4} \wedge \mathrm{Dum})) \wedge \neg\Box^{n \text{ div } 2+1}\mathrm{Dum}_1$
$\to \Diamond^{n \text{ div } 2+1}\neg(\mathrm{A4}\{\Box(p_0 \to \Box p_0) \to p_0/p_0\} \wedge \Box\mathrm{A4} \wedge \mathrm{Dum} \wedge \mathrm{Dum}\{p_0 \to \Box p_0/p_0\})$
$\vee \bigvee_{i=n \text{ div } 2+2,\ldots,n-1}(\Diamond^i\neg(\Box\mathrm{A4} \wedge \mathrm{Dum}))$

k_dum_n

Idea: Dum is not provable in K plus any instances of Dum_4 and A4.

Hiding: As for *k_dum_p*.

$$\begin{aligned} k\_dum\_n(n) :\equiv & \textstyle\bigwedge_{i=1,\ldots,n \text{ div } 2}(\Box^i(\Box\mathrm{A4} \wedge \mathrm{Dum}_4)) \wedge \neg\Box^{n+1}\mathrm{Dum} \\ & \to \Diamond^{n+1}\neg(\mathrm{A4}\{\Box(p_0 \to \Box p_0) \to p_0/p_0\} \wedge \Box\mathrm{A4} \wedge \mathrm{Dum}_4 \wedge \mathrm{Dum}_4\{p_0 \to \Box p_0/p_0\}) \\ & \quad \vee \textstyle\bigvee_{i=n \text{ div } 2+2,\ldots,n-1}(\Diamond^i\neg(\Box\mathrm{A4} \wedge \mathrm{Dum}_4)) \end{aligned}$$

k_grz_p

Idea: $\mathsf{K} \vdash \Box\mathrm{Grz} \wedge \mathrm{Grz}\{C() \wedge \mathrm{A4}\{C()/p_0\}/p_0\} \to \mathrm{Grz}_1$, where $C()$ is defined as below.

Hiding: Many superfluous instances with three variables, and iterated $\Box$ inside the instances.

$$\begin{aligned} k\_grz\_p(n) :\equiv & \Box\mathrm{Grz}\{p_2/p_0\} \wedge \textstyle\bigwedge_{i=1,\ldots,n-1}(l(i)) \wedge \mathrm{Grz}\{C() \wedge \mathrm{A4}\{C()/p_0\}/p_0\} \\ & \to \mathrm{Grz}_1\{p_1/p_0\} \vee \mathrm{Grz}_1\{p_2/p_0\} \vee \mathrm{Grz}_1\{p_3/p_0\} \end{aligned}$$

$$l(i) :\equiv \begin{cases} \mathrm{Grz}\{l2(i \text{ div } 4)/p_0\} & i \bmod 4 = 0 \\ \mathrm{Grz}\{\Box l2(i \text{ div } 4) \vee p_1/p_0\} & i \bmod 4 = 1 \\ \mathrm{Grz}\{\Box l2(i \text{ div } 4) \vee p_1 \vee p_2/p_0\} & i \bmod 4 = 2 \\ \mathrm{Grz}\{\Box l2(i \text{ div } 4) \vee p_1 \vee p_2 \vee p_3/p_0\} & \text{otherwise} \end{cases}$$

$$l2(i) :\equiv \begin{cases} \mathsf{false} & i = 0 \\ \Box l2(i-1) \vee p_1 \vee p_2 \vee p_3 \vee p_4 & \text{otherwise} \end{cases}$$

$$C() :\equiv \Box(p_2 \to \Box p_2) \to p_2$$

k_grz_n

Idea: Grz is not provable in K plus instances of Grz_1.

Hiding: As for *k_grz_p*.

$$\begin{aligned} k\_grz\_n(n) :\equiv & \Box\mathrm{Grz}_1\{p_2/p_0\} \wedge \textstyle\bigwedge_{i=1,\ldots,n-1}(l(i)) \wedge \mathrm{Grz}_1\{C() \wedge \mathrm{A4}\{C()/p_0\}/p_0\} \\ & \to \mathrm{Grz}\{p_1/p_0\} \vee \mathrm{Grz}\{p_2/p_0\} \vee \mathrm{Grz}\{p_3/p_0\} \end{aligned}$$

$$l(i) :\equiv \begin{cases} \mathrm{Grz}_1\{l2(i \text{ div } 4)/p_0\} & i \bmod 4 = 0 \\ \mathrm{Grz}_1\{\Box l2(i \text{ div } 4) \vee p_1/p_0\} & i \bmod 4 = 1 \\ \mathrm{Grz}_1\{\Box l2(i \text{ div } 4) \vee p_1 \vee p_2/p_0\} & i \bmod 4 = 2 \\ \mathrm{Grz}_1\{\Box l2(i \text{ div } 4) \vee p_1 \vee p_2 \vee p_3/p_0\} & \text{otherwise} \end{cases}$$

l2, *C* as in *k_grz_p*.

k_lin_p

Idea: $\mathsf{K} \vdash H2 \to \mathrm{L}$, where *H2* is the instance of H defined below.

Hiding: Superfluous instances of *H2* with other variables.

$$k\_lin\_p(n) :\equiv \textstyle\bigvee_{i=1,\ldots,n \text{ div } 3}(\neg H2(p_i, p_{i+1})) \vee \mathrm{L}\{p_n/p_0, p_n/p_1\} \vee \bigvee_{i=n \text{ div } 3+1,\ldots,n}(\neg H2(p_i, p_{i+1}))$$

$$H2(A, B) :\equiv \mathrm{H}\{A \wedge \Box A \wedge A \to B/p_0, \neg A \to \neg(\Box B \wedge B)/p_1\}$$

k_lin_n

Idea: L^+ is not provable in K plus any instances of L.

$$\begin{aligned} k\_lin\_n(n) :\equiv & \textstyle\bigvee_{i=1,\ldots,2n-2}(\neg\mathrm{L}\{\Diamond p_i/p_0, p_{i+1}/p_1\} \vee \neg\mathrm{L}\{p_i \to \Box p_{i+1}/p_0, p_{i+1}/p_1\}) \\ & \vee \mathrm{L}^+\{p_n/p_0, p_n/p_1\} \\ & \vee \textstyle\bigvee_{i=2n,\ldots,4n-4}(\neg\mathrm{L}\{\Diamond p_i/p_0, p_{i+1}/p_1\} \vee \neg\mathrm{L}\{p_i \to \Box p_{i+1}/p_0, p_{i+1}/p_1\}) \end{aligned}$$

k_path_p

Idea: Without hiding, the formula would just be $\Box p_{\mathrm{path\_el}(1,n)} \vee \Diamond(\neg p_{\mathrm{path\_el}(1,n)} \wedge \Box p_{\mathrm{path\_el}(2,n)}) \vee \ldots \vee \Diamond(\neg p_{\mathrm{path\_el}(n-1,n)} \wedge \Box p_{\mathrm{path\_el}}(n,n)) \vee \Diamond^n p_{\mathrm{path\_el}(n,n)}$.

Hiding: The formula above is the path from the entry to the exit of a labyrinth described by *k_path_p*. It makes no difference whether one starts at the entry or at the exit.

$$\begin{aligned} k\_path\_p(n) :\equiv & (\Box p_1 \vee \Box p_{path\_el(1,n)} \vee \Box p_3 \vee \Box p_5) \\ & \vee \textstyle\bigvee_{i=1,\ldots,n;j=1,\ldots,6}(left\_to\_right(i,j,n) \vee right\_to\_left(i,j,n)) \\ & \vee (\Diamond^n\neg p_2 \vee \Diamond^n\neg p_4 \vee \Diamond^n\neg p_{path\_el(n,n)} \vee \Diamond^n\neg p_6) \end{aligned}$$

$$path\_el(l,n) :\equiv \begin{cases} 2 & l = 1 \\ modg(path\_el(l-1,n)+3,6) & l > n \text{ div } 2 \\ modg(path\_el(l-1,n)+5,6) & \text{otherwise} \end{cases}$$

$$left\_to\_right(l,k,n) :\equiv \begin{cases} \mathsf{false} & l = n \\ \mathsf{false} & l \neq n, k \bmod 2 = 0, k \neq path\_el(l,n) \\ lists2fml(l,k,[path\_el(l+1,n)]) & l \neq n, k \bmod 2 = 0, k = path\_el(l,n) \\ lists2fml(l,k,[1,3,path\_el(l+1,n),5]) & l \neq n, k \bmod 2 \neq 0, k = path\_el(l,n) \\ lists2fml(l,k,delete(path\_el(l+1,n),1,3,5)) & \text{otherwise} \end{cases}$$

$$right\_to\_left(l,k,n) :\equiv \begin{cases} \mathsf{false} & l = 1 \\ \mathsf{false} & l \neq 1, k \bmod 2 = 1 \\ lists2fml\_back(l-1,k,delete(path\_el(l-1,n),2,4,6)) & \text{otherwise} \end{cases}$$

$$lists2fml(l,k,s) :\equiv \bigvee_{x\in s}(\Diamond^l(\neg p_k \wedge \Box p_x))$$

$$lists2fml\_back(l,k,s) :\equiv \bigvee_{x\in s}(\Diamond^l(\neg p_x \wedge \Box p_k))$$

$$delete(x,y1,y2,y3) :\equiv \begin{cases} [y2,y3] & x = y1 \\ [y1,y3] & x = y2 \\ [y1,y2] & x = y3 \\ [y1,y2,y3] & \text{otherwise} \end{cases}$$

$$modg(n_1,n_2) :\equiv \begin{cases} n_2 & n_1 \bmod n_2 = 0 \\ n_1 \bmod n_2 & \text{otherwise} \end{cases}$$

k_path_n

Idea: As for *k_path_p*, but one piece of the path is missing (cp. the different definitions of *left_to_right* in *k_path_p* and *k_path_n*).

Hiding: As for *k_path_p*.

$$k\_path\_n(n) :\equiv (\Box p_1 \vee \Box p_{path\_el(1,n+1)} \vee \Box p_3 \vee \Box p_5) \vee \bigvee_{i=1,\ldots,n+1;j=1,\ldots,6}(left\_to\_right(i,j,n+1) \vee right\_to\_left(i,j,n+1)) \vee (\Diamond^{n+1}\neg p_2 \vee \Diamond^{n+1}\neg p_4 \vee \Diamond^{n+1}\neg p_{path\_el(n+1,n+1)} \vee \Diamond^{n+1}\neg p_6)$$

path_el, *right_to_left*, *lists2fml*, *lists2fml_back*, *delete*, *modg* as in *k_path_p*.

$$left\_to\_right(l,k,n) :\equiv \begin{cases} \mathsf{false} & l = n \\ \mathsf{false} & l \neq n, k \bmod 2 = 0, k \neq path\_el(l,n) \\ \mathsf{false} & l \neq n, k \bmod 2 = 0, k = path\_el(l,n), l = n \text{ div } 2 \\ lists2fml(l,k,[path\_el(l+1,n)]) & l \neq n, k \bmod 2 = 0, k = path\_el(l,n), l \neq n \text{ div } 2 \\ lists2fml(l,k,[1,3,5]) & l \neq n, k \bmod 2 \neq 0, k = path\_el(l,n), l = n \text{ div } 2 \\ lists2fml(l,k,[1,3,path\_el(l+1,n),5]) & l \neq n, k \bmod 2 \neq 0, k = path\_el(l,n), l \neq n \text{ div } 2 \\ lists2fml(l,k,delete(path\_el(l+1,n),1,3,5)) & \text{otherwise} \end{cases}$$

k_ph_p

Idea: The pigeonhole formulas. We assume $n < 100$.

Hiding: Some $\Box$ and $\Diamond$.

$$k\_ph\_p(n) :\equiv \Diamond left(n) \rightarrow \Diamond right(n)$$

$$left(n) :\equiv \bigwedge_{i=1,\ldots,n+1}(\bigvee_{j=1,\ldots,n}(l(i,j)))$$

$$right(n) :\equiv \bigvee_{j=1,\ldots,n;i_1=1,\ldots,n+1;i_2=i_1+1,\ldots,n+1}(l(i_1,j) \wedge l(i_2,j))$$

$$l(i,j) :\equiv \begin{cases} \Box p_{100i+j} & i < j \\ p_{100i+j} & \text{otherwise} \end{cases}$$

k_ph_n

Idea: The pigeonhole formulas, with one missing conjunct on the right hand side. We assume $n < 100$.

Hiding: Some $\Box$ and $\Diamond$.

$$k\_ph\_n(n) :\equiv \Diamond left(n) \rightarrow \Diamond right(n)$$

left, *l* as in *k_ph_p*.

$$right(n) :\equiv \bigvee_{j=1,\ldots,n;i_1=1,\ldots,n+1;i_2=i_1+1,\ldots,n+1}(l2(n,i_1,j) \wedge l2(n,i_2,j))$$

$$l2(n,i,j) :\equiv \begin{cases} \neg l(i,j) & i=j, i=(2n) \text{ div } 3+1 \\ l(i,j) & \text{otherwise} \end{cases}$$

k_poly_p

Idea: The formula $(p_1 \leftrightarrow p_2) \vee (p_2 \leftrightarrow p_3) \vee \ldots \vee (p_{n-1} \leftrightarrow p_n) \vee (p_n \leftrightarrow p_1)$ says: If we have a polygon with n vertices, and all the vertices are either black or white, then two adjacent vertices have the same colour. If n is odd, then this formula is provable in CPC.
Hiding: Many $\Box$, $\Diamond$, and superfluous subformulas.

$$k\_poly\_p(n) :\equiv \begin{cases} poly(3n+1) & n \bmod 2 = 0 \\ poly(3n) & n \bmod 2 = 1 \end{cases}$$

$$poly(n) :\equiv \Box^{n+1} \bigwedge\nolimits_{i=1,\ldots,n+1}(p_i) \vee f(n,n) \vee \Box^{n+1} \bigwedge\nolimits_{i=1,\ldots,n+1}(\neg p_{2i})$$

$$f(i,n) :\equiv \begin{cases} \mathsf{false} & i=0 \\ \Diamond(f(i-1,n) \vee \Diamond^i(p_n \leftrightarrow p_1)) \vee \Box p_{i+2} & i=n \\ \Diamond(f(i-1,n) \vee \Diamond^i(p_i \leftrightarrow p_{i+1})) \vee \Box p_{i+2} & \text{otherwise} \end{cases}$$

k_poly_n

Idea: As for *k_poly_p*, but for an even number of vertices.
Hiding: Many $\Box$, $\Diamond$, and superfluous subformulas. The superfluous subformulas do not influence the non-provability.

$$k\_poly\_n(n) :\equiv \begin{cases} poly(3n) & n \bmod 2 = 0 \\ poly(3n+1) & n \bmod 2 = 1 \end{cases}$$

poly, f as in *k_poly_p*.

k_t4p_p

Idea: $\mathsf{K} \vdash \mathrm{T}\{\Diamond p_0/p_0\} \wedge \Box\mathrm{T}\{\neg\Box\Diamond p_0/p_0\} \wedge \mathrm{A4}\{\Diamond p_0/p_0\} \wedge \Box(\Diamond\Box\Diamond p_0 \rightarrow (p_0 \rightarrow \Box p_0)) \rightarrow \Diamond\Box p_0 \vee \Diamond\Box\neg p_0$.
Hiding: Superfluous subformulas ($\Diamond\neg p_3$, $\Diamond p_4$), superfluous instances of A4 and A5, nested $\Box$.

$$k\_t4p\_p(n) :\equiv E(n) \vee \mathrm{nnf}(\neg C(n)) \vee \Diamond p_4$$

$$C(i) :\equiv \begin{cases} ((\Box\Diamond p_0 \rightarrow \Diamond p_1) \wedge \Box(\Box\neg\Box\Diamond p_1 \rightarrow \neg\Box\Diamond p_0) \wedge (\Box\Diamond p_0 \rightarrow \Box\Box\Diamond p_1) & \\ \quad \wedge\Box(\Diamond\Box\Diamond p_0 \wedge p_0 \rightarrow \Box p_1) \wedge \Box\Diamond p_1)\{p_0 \wedge \Diamond\neg p_3/p_1\} & i=0 \\ \Box\mathrm{A4}\{p_1/p_0\} \wedge \Box C(i-1) \wedge \Box\mathrm{A4}\{\Diamond p_1/p_0\} & \text{otherwise} \end{cases}$$

$$E(i) :\equiv \begin{cases} \Diamond\Box p_0 & i=0 \\ \Diamond\neg\mathrm{A4}\{\neg p_1/p_0\} \vee \Box E(i-1) \vee \Box\mathrm{A5}\{p_1/p_0\} & \text{otherwise} \end{cases}$$

k_t4p_n

Idea: $\Diamond\Box p_0$ is not provable in K plus any instances of T, A4, and $\Box(\Diamond\Box\Diamond p_0 \rightarrow (p_0 \rightarrow \Box p_0))$.
Hiding: As for *k_t4p_p*.

$$k\_t4p\_n(n) :\equiv E(2n-1) \vee \mathrm{nnf}(\neg C(2n-1)) \vee \Diamond p_4$$

$$C(i) :\equiv \begin{cases} ((\Box\Diamond p_0 \rightarrow \Diamond p_1) \wedge \Box(\Box\neg\Box\Diamond p_1 \rightarrow \neg\Box\Diamond p_0) \wedge (\Box\Diamond p_0 \rightarrow \Box\Box\Diamond p_1) & \\ \quad \wedge\Box(\Diamond\Box\Diamond p_0 \wedge p_0 \rightarrow \Box p_1))\{p_0 \wedge \Diamond\neg p_3/p_1\} & i=0 \\ \Box\mathrm{A4}\{p_1/p_0\} \wedge \Box C(i-1) \wedge \Box\mathrm{A4}\{\Diamond p_1/p_0\} & \text{otherwise} \end{cases}$$

E as in *k_t4p_p*.

5.3 Formulas for KT

kt_45_p

Idea: $\mathsf{KT} \vdash \mathrm{A5}\{\Box p_0/p_0\} \wedge \Box\mathrm{A5}\{\neg p_0/p_0\} \rightarrow \mathrm{A4}$.
Hiding: The left hand side occurs with 1 to n $\Box$, the right hand side occurs just with n $\Box$ in front. Additionally some superfluous instances, and the whole formula in negation normal form.

$$kt\_45\_p(n) :\equiv \mathrm{nnf}(\bigvee\nolimits_{i=1,\ldots,n}(\Box^n\mathrm{A4} \vee \neg\Box^i\mathrm{D}_2 \vee \neg\Box^i\mathrm{A5}\{\Diamond\neg p_0/p_0\} \vee \neg\Box^i\Box\mathrm{A5} \vee \neg\Box^i\mathrm{B}))$$

kt_45_n

Idea: A5 is not provable in KT plus any instances of A4.
Hiding: As for *kt_45_p*.

$kt\_45\_n(n) :\equiv \mathrm{nnf}(\bigvee_{i=1,\ldots,n}(\Box^n(\Box p_0 \vee \Box\Diamond\neg p_0) \vee \neg\Box^i \mathrm{A4} \vee \neg\Box^i \mathrm{A4}\{\Diamond p_0/p_0\} \vee \neg\Box^i \mathrm{T}$
$\vee \neg\Box^i \mathrm{A4}\{\Diamond p_0 \to p_0/p_0\} \vee \neg\Box^i \mathrm{A4}\{\Box p_0 \to p_0/p_0\}))$

kt_branch_p

Idea: The branching formula as described in [6], plus a negation symbol in front and the additional subformula $\neg\Box^n p_{(n \text{ div } 3)+1}$ in order to make the formula provable.

$kt\_branch\_p(n) :\equiv \neg(p_{100} \wedge \neg p_{101} \wedge \Box^n(bdepth(n) \wedge det(n) \wedge branching(n))) \vee \neg\Box^n p_{(n \text{ div } 3)+1}$

bdepth, *det*, *branching* as in *k_branch_p*.

kt_branch_n

Idea: The branching formula as defined in [6].

$kt\_branch\_n(n) :\equiv \neg(p_{100} \wedge \neg p_{101} \wedge \Box^n(bdepth(n) \wedge det(n) \wedge branching(n)))$

bdepth, *det*, *branching* as in *k_branch_p*.

kt_dum_p

Idea: $\mathsf{KT} \vdash \mathrm{A4}\{\Box(p_0 \to \Box p_0) \to p_0\} \wedge \Box\mathrm{A4} \wedge \mathrm{Dum} \wedge \mathrm{Dum}\{p_0 \to \Box p_0/p_0\} \to \mathrm{Dum}_1$.

Hiding: Some of he formulas on the left hand side of the implication occur with 1 to $n-1$ $\Box$ in front, the right hand side occurs just once with $(n \text{ div } 2)+1$ $\Box$ in front.

$kt\_dum\_p(n) :\equiv \bigwedge_{i=1,\ldots,n \text{ div } 2}(\Box^i \mathrm{A4}) \wedge \neg\Box^{(n \text{ div } 2)+1}\mathrm{Dum}_1$
$\to \Diamond^{(n \text{ div } 2)+1}\neg(\mathrm{A4}\{\Box(p_0 \to \Box p_0) \to p_0/p_0\} \wedge \Box\mathrm{A4} \wedge \mathrm{Dum} \wedge \mathrm{Dum}\{p_0 \to \Box p_0/p_0\})$
$\vee \bigvee_{i=n \text{ div } 2+2,\ldots,n-1}(\Diamond^i\neg\mathrm{A4})$

kt_dum_n

Idea: Grz is not provable in KT plus any instances of Dum.

Hiding: As for *kt_dum_p*.

$kt\_dum\_n(n) :\equiv \bigwedge_{i=1,\ldots,n \text{ div } 2}(\Box^i(\Box\mathrm{A4} \wedge \mathrm{Dum})) \wedge \neg\Box^{n \text{ div } 2+1}\mathrm{Grz}$
$\to \Diamond^{n \text{ div } 2+1}\neg(\mathrm{A4}\{\Box(p_0 \to \Box p_0) \to p_0/p_0\} \wedge \Box\mathrm{A4} \wedge \mathrm{Dum} \wedge \mathrm{Dum}\{p_0 \to \Box p_0/p_0\})$
$\vee \bigvee_{i=n \text{ div } 2+2,\ldots,n-1}(\Diamond^i\neg(\Box\mathrm{A4} \wedge \mathrm{Dum}))$

kt_grz_p

Idea: $\mathsf{KT} \vdash \Box\mathrm{Grz} \wedge \mathrm{Grz}\{C() \wedge (\mathrm{A4}\{C()/p_0\})/p_0\} \to \mathrm{Grz}_1$, where C is defined as below.

Hiding: Many superfluous instances with iterated $\Box$ inside the instances. The subformulas $\Diamond\Box\neg p_0$ and $\Diamond p_0$ are the parts of an instance of T.

$kt\_grz\_p(n) :\equiv \Box\mathrm{Grz}\{p_2/p_0\} \wedge \bigwedge_{i=1,\ldots,n-1}(l(i)) \wedge \mathrm{Grz}\{C() \wedge (\mathrm{A4}\{C()/p_0\})/p_0\}$
$\to \mathrm{Grz}_1\{p_1/p_0\} \vee \mathrm{Grz}_1\{p_2/p_0\} \wedge \Diamond\Box\neg p_0 \vee \mathrm{Grz}_1\{p_3/p_0\} \vee \mathrm{Grz}_1\{p_2/p_0\} \wedge \Diamond p_0$

l, $l2$, C as in *k_grz_p*.

kt_grz_n

Idea: A5 is not provable in KT plus instances of Grz_1.

Hiding: As for *k_grz_p*.

$kt\_grz\_n(n) :\equiv \Box\mathrm{Grz}_1\{p_2/p_0\} \wedge \bigwedge_{i=1,\ldots,n-1}(l(i)) \wedge \mathrm{Grz}_1\{C() \wedge \mathrm{A4}\{C()/p_0\}/p_0\}$
$\to \mathrm{A5}\{p_1/p_0\} \vee \mathrm{A5}\{p_2/p_0\} \vee \mathrm{A5}\{p_3/p_0\}$

$$l(i) :\equiv \begin{cases} \mathrm{Grz}_1\{l2(i \text{ div } 4)/p_0\} & i \bmod 4 = 0 \\ \mathrm{Grz}_1\{\Box l2(i \text{ div } 4) \vee p_1/p_0\} & i \bmod 4 = 1 \\ \mathrm{Grz}_1\{\Box l2(i \text{ div } 4) \vee p_1 \vee p_2/p_0\} & i \bmod 4 = 2 \\ \mathrm{Grz}_1\{\Box l2(i \text{ div } 4) \vee p_1 \vee p_2 \vee p_3/p_0\} & \text{otherwise} \end{cases}$$

$l2$, C as in *k_grz_p*.

kt_md_p

Idea: In backward proof search, we have to find the way to $g(n, n, \neg p_1)$ through a lot of $\Box$ and $\Diamond$.

$kt\_md\_p(n) :\equiv$
$p_1 \vee \bigvee_{i=1,\ldots,n-1}(g(i,n,\neg p_1 \wedge \Diamond f(1,n,p_2)) \vee g(i,n,\neg p_1 \wedge \Diamond f(1,n,p_1)) \vee g(i,n,\neg p_2 \wedge \Diamond f(1,n,p_1)))$
$\vee\, g(n,n,\neg p_1)$
$g(i,n,A) :\equiv \begin{cases} A & i = 1 \\ f(i,n,g(i-1,n,A)) & \text{otherwise} \end{cases}$
$f(i,n,A) :\equiv \Diamond^{i-1}\Box\Diamond^{n-i}A$

kt_md_n

Idea: Similar to *kt_md_p*, but simpler. The subformula $g(n,n,\neg p_1)$ is missing in order to make the formulas not provable.

$kt\_md\_n(n) :\equiv p_1 \vee \bigvee_{i=1,\ldots,n-1}(g(i,n,\neg p_1 \wedge \Diamond f(1,n,p_1)))$
f, g as in *kt_md_p*.

kt_path_p

Idea, hiding: As for *k_path_p*, but we use new variables on each level of the labyrinth. (The formulas *k_path_p* collapse in KT since $\mathsf{KT} \vdash \Diamond^n A \to A$.)

$kt\_path\_p(n) :\equiv (\Box p_{11} \vee \Box p_{10+path\_el(1,n)} \vee \Box p_{13} \vee \Box p_{15})$
$\vee \bigvee_{i=1,\ldots,n;j=1,\ldots,6}(left\_to\_right(i,j,n) \vee right\_to\_left(i,j,n))$
$\vee\, (\Diamond^n\neg p_{10n+2} \vee \Diamond^n\neg p_{10n+4} \vee \Diamond^n\neg p_{10n+path\_el(n,n)} \vee \Diamond^n\neg p_{10n+6})$
path_el, *left_to_right*, *right_to_left*, *delete*, *modg* as in *k_path_p*.
$lists2fml(l,k,s) :\equiv \bigvee_{x\in s}(\Diamond^l(\neg p_{10l+k} \wedge \Box p_{10(l+1)+x}))$
$lists2fml\_back(l,k,s) :\equiv \bigvee_{x\in s}(\Diamond^l(\neg p_{10l+x} \wedge \Box p_{10(l+1)+k}))$

kt_path_n

Idea, hiding: As for *k_path_n*, but we use new variables on each level of the labyrinth. (The formulas *k_path_p* collapse in KT since $\mathsf{KT} \vdash \Diamond^n A \to A$.)

$kt\_path\_n(n) :\equiv$
$(\Box p_{11} \vee \Box p_{10+path\_el(1,n+1)} \vee \Box p_{13} \vee \Box p_{15})$
$\vee \bigvee_{i=1,\ldots,n+1;j=1,\ldots,6}(left\_to\_right(i,j,n+1) \vee right\_to\_left(i,j,n+1))$
$\vee\, (\Diamond^{n+1}\neg p_{10(n+1)+2} \vee \Diamond^{n+1}\neg p_{10(n+1)+4} \vee \Diamond^{n+1}\neg p_{10(n+1)+path\_el(n+1,n+1)} \vee \Diamond^{n+1}\neg p_{10(n+1)+6})$
path_el, *left_to_right*, *right_to_left*, *delete*, *modg* as in *k_path_n*.
lists2fml, *lists2fml_back* as in *kt_path_p*.

kt_ph_p

Idea, hiding: As for *k_ph_p*.

$kt\_ph\_p(n) :\equiv left(n) \to \Diamond right(n)$
left, *right*, l as in *k_ph_p*.

kt_ph_n

Idea, hiding: As for *k_ph_n*.

$kt\_ph\_n(n) :\equiv left(n) \to \Diamond right(n)$
left, *right*, l, *l2* as in *k_ph_n*.

kt_poly_p

Idea, hiding: As for *k_poly_p*.

$kt\_poly\_p(n) :\equiv \begin{cases} poly(5n+1) & n \bmod 2 = 0 \\ poly(5n) & n \bmod 2 = 1 \end{cases}$
poly as in *k_poly_p*.
$f(i,n) :\equiv \begin{cases} \mathsf{false} & i = 0 \\ \Diamond(f(i-1,n) \vee \Diamond^{i+2}(p_n \leftrightarrow p_1)) \vee \Box p_{i+2} & i = n \\ \Diamond(f(i-1,n) \vee \Diamond^{i+2}(p_i \leftrightarrow p_{i+1})) \vee \Box p_{i+2} & \text{otherwise} \end{cases}$

kt_poly_n

Idea, hiding: As for *k_poly_n*.

$kt\_poly\_n(n) :\equiv \begin{cases} poly(3n) & n \bmod 2 = 0 \\ poly(3n+1) & n \bmod 2 = 1 \end{cases}$

poly, f as in *kt_poly_p*.

kt_t4p_p

Idea, hiding: As for *k_t4p_p*.

$kt\_t4p\_p(n) :\equiv E(n) \vee \mathrm{nnf}(\neg C(n)) \vee \Diamond p_4$

$C(i) :\equiv \begin{cases} ((\Box\Diamond p_0 \to \Box\Box\Diamond p_1) \wedge \Box(\Diamond\Box\Diamond p_0 \wedge p_0 \to \Box p_1) \wedge \Box\Diamond p_1)\{p_0 \wedge \Diamond\neg p_3/p_1\} & i = 0 \\ \Box\mathrm{A4}\{p_1/p_0\} \wedge \Box C(i-1) & \text{otherwise} \end{cases}$

E as in *k_t4p_p*.

kt_t4p_n

Idea, hiding: As for *k_t4p_n*.

$kt\_t4p\_n(n) :\equiv E(2n-1) \vee \mathrm{nnf}(\neg C(2n-1)) \vee \Diamond p_4$

$C(i) :\equiv \begin{cases} ((\Box\Diamond p_0 \to \Diamond p_1) \wedge (\Box\Diamond p_0 \to \Box\Box\Diamond p_1) \\ \quad \wedge \Box(\Diamond\Box\Diamond p_0 \wedge p_0 \to \Box p_1))\{p_0 \wedge \Diamond\neg p_3/p_1\} & i = 0 \\ \Box\mathrm{A4}\{p_1/p_0\} \wedge \Box C(i-1) & \text{otherwise} \end{cases}$

E as in *k_t4p_p*.

5.4 Formulas for S4

s4_45_p

Idea: $\mathsf{S4} \vdash \mathrm{A5}\{\Box p_0/p_0\} \wedge \Box\mathrm{A5}\{\neg p_0/p_0\} \to \mathrm{A5}$.

Hiding: The left hand side occurs with 1 to n $\Box$, the right hand side occurs just with n $\Box$ in front. Additionally some superfluous instances, and the whole formula in negation normal form.

$s4\_45\_p(n) :\equiv \mathrm{nnf}(\bigvee_{i=1,\ldots,n}(h(n,\mathrm{A5}) \vee \neg h(i,\mathrm{D}_2) \vee \neg h(i,\mathrm{A5}\{\Diamond\neg p_0/p_0\}) \vee \neg h(i,\Box\mathrm{A5}) \vee \neg h(i,\mathrm{B})))$

$h(i,A) :\equiv \begin{cases} A & i = 0 \\ \Box p_0 \vee \Box h(i-1,A) \vee \Box p_1 & \text{otherwise} \end{cases}$

s4_45_n

Idea: A5 is not provable in S4.

Hiding: As for *s4_45_p*.

$s4\_45\_n(n) :\equiv \mathrm{nnf}(\bigvee_{i=1,\ldots,n}(h(n,(\Box p_0 \vee \Box\Diamond\neg p_0)) \vee \neg h(i,\mathrm{A4}) \vee \neg h(i,\mathrm{A4}\{\Diamond p_0/p_0\}) \vee \neg h(i,\mathrm{T}) \vee \neg h(i,\mathrm{A4}\{\Box p_0 \to p_0/p_0\}) \vee \neg h(i,\mathrm{T}\{\Box p_0 \to p_0/p_0\})))$

h as in *s4_45_p*.

s4_branch_p

Idea: The branching formula as described in [6], plus a negation symbol in front and the additional subformula $\Box p_{(n \text{ div } 3)+1}$ in order to make the formula provable.

$s4\_branch\_p(n) :\equiv \neg(p_{100} \wedge \neg p_{101} \wedge \Box(bdepth(n) \wedge det(n) \wedge branching(n))) \vee \neg\Box p_{(n \text{ div } 3)+1}$

bdepth, *det*, *branching* as in *k_branch_p*.

s4_branch_n

Idea: The branching formula as defined in [6].

$s4\_branch\_n(n) :\equiv \neg(p_{100} \wedge \neg p_{101} \wedge \Box(bdepth(n) \wedge det(n) \wedge branching(n)))$

bdepth, *det*, *branching* as in *k_branch_p*.

s4_grz_p

Idea: $\mathsf{S4} \vdash \Box\mathrm{Grz} \wedge \mathrm{Grz}\{C() \wedge (\mathrm{A4}\{C()/p_0\})/p_0 \ \to \mathrm{Grz}_1$, where C is defined as below.

Hiding: Many superfluous instances with iterated $\Box$ inside the instances. The subformulas $\neg\Box\Diamond p_0$ and $\neg\Diamond\neg(\Box\Diamond p_0 \vee p_1)$ are the parts of a weakened instance of *A4*.

$s4\_grz\_p(n) :\equiv$
$\quad \Box\mathrm{Grz}\{p_2/p_0\} \wedge \bigwedge_{i=1,\ldots,n-1}(l(i)) \wedge \mathrm{Grz}\{C() \wedge (\mathrm{A4}\{C()/p_0\})/p_0\}$
$\quad \rightarrow \mathrm{Grz}_1\{p_1/p_0\} \vee \mathrm{Grz}_1\{p_2/p_0\} \wedge \neg\Box\Diamond p_0 \vee \mathrm{Grz}_1\{p_3/p_0\} \vee \mathrm{Grz}_1\{p_2/p_0\} \wedge \neg\Diamond\neg(\Box\Diamond p_0 \vee p_1)$
l, *l2*, C as in *k_grz_p*.

s4_grz_n

Idea: A5 is not provable in S4 plus instances of Grz_1.
Hiding: As for *k_grz_p*.

$s4\_grz\_n(n) :\equiv \Box\mathrm{Grz}_1\{p_2/p_0\} \wedge \bigwedge_{i=1,\ldots,n-1}(l(i)) \wedge \mathrm{Grz}_1\{C() \wedge \mathrm{A4}\{C()/p_0\}/p_0\}$
$\qquad \rightarrow \mathrm{A5}\{p_1/p_0\} \vee \mathrm{A5}\{p_2/p_0\} \vee \mathrm{A5}\{p_3/p_0\}$

$$l(i) :\equiv \begin{cases} \mathrm{Grz}_1\{l2(i \text{ div } 4)/p_0\} & i \bmod 4 = 0 \\ \mathrm{Grz}_1\{\Box l2(i \text{ div } 4) \vee p_1/p_0\} & i \bmod 4 = 1 \\ \mathrm{Grz}_1\{\Box l2(i \text{ div } 4) \vee p_1 \vee p_2/p_0\} & i \bmod 4 = 2 \\ \mathrm{Grz}_1\{\Box l2(i \text{ div } 4) \vee p_1 \vee p_2 \vee p_3/p_0\} & \text{otherwise} \end{cases}$$

l2, C as in *k_grz_p*.

s4_ipc_p

Idea: We embed a formula that is provable in intuitionistic propositional logic in S4.

$s4\_ipc\_p(n) :\equiv \bigwedge_{i=1,\ldots,n}(f(i,n)) \rightarrow \mathsf{false}$
$f(i,n) :\equiv \Box(\Box(\Box p_i \rightarrow \bigwedge_{j=1,\ldots,n}(\Box p_j)) \rightarrow \mathsf{false})$

s4_ipc_n

Idea: We embed a formula that is not provable in intuitionistic propositional logic in S4.

$s4\_ipc\_n(n) :\equiv \bigwedge_{i=1,\ldots,n}(f2(i,n)) \rightarrow \mathsf{false}$

$$f2(i,n) :\equiv \begin{cases} \mathsf{true} & i = (n+1) \text{ div } 2 \\ f(i,n) & \text{otherwise} \end{cases}$$

f as in *s4_ipc_p*.

s4_md_p

Idea: In backward proof search, we have to find the way to $g(n,n,\neg p_1)$ through a lot of $\Box$ and $\Diamond$.

$s4\_md\_p(n) :\equiv$
$\quad p_1$
$\quad \vee \bigvee_{i=1,\ldots,n-1}(g(i,n,\neg p_1 \wedge \Diamond f(1,n,p_2)) \vee g(i,n,\neg p_1 \wedge \Diamond f(1,n,p_1)) \vee g(i,n,\neg p_2 \wedge \Diamond f(1,n,p_1)))$
$\quad \vee g(n,n,\neg p_1)$

g as in *kt_md_p*.
$f(i,n,A) :\equiv h(i-1, \Box h(n-i,A))$

$$h(i,A) :\equiv \begin{cases} A & i = 0 \\ \Diamond h(i-1,A) \vee p_2 & \text{otherwise} \end{cases}$$

s4_md_n

Idea: Similar to *s4_md_p*, but simpler. The subformula $g(n,n,\neg p_1)$ is missing in order to make the formulas not provable.

$s4\_md\_n(n) :\equiv p_1 \vee \bigvee_{i=1,\ldots,n-1}(g(i,n,\neg p_1 \wedge \Diamond f(1,n,p_1)))$
f, g, h as in *s4_md_p*.

s4_path_p

Idea, hiding: As for *kt_path_p*.

$s4\_path\_p(n) :\equiv (\Box\Box p_{11} \vee \Box\Box p_{10+path\_el(1,n)} \vee \Box\Box p_{13} \vee \Box\Box p_{15})$
$\qquad \vee \bigvee_{i=1,\ldots,n;j=1,\ldots,6}(left\_to\_right(i,j,n) \vee right\_to\_left(i,j,n))$
$\qquad \vee \Diamond(\Diamond\neg p_{10n+2} \vee \Diamond\neg p_{10n+4} \vee \Diamond\neg p_{10n+path\_el(n,n)} \vee \Diamond\neg p_{10n+6})$

path_el, *left_to_right*, *right_to_left*, *delete*, *modg*, *lists2fml*, *lists2fml_back* as in *kt_path_p*.

s4_path_n

Idea, hiding: As for *kt_path_n*.

$s4\_path\_n(n) :\equiv$
$(\Box\Box p_{11} \vee \Box\Box p_{10+path\_el(1,n+1)} \vee \Box\Box p_{13} \vee \Box\Box p_{15})$
$\vee \bigvee_{i=1,\ldots,n+1;j=1,\ldots,6}(left\_to\_right(i,j,n+1) \vee right\_to\_left(i,j,n+1))$
$\vee \Diamond(\Diamond\neg p_{10(n+1)+2} \vee \Diamond\neg p_{10(n+1)+4} \vee \Diamond\neg p_{10(n+1)+path\_el((n+1),(n+1))} \vee \Diamond\neg p_{10(n+1)+6})$

path_el, *left_to_right*, *right_to_left*, *delete*, *modg*, *lists2fml*, *lists2fml_back* as in *kt_path_n*.

s4_ph_p

Idea: The pigeonhole formulas. We assume $n < 100$.
Hiding: Some $\Box$ and $\Diamond$.

$s4\_ph\_p(n) :\equiv left(n) \rightarrow \Diamond right(n)$
$right(n) :\equiv \bigvee_{j=1,\ldots,n;i_1=1,\ldots,n+1;i_2=i_1+1,\ldots,n+1}(\Diamond(l(i_1,j) \wedge l(i_2,j)))$
left, l as in *k_ph_p*.

s4_ph_n

Idea: The pigeonhole formulas, with one missing conjunct on the right hand side. We assume $n < 100$.
Hiding: Some $\Box$ and $\Diamond$.

$s4\_ph\_n(n) :\equiv left(n) \rightarrow \Diamond right(n)$
$right(n) :\equiv \bigvee_{j=1,\ldots,n;i_1=1,\ldots,n+1;i_2=i_1+1,\ldots,n+1}(\Diamond(l2(n,i_1,j) \wedge l2(n,i_2,j)))$
left, l as in *k_ph_p*.
$l2(n,i,j) :\equiv \begin{cases} \neg l(i,j) & i = j, i = (2n) \text{ div } 3 + 1 \\ l(i,j) & \text{otherwise} \end{cases}$

s4_s5_p

Idea: A formula that is provable in S5 embedded in S4.
Hiding: Some superfluous subformulas.

$s4\_s5\_p(n) :\equiv \Box\Diamond(\Box\bigvee_{i=1,\ldots,3n-2}(p_i \leftrightarrow p_{i+1}) \vee \Box p_{3n} \vee f(1,3n-1)) \vee \Box(\Diamond p_1 \rightarrow \neg p_{3n})$
$f(i,n) :\equiv \begin{cases} \mathsf{false} & i = n \\ \Diamond(p_i \wedge \neg p_{i+1} \vee \neg p_i \wedge p_{i+1}) \vee \Box f(i+1,n) & \text{otherwise} \end{cases}$

s4_s5_n

Idea: A formula that is not provable in S5 embedded in S4.
Hiding: Some superfluous subformulas.

$s4\_s5\_n(n) :\equiv \Box\Diamond(\Box p_{6n} \vee f(1,6n-5)) \vee \Box(\Diamond p_1 \rightarrow \neg p_{6n})$
f as in *s4_s5_p*.

s4_t4p_p

Idea, hiding: As for *k_t4p_p*.

$s4\_t4p\_p(n) :\equiv E(n) \vee \text{nnf}(\neg C(2n-1)) \vee \Diamond p_4$
$C(i) :\equiv \begin{cases} (\Box(\Diamond\Box\Diamond p_0 \wedge p_0 \rightarrow \Box p_1) \wedge \Box\Diamond p_1)\{p_0 \wedge \Diamond\neg p_3/p_1\} & i = 0 \\ \text{Dum}\{p_1/p_0\} \wedge \Box C(i-1) & \text{otherwise} \end{cases}$
E as in *k_t4p_p*.

s4_t4p_n

Idea, hiding: As for *k_t4p_n*.

$s4\_t4p\_n(n) :\equiv E(2n-1) \vee \text{nnf}(\neg C(4n-1)) \vee \Diamond p_4$
$C(i) :\equiv \begin{cases} ((\Box\Diamond p_0 \rightarrow \Diamond p_1) \wedge \Box(\Diamond\Box\Diamond p_0 \wedge p_0 \rightarrow \Box p_1))\{p_0 \wedge \Diamond\neg p_3/p_1\} & i = 0 \\ \text{Dum}\{p_1/p_0\} \wedge \Box C(i-1) & \text{otherwise} \end{cases}$
E as in *k_t4p_p*.

6 Benchmark results for the LWB

Prover: Logics Workbench (LWB), version 1.0 (`http://lwbwww.unibe.ch:8080/LWBinfo.html`, [7]).
Backward proof search in two-sided sequent calculi. Use-check (similar to [8]) helps to cut off superfluous branches. In the case of S4, a loop-check is used (see [9]).
Programming language: C++ . Compiler: Sun C++ 4.0.1 . Operating system: Solaris 2.4 .

Availability: The binaries of the LWB 1.0 are available via the LWB home page (choose `about the` LWB, `install the` LWB).
You can also use the LWB 1.0 via WWW. Choose `run a session via WWW` on the LWB home page and type in your request.

Additional facilities of the prover:

- graphical user interface
- built-in programming language
- progress indicator: a slider shows how the proof search is advancing
- various functions to convert formulas

Hardware: Sun SPARCstation 5, main memory: 80MB, 1 CPU (70 MHz microSPARC II).

Timing: The timing includes the parsing of the formulas and the construction of the corresponding data structure. The files loaded by the LWB have the following form:

```
load(k);
timestart; provable(box p0 -> box box p0); timestop;
quit;
```

Results:

class	$n_{\mathrm{p},i}$	class	$n_{\mathrm{n},i}$
k_branch_p	6	*k_branch_n*	7
k_d4_p	8	*k_d4_n*	6
k_dum_p	13	*k_dum_n*	19
k_grz_p	7	*k_grz_n*	13
k_lin_p	11	*k_lin_n*	8
k_path_p	12	*k_path_n*	10
k_ph_p	4	*k_ph_n*	8
k_poly_p	8	*k_poly_n*	11
k_t4p_p	8	*k_t4p_n*	7

class	$n_{\mathrm{p},i}$	class	$n_{\mathrm{n},i}$
kt_45_p	5	*kt_45_n*	4
kt_branch_p	5	*kt_branch_n*	6
kt_dum_p	5	*kt_dum_n*	10
kt_grz_p	6	*kt_grz_n*	> 20
kt_md_p	5	*kt_md_n*	5
kt_path_p	10	*kt_path_n*	9
kt_ph_p	4	*kt_ph_n*	8
kt_poly_p	14	*kt_poly_n*	2
kt_t4p_p	5	*kt_t4p_n*	7

class	$n_{\mathrm{p},i}$	class	$n_{\mathrm{n},i}$
s4_45_p	3	*s4_45_n*	5
s4_branch_p	11	*s4_branch_n*	7
s4_grz_p	9	*s4_grz_n*	> 20
s4_ipc_p	8	*s4_ipc_n*	7
s4_md_p	8	*s4_md_n*	6
s4_path_p	8	*s4_path_n*	6
s4_ph_p	4	*s4_ph_n*	8
s4_s5_p	4	*s4_s5_n*	9
s4_t4p_p	9	*s4_t4p_n*	12

In order to make absolutely clear how we obtained the numbers in the tables above from the run times, we give the run times for some of the formulas in the classes *k_branch_p*. *k_branch_p*(6) is the last formula that could be decided in less than 100 seconds; therefore we enter the numbers 6 on the left hand side of the first line of the table for K.

formula	run time (in seconds)
$k\_branch\_p(1)$	0.02
$k\_branch\_p(2)$	0.06
$k\_branch\_p(3)$	0.28
$k\_branch\_p(4)$	1.80
$k\_branch\_p(5)$	11.89
$k\_branch\_p(6)$	74.64
$k\_branch\_p(7)$	503.94

7 Availability of the formulas

You can get the first 15 formulas of each class as well as the LWB programs that generated these formulas. Load the LWB home page `http://lwbwww.unibe.ch:8080/LWBinfo.html` in a WWW browser and then choose the item `benchmarks`.
The formulas are in infix notation. The connectives are `~` for $\neg$, `&` for $\wedge$, `v` for $\vee$, `->` for $\rightarrow$, `<->` for $\leftrightarrow$. No brackets are omitted in order to make the conversion into other formats easier.
It can happen that for some classes you need more than 15 formulas. Then you have several possibilities:

- Write a procedure that generates the formulas in this class according to the definitions in section 5. Please make sure that you generate the same formulas as we do by comparing the first 15 formulas.
- Get the LWB procedures we used to generate the formulas, install the LWB, and generate the required formulas.
- Use the LWB via WWW. You have to replace the `read` statements in the files by the contents of the corresponding files.

In order to avoid inconsistencies between the formulas in this section and the formulas used in section 6, we implemented a Perl program that automatically converts the LaTeX definitions into LWB programs.

8 Conclusion

We stated postulates for benchmark methods for proof search procedures. and presented a benchmark method for the propositional modal logics K, KT, S4 that largely satisfies these postulates.
Until now often just a few arbitrary — and often very easy — formulas were used to judge the efficiency of such theorem provers. With our method it is now possible to compare these provers in a standardized and fair way.
Since we use but scalable formulas, the method will still be applicable when there are much faster machines and more efficient proof search procedures.
Of course the postulates are not limited to benchmark tests for modal logic theorem provers, but seem also reasonable for other logics.

References

[1] P. Balsiger and A. Heuerding. Comparison of theorem provers for modal logics – introduction and summary. In H. de Swart, editor, *TABLEAUX '98*, LNAI 1397, pages 25–26, 1998.

[2] L. Catach. Tableaux: A general theorem prover for modal logics. *Journal of Automated Reasoning*, 7:489–510, 1991.

[3] H. de Swart, editor. *Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX '98*, LNAI 1397. Springer, 1998.

[4] S. Demri. Uniform and non uniform strategies for tableaux calculi for modal logics. *Journal of Applied Non-Classical Logics*, 5(1):77–96, 1995.

[5] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures – the case study of modal K. In *CADE 96*, LNCS, 1996.

[6] J. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[7] A. Heuerding, G. Jäger, S. Schwendimann, and M. Seyfried. Propositional logics on the computer. In P. Baumgartner, R. Hähnle, and J. Posegga, editors, *Tableaux 95*, LNCS 918, pages 310–323, 1995.

[8] A. Heuerding and S. Schwendimann. On the modal logic K plus theories. In H. Kleine Büning, editor, *CSL 95*, LNCS 1092, pages 308–319, 1996.

[9] A. Heuerding, M. Seyfried, and H. Zimmermann. Efficient loop-check for backward proof search in some non-classical propositional logics. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Tableaux 96*, LNCS 1071, pages 210–225, 1996.

[10] F. Pelletier. Seventy-five problems for testing automatic theorem provers. *Journal of Automated Reasoning*, 2:191–216, 1986.

[11] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings 10th AAAI*, pages 47–59, 1992.

[12] G. Sutcliffe, C. Suttner, and T. Yemenis. The TPTP problem library. In A. Bundy, editor, *CADE 12*, LNCS 814, pages 252–266, 1994.