Destructive Modal Resolution *

Melvin Fitting
mlflc@cunyvm.cuny.edu
Dept. Mathematics and Computer Science
Lehman College (CUNY), Bronx, NY 10468
Depts. Computer Science, Philosophy, Mathematics
Graduate Center (CUNY), 33 West 42nd Street, NYC, NY 10036 †

August 3, 1989

Abstract

We present non-clausal resolution systems for propositional modal logics whose Kripke models do not involve symmetry, and for first order versions whose Kripke models do not involve constant domains. We give systems for K, T, $K4$ and $S4$; other logics are also possible. Our systems do not require preliminary reduction to a normal form and, in the first order case, intermingle resolution steps with Skolemization steps.

1 Introduction

One of the attractive features of non-classical logics is that they provide magnifying glasses for the closer examination of techniques developed for the classical case. An obvious example of such a technique is that of reduction to normal form — clause form, generally. Most non-classical logics have no simple notion of a normal form, or maybe none at all, so one is forced to consider just what the reduction to normal form was intended to accomplish. This is a good question even in the classical case. A logic like propositional $S5$ has a reasonable normal form — one can convert a formula into an equivalent version with no nested modalities. But this does not extend to the first order setting, so one is forced to reconsider the role of quantifiers and their relationships with the propositional connectives. In short, things that are taken for granted classically can not be in non-classical settings. Thus we have a kind of ready-made laboratory for the deeper investigation of familiar techniques that we may have thought were fully understood.

A case in point is the basic notion of resolution itself. Classically there are several variations that go under the general heading of resolution, but there is an obvious family resemblance. It is clear they are all versions of the same idea. But when it comes to extending resolution techniques

*The author wishes to thank the referee of an earlier draft of this paper for helpful comments, and for catching a fundamental error.

†Research partly supported by NSF Grant CCR-8702307 and PSC-CUNY Grant 6-67295.

to various modal logics it turns out there are several competing ways of doing so, they have little resemblance to each other, and one can argue about which deserves the title resolution. See [18] for the kinds of issues involved. So far several distinct versions of resolution for modal logics have been proposed. [21] presents a system in which explicit notation is introduced to denote possible worlds. This is related to the tableau systems of [7] and some of those in [9] (see also [24]). Most competing systems do not use this device though. See [1], [2], [14], [6], [5], [3], [17], [19] and [20].

The idea of *context* is explicit or implicit in various systems proposed for modal logics. In a Kripke model a modal operator corresponds to a shift from one possible world to another. Different parts of a formula can be within the scopes of different modal operators, and so may not be applicable to the same worlds. Occurrences of the same literal may not mean the same thing in different contexts — within different nestings of modal operators. One solution has been to introduce notation for context, via prefixes or world paths, as mentioned above. This is not the approach we take here. We noted some time ago, in the framework of tableaux systems, that for certain modal logics only one context at a time needed to be considered, and when one shifted to a new context details about the old one could be forgotten. Logics amenable to such treatment tend to be those whose Kripke models do not involve symmetry — models that do not permit going back to earlier worlds, hence those for which earlier contexts will not come back to haunt us.

We call resolution systems of the sort presented here *destructive* because they loose information during a context shift. They are suitable for logics like K and $S4$ and are not suitable for $S5$, whose models require symmetry. In first order versions they are suitable for varying domain logics but not for constant domain ones (i.e. no Barcan formula). With constant domain models, whatever exists in any world exists here, so destruction of information during a context shift can be dangerous. Of course this is too loose a comment to have any real force. But the fact is, constant domain logics seem not treatable by methods like ours; varying domain logics are fairly straightforward.

The destructive resolution systems presented here are non-clausal, or rather the steps that classically would amount to a reduction to clause form can be intermingled with resolution applications. This is how we deal with the non-existence of simple clause forms in modal logics. Also we have had to add a special rule to cover disjunctions, one that is provably eliminable in the classical case. The added rule uses a higher type of object than a clause set. In this respect it is somewhat similar to the Splitting Rule, used in the Davis-Putnam theorem proving technique in classical logic [4], a rule not needed in classical resolution systems. We feel that this rule does not represent a pecularity of modal logic, but rather its absence of classical. Finally, the first order version of our system intermingles the steps of Skolemization with those of resolution, another technique that is possible, but not necessary classically.

Our resolution system is based on an intuitionistic system in [10], and is strongly influenced by tableau systems presented in [9] and implemented in [11]. We feel it has an advantage of simplicity over most of the systems already in the literature — that is, it is easier to understand. This is not a fundamental issue when it comes to automation, however, since a less intelligible system may be more efficient. We strongly urge a multiplicity of approaches, and the full investigation of all. Someday someone will begin the work of comparison, and then some lines will perhaps be abandoned. Until then, let us not be short on resolution.

Figure 1: The α and β cases.

α	α_1	α_2
$X \wedge Y$	X	Y
$\neg(X \vee Y)$	$\neg X$	$\neg Y$
$\neg(X \supset Y)$	X	$\neg Y$

β	β_1	β_2
$\neg(X \wedge Y)$	$\neg X$	$\neg Y$
$X \vee Y$	X	Y
$X \supset Y$	$\neg X$	Y

Figure 2: The ν and π cases.

ν	ν_0
$\Box X$	X
$\neg\Diamond X$	$\neg X$

π	π_0
$\neg\Box X$	$\neg X$
$\Diamond X$	X

2 Propositional Syntax

We write P, Q, ...for propositional variables, X, X_1, ..., for arbitrary formulas. We will be using the *uniform notation* of [22], extended to the modal case as in [9] and [11]. Doing so adds no complexity, and gives us the ability to treat many propositional connectives simultaneously. Consequently we take as primitive $\neg$, $\wedge$, $\vee$, $\supset$, $\Box$ and $\Diamond$. (Other binary connectives could be added to the list if desired, the uniform notation mechanism, as formulated in [22], can deal with anything except $\equiv$ and $\not\equiv$.) Formulas whose major connective is binary, and negations of such formulas, are divided into two categories: *conjunctives*, or α formulas, and *disjunctives*, or β formulas. For each α formula two *components*, α_1 and α_2 are defined. Similarly for each β formula, β_1 and β_2 are defined. The cases and their definitions are shown in Figure 1. The intuition here is that a formula of type α is true if and only if both corresponding α_1 and α_2 are true; β is true if and only if β_1 or β_2 is true.

In a similar way the modal cases are divided up into ν formulas (those that are necessary) and π formulas (those that are possible). Each has a single component, denoted ν_0 and π_0 respectively. These are given in Figure 2. Here the intuition requires minimal knowledge of Kripke models to present — we assume such knowledge. To say a formula of type ν is true at a possible world is to say ν_0 is true at every world accessible from that world; to say π is true is to say π_0 is true at some accessible world.

Finally we introduce special notation for generalized disjunctions and conjunctions. We will treat $[X_1, X_2, \ldots, X_n]$ as synonymous with $X_1 \vee X_2 \vee \ldots \vee X_n$ (parenthesized somehow), and $\langle X_1, X_2, \ldots, X_n \rangle$ as synonymous with $X_1 \wedge X_2 \wedge \ldots \wedge X_n$. As usual, $[\,]$ is identified with $false$, and $\langle\rangle$ with $true$.

If X_1, X_2, ..., X_n are formulas, we will refer to $[X_1, X_2, \ldots, X_n]$ as a *clause*, though this is not strictly correct since the X_i are not required to be literals. Likewise if each C_i is a clause, we will refer to $\langle C_1, C_2, \ldots, C_n \rangle$ as a *clause list*. Finally, there is one more level we will need: if L_1, L_2, ..., L_n are clause lists, we call $[L_1, L_2, \ldots, L_n]$ a *block*. Blocks are not needed in classical resolution,

though they were implicitly present in the Davis-Putnam method, [4], and are used explicitly in the discussion of it in [12].

3 Classical Propositional Resolution

For comparison purposes we begin with a classical system, formulated using the uniform notation presented above. The system does not require a previous reduction to clause form — the reduction steps are built in. This will be of more importance when we come to the modal case later on, though it is dispensable here. Also, *blocks* are present, though they play no role here. They will become significant when we introduce modal rules in the next section.

We begin with some terminology that will be useful to us. We call a clause list containing the empty clause *closed*, and a block *closed* if every clause list in it is closed.. We call a block *closable* if a closed block can be derived from it using rules of derivation to be given below. As usual with resolution, this is a refutation system; we say X is a *theorem* if the block $[\langle[\neg X]\rangle]$ is closable. In the classical system, blocks never contain more than one clause list, and so the block mechanism can be tacitly ignored for the present. Note that the rules below are of two kinds. Some are *replacement* rules; a formula, or a clause is removed and one or more new ones are added in its place. Others are *addition* rules; clauses are added, but none are removed. Now, the rules for deriving one block from another are as follows.

Propositional Reduction Rules

Double Negation An occurrence of a formula $\neg\neg Z$ may be *replaced* by an occurrence of Z. Schematically,

$$\frac{\neg\neg Z}{Z}$$

Conjunction If an α formula occurs in a clause C, the clause C may be *replaced* by two clauses, C_1 and C_2, which are like C except that C_1 contains α_1 in place of α and C_2 contains α_2 in place of α. Schematically,

$$\frac{[A_1, \ldots, A_n, \alpha]}{[A_1, \ldots, A_n, \alpha_1], [A_1, \ldots, A_n, \alpha_2]}$$

Disjunction An occurrence of a β formula may be *replaced* by the two formulas β_1 and β_2. Schematically,

$$\frac{\beta}{\beta_1, \beta_2}$$

Resolution Rule If a clause list contains two clauses C_1 and C_2 where C_1 contains Z and C_2 contains $\neg Z$ then the clause C may be *added* to the clause list, where C contains the members of C_1 except for occurrences of Z and the members of C_2 except for occurrences of $\neg Z$. Schematically,

$$\frac{[Z, X_1, \ldots, X_k], [\neg Z, Y_1, \ldots, Y_m]}{[X_1, \ldots, X_k, Y_1, \ldots, Y_m], [Z, X_1, \ldots, X_k], [\neg Z, Y_1, \ldots, Y_m]}$$

Example. The following is a proof, in this system, of $(P \wedge Q) \supset \neg(\neg P \vee \neg Q)$. We omit reasons for the steps.

1. $[\langle[\neg((P \wedge Q) \supset \neg(\neg P \vee \neg Q))]\rangle]$
2. $[\langle[P \wedge Q], [\neg\neg(\neg P \vee \neg Q)]\rangle]$
3. $[\langle[P \wedge Q], [\neg P \vee \neg Q]\rangle]$
4. $[\langle[P], [Q], [\neg P \vee \neg Q]\rangle]$
5. $[\langle[P], [Q], [\neg P, \neg Q]\rangle]$
6. $[\langle[\neg Q], [P], [Q], [\neg P, \neg Q]\rangle]$
7. $[\langle[\,], [\neg Q], [P], [Q], [\neg P, \neg Q]\rangle]$

A few observations are in order. First, the system is essentially non-deterministic. It is sound no matter in what order the rules are applied. Second, the system is complete, and remains so if the restriction is imposed that all Reduction Rules must be applied before any Resolution Rules. In fact this is essentially the conventional version of resolution, because use of the Reduction Rules until no more applications are possible results in a conversion to clause form. But we have not imposed such a requirement, thus allowing a version of non-clausal resolution.

Finally we introduce an additional rule that is sound but is not needed for completeness in the classical case — it is an admissible rule. When we come to the modal version this rule is no longer a derivable one, and must be taken as primitive. As we remarked earlier, it is similar to the Splitting Rule in the Davis-Putnam method, which preceded the introduction of resolution ([4], [16]). We call it a Special Case Rule. Unlike the rules above, it does not operate on formulas or clauses, but on *clause lists*. It is here that block structures are needed.

Definition 3.1 *We say C_1, C_2 is a* partition *of the clause C if C_1 and C_2 are disjoint and the members of C_1 together with those of C_2 are exactly the members of C.*

Special Case Rule If a clause list L contains a clause C, and C_1, C_2 is a partition of C, then L can be *replaced* by two clause lists, L_1 and L_2, where L_1 is like L except that it contains C_1 in place of C, and L_2 is like L but with C_2 in place of C. Schematically,

$$\frac{\langle[X_1, \ldots, X_n, Y_1, \ldots, Y_k], C_1, \ldots, C_m\rangle}{\langle[X_1, \ldots, X_n], C_1, \ldots, C_m\rangle, \langle[Y_1, \ldots, Y_k], C_1, \ldots, C_m\rangle}$$

4 Propositional Modal Logic — K

We assume Kripke models are familiar. K is the logic determined by the class of all normal propositional models with no restrictions imposed on the accessibility relation. Axiomatically it is characterized by taking all tautologies and all formulas of the form $\Box(A \supset B) \supset (\Box A \supset \Box B)$ as axioms and modus ponens and necessitation as rules of inference.

Definition 4.1 *For each clause C we define two related clauses as follows. $C^\sharp$ consists of all formulas ν_0 such that ν is in C. $C^\flat$ consists of all formulas π_0 such that π is in C.*

Example. If $C = [\Box(X \wedge Y), \neg\Box A, \Diamond(B \supset C), \neg C]$ then $C^\sharp = [X \wedge Y]$ and $C^\flat = [\neg A, B \supset C]$.

The basic facts about these operations are easy to state. If a clause C consists entirely of ν formulas and C is true at a possible world of a Kripke model, $C^\sharp$ is true at *every* accessible world. If C consists entirely of π formulas and C is true at a possible world, $C^\flat$ is true at *some* accessible world.

The resolution system we propose includes all the rules of Section 3, including the Special Case Rule, together with one more Reduction Rule. This additional rule is peculiar in that it acts 'globally' throughout a clause list, rather than 'locally'.

K–π **Rule** Suppose L is a clause list containing a clause C consisting entirely of π formulas. Then L may be *replaced* by the clause list L^* containing the following clauses: $C^\flat$ and, for each clause S in L consisting entirely of ν formulas, the clause $S^\sharp$.

Example. By this rule the clause list $\langle[P], [\Box Q, R], [\Box P, \neg\Diamond Q], [\neg\Box R, \Diamond S]\rangle$ may be replaced by $\langle[P, \neg Q], [\neg R, S]\rangle$. Here $C = [\neg\Box R, \Diamond S]$.

The intuition here is simple. Suppose we have a clause list L containing a clause C consisting entirely of π formulas. Let L^* be derived from L by the K–π Rule. Finally suppose L is satisfiable, that is, there is some Kripke model and some world Γ of it at which L is true. Since C is true at Γ, as we observed above, there is a world Δ, accessible from Γ, at which $C^\flat$ is true. Also for every clause S of L consisting entirely of ν formulas, $S^\sharp$ will be true at every world accessible from Γ, in particular at Δ. Thus L^* is also satisfiable, but at the world Δ instead of at the world Γ. In effect, we have established the soundness of the K–π rule.

Application of the K–π Rule involves a 'leap' from one world to another — a context shift. It is a forgetful leap — information is lost. Formulas are removed or modified, and so we call our resolution system *destructive*. In fact, every rule except the K–π rule turns a block into an *equivalent* block, but the K–π rule need not do so. It is because of this destructive aspect that symmetric logics, like $S5$, can not be treated by this kind of tableau. In symmetric logics we can return to an earlier world, and we need a mechanism for remembering what we knew when we were there last. For this purpose prefixes (as in [7] and [9]) or world paths (as in [21]) are used.

Example. The following is a proof, in this K system, of $(\Box P \wedge \Box Q) \supset \Box(P \wedge Q)$.

1. $[\langle[\neg((\Box P \wedge \Box Q) \supset \Box(P \wedge Q)]\rangle]$
2. $[\langle[\Box P \wedge \Box Q], [\neg\Box(P \wedge Q)]\rangle]$
3. $[\langle[\Box P], [\Box Q], [\neg\Box(P \wedge Q)]\rangle]$
4. $[\langle[P], [Q], [\neg(P \wedge Q)]\rangle]$
5. $[\langle[P], [Q], [\neg P, \neg Q]\rangle]$

6. $[\langle[\neg Q],[P],[Q],[\neg P,\neg Q]\rangle]$

7. $[\langle[\,],[\neg Q],[P],[Q],[\neg P,\neg Q]\rangle]$

Here 4) is from 3) by the $K-\pi$ Rule.

There are two somewhat hidden branch structures inherent in destructive resolution. The simplest concerns the explicitly modal rule. Suppose we have the clause list $\langle[\Box P],[\Diamond\neg P],[\Diamond Q]\rangle$. The K–π Rule can be applied to this in two different, and incompatible ways. If we apply it to the clause $[\Diamond Q]$ the clause list turns into $\langle[P],[Q]\rangle$ which can not lead to a closed clause list. On the other hand, if we use the clause $[\Diamond\neg P]$ we get the clause list $\langle[P],[\neg P]\rangle$, and this does allow us to derive the empty clause. Such a lack of confluence does not come up in the classical case. A deterministic implementation of this theorem prover will be required to remember branch points like this and be ready to backtrack if a closed clause list is not found. [15] and [23] have shown that propositional modal logics are inherently more complex than classical logic, so some such problem must be expected. On the other hand, this kind of branching is amenable to parallel treatment. Rather than choosing a clause to work with, a process can be started for each. These can proceed independently of each other, and if either produces a closed block, we have a proof. The issue is much like that of *or parallelism* in parallel logic programming.

The other kind of branch point involves the Special Case Rule, and here things are more complicated. If we have a block containing several clause lists, once again parallel treatment is possible; a process can be started for each clause list, and each must produce the empty clause for us to have a proof. In the propositional case this is straightforward, since clause lists are essentially independent of each other. More precisely, the only rule that involves any interaction is the Resolution Rule itself, and this is at the clause level, within clause lists. But, when we come to the first-order case we will see that what is done with one clause list in a block can have an effect on other clause lists — in effect, communication is involved. There is a strong similarity with *and parallelism* in logic programming.

The Special Case Rule can not be eliminated. It is not hard to see that the K valid formula $(\Box P\vee\Box Q)\supset(\neg\Diamond\neg P\vee\neg\Diamond\neg Q)$ is not provable without using it. This is in sharp contrast with the classical case, in which the rule is sound, but not needed for completeness. Useful restrictions on the applicability of the Special Case Rule would be most welcome.

Soundness of the resolution system for K is established in the usual way, except that Kripke models must be used. Call a block B *satisfiable* if there is some Kripke model and some possible world of it at which B is true. A closed block is not satisfiable. Also each of the rules proposed above preserves satisfiability. Then a proof of X demonstrates that the initial block $[\langle[\neg X]\rangle]$ was not satisfiable, and it follows that X is true at all worlds of all Kripke models.

The easiest way to demonstrate completeness is to use the Model Existence Theorem for K ([8], [9]). Actually, we need a slightly strengthened version; we state it and sketch the proof.

Definition 4.2 *Let $\mathcal{C}$ be a collection of sets of formulas. $\mathcal{C}$ is a* weak K consistency property *if it meets the following conditions for each $S\in\mathcal{C}$:*

1. *S does not contain any literal and its complement;*

2. $\neg\neg Z \in S \Longrightarrow S^0 \cup \{Z\} \in \mathcal{C}$, *where* S^0 *is* S *without* $\neg\neg Z$;
3. $\alpha \in S \Longrightarrow S^0 \cup \{\alpha_1, \alpha_2\} \in \mathcal{C}$, *where* S^0 *is* S *without* α;
4. $\beta \in S \Longrightarrow S^0 \cup \{\beta_1\} \in \mathcal{C}$ *or* $S^0 \cup \{\beta_2\} \in \mathcal{C}$, *where* S^0 *is* S *without* β;
5. $\pi \in S \Longrightarrow S^\sharp \cup \{\pi_0\} \in \mathcal{C}$.

The definition of K consistency property in [9] is similar to this, except that in items 2), 3) and 4), S appears in place of S^0.

Definition 4.3 *Call a set* U upward saturated *provided:*

1. $Z \in U \Longrightarrow \neg\neg Z \in U$;
2. $\alpha_1, \alpha_2 \in U \Longrightarrow \alpha \in U$;
3. $\beta_1 \in U$ *or* $\beta_2 \in U \Longrightarrow \beta \in U$.

The upward saturation *of* S, *denoted* S^u *is the smallest upward saturated set extending* S.

If $\mathcal{C}$ is a weak K consistency property, and we set $\mathcal{C}^*$ to be the collection of subsets of S^u for all $S \in \mathcal{C}$, $\mathcal{C}^u$ is a K consistency property extending $\mathcal{C}$. This takes a little checking, which we leave to you. But as an easy consequence, if S is a member of a *weak* K consistency property, S will also be a member of a K consistency property, and hence satisfiable in some K model, by the K Model Existence Theorem.

Suppose we call a finite set $\{X_1, \ldots, X_n\}$ of formulas *consistent* if the block $[\langle[X_1], \ldots, [X_n]\rangle]$ is not closable. It is easy to verify that this is a weak K consistency property. Most of the conditions are straightforward — the Special Case Rule is directly involved in checking the β condition. If X is not provable, the block $[\langle[\neg X]\rangle]$ is not closable, and so the set $\{\neg X\}$ is consistent. Then using the Model Existence Theorem for K, this set is satisfiable, and hence X is not K valid. Incidentally, this argument establishes the stronger fact that if X is provable, it has a proof in which all applications of the Resolution Rule are at the literal level. Further, one can restrict applications of the Special Case Rule to the splitting of two-element clauses into two singleton clauses.

Alternately, if one does not wish to use the Model Existence Theorem, one can prove completness as follows. Above we defined consistency for finite sets. Now call an infinite set consistent if every finite subset is. Then one can show a version of Lindenbaum's Lemma: every consistent set is extendable to a maximal consistent one. Now construct a Kripke model by taking the possible worlds to be the maximal consistent sets of formulas, and imitating the usual completeness proof for axiomatically formulated versions of K. In effect, one re-traces the argument for the Model Existence Theorem by doing so, however.

5 Other Propositional Modal Logics

Several modal logics have resolution systems similar to the one presented for K. In this section we briefly sketch systems for T, $K4$ and $S4$. The style of resolution system we have chosen does not lend itself directly to $S5$, or generally to any logic whose Kripke models must be symmetric. We have some brief comments on how to treat $S5$ later on, though.

T–ν **Rule** If a clause list contains a clause C with an occurrence of a formula ν then another clause may be *added* to the clause list that is like C except that it contains occurrences of ν_0 where C has ν. Schematically,

$$\frac{[\nu, A_1, \ldots, A_n]}{[\nu_0, A_1, \ldots, A_n], [\nu, A_1, \ldots, A_n]}$$

Adding the T–ν Rule to the K system of Section 4 yields a sound and complete system for propositional T.

For $K4$ we keep the classical rules of Section 3 but we substitute the following for the K–π Rule. (The difference between this and the K–π Rule is that clauses $S^\sharp$ are added, but clauses S are not removed.)

$K4$–π **Rule** Suppose L is a clause list containing a clause C consisting entirely of π formulas. Then L may be replaced by the clause list L^* consisting of the following clauses: $C^\flat$ and, for each clause S in L consisting entirely of ν formulas, the clauses S and $S^\sharp$.

The motivation for this rule is the same as it was for the K–π rule; conceptually it involves a context shift or "jump" from one possible world to another. But in $K4$ models, involving transitivity, the information that can be passed from one world to another is more elaborate than in K models. In particular, if $\Box X$ is true at a world of a transitive model, $\Box X$ will also be true at all accessible worlds. So again, if S consists entirely of ν formulas and S is true at a world of a transitive model, $S^\sharp$ will be true at any accessible world.

Finally for $S4$ two different versions are possible. Most simply, just use the classical propositional rules, the T–ν Rule and the $K4$–π Rule. Alternately, instead of the $K4$–π Rule one can use the following.

$S4$–π **Rule** Suppose L is a clause list containing a clause C consisting entirely of π formulas. Then L may be replaced by the clause list L^* consisting of the following clauses: $C^\flat$, and each clause S in L that consists entirely of ν formulas.

Example. The following is a proof, in the second $S4$ system, of $\Box(A \supset B) \supset \Box(\Diamond A \supset \Diamond B)$.

1. $[\langle[\neg(\Box(A \supset B) \supset \Box(\Diamond A \supset \Diamond B))]\rangle]$
2. $[\langle[\Box(A \supset B)], [\neg\Box(\Diamond A \supset \Diamond B)]\rangle]$
3. $[\langle[\Box(A \supset B)], [\neg(\Diamond A \supset \Diamond B)]\rangle]$
4. $[\langle[\Box(A \supset B)], [\Diamond A], [\neg\Diamond B]\rangle]$
5. $[\langle[\Box(A \supset B)], [A], [\neg\Diamond B]\rangle]$
6. $[\langle[A \supset B], [\Box(A \supset B)], [A], [\neg\Diamond B]\rangle]$
7. $[\langle[\neg B], [A \supset B], [\Box(A \supset B)], [A], [\neg\Diamond B]\rangle]$

8. $[\langle[\neg B], [\neg A, B], [\Box(A \supset B)], [A], [\neg\Diamond B]\rangle]$
9. $[\langle[\neg A], [\neg B], [\neg A, B], [\Box(A \supset B)], [A], [\neg\Diamond B]\rangle]$
10. $[\langle[\,], [\neg A], [\neg B], [\neg A, B], [\Box(A \supset B)], [A], [\neg\Diamond B]\rangle]$

Here 3) is from 2) and 5) is from 4) by the $S4$–π Rule; and 6) is from 5) and 7) is from 6) by the T–ν Rule.

It is easy to see that one can obtain the effect of an application of the $K4$–π Rule by applications of the $S4$–π Rule and the T–ν Rule. So completeness of the version with the $K4$–π Rule immediately gives completeness of both versions. As in Section 4, completeness can be shown using appropriate Model Existence Theorems ([8], [9]), or directly. Details are not difficult, and are omitted here.

A different approach to T, $K4$ and $S4$ is possible. In [11] we gave tableau systems for these logics directly, and also showed how they could be embedded into K, so that only a K system needed to be implemented. A similar thing is possible here. Once again we omit details.

Finally a word about $S5$. One way of dealing with the propositional version is to cheat. It is well-known that, in the propositional case, every formula is equivalent in $S5$ to a formula with no nested modal operators (a formula of *modal degree one*). There is an algorithm for doing such a conversion in [9] for instance. Also it is the case that propositional $S5$ and propositional $S4$ have the same valid formulas of modal degree one (again there is a proof in [9]). So to verify a formula in propositional $S5$, convert it into a formula of modal degree one and use an $S4$ theorem prover.

6 First Order Modal Logics

First order modal logics come in several varieties. Constant symbols can be rigid, naming the same thing in each world, or non-rigid (also known as flexible). Quantifier domains associated with possible worlds can be entirely arbitrary, or they can be the same from world to world, or they can meet a so-called *monotonicity* condition: if world Δ is accessible from world Γ then the domain associated with Γ is a subset of the one associated with Δ. The propositional resolution system we have been considering does not lend itself to 'constant domain' or 'arbitrary domain' logics, but only to those meeting the monotonicity condition. Likewise non-rigid symbols do not seem treatable. (The issue of non-rigid designators is an important one, about which a certain amount of confusion exists in the literature; see [13] for a fuller discussion.)

From now on we assume we have a first-order modal language formulated in the standard way. In Kripke models for this language we assume domains meet the monotonicity condition, and constant symbols are rigid. With these restrictions, we consider first-order versions of the propositional logics investigated earlier. We are informal about syntax matters. The main point is that we take both quantifiers as primitive, and we continue the system of uniform notation as in [22]. Quantified formulas and their negations are divided into *universals*, or γ formulas, and *existentials*, or δ formulas. For each such formula, and for each term t, a t *instance* is defined. These notions are given in Figure 3.

With modal logics one can not Skolemize ahead of time. Essentially this is because quantifiers and modal operators don't generally commute. The solution that we adopt for this problem is the

Figure 3: The γ and δ cases.

γ	$\gamma(t)$
$\forall(x)\Phi$	$\Phi\{x/t\}$
$\neg\exists(x)\Phi$	$\neg\Phi\{x/t\}$

δ	$\delta(t)$
$\neg\forall(x)\Phi$	$\neg\Phi\{x/t\}$
$\exists(x)\Phi$	$\Phi\{x/t\}$

same one we used in [10] and [11]: we intermingle Skolemizing with resolution. Roughly, we only carry out a part of the Skolemization process when we hit a quantifier in the course of a proof. Of course this is an option for classical logic too, but in the modal case it is forced on us. Now, the quantifier rules we want are simply these.

Quantifier Reduction Rules

Universal If a γ formula occurs in a clause C then a new clause C' may be *added*, where C' is like C except that in place of γ, C' contains $\gamma(x)$, where x is a previously unused free variable. Schematically,

$$\frac{[\gamma, A_1, \ldots, A_n]}{[\gamma(x), A_1, \ldots, A_n], [\gamma, A_1, \ldots, A_n]}$$

Existential An occurrence of a δ formula in a clause may be *replaced* by the formula $\delta(f(x_1, \ldots, x_n))$, where f is a previously unused function symbol (a Skolem function symbol) and $x_1, \ldots, x_n$ are all the free variables that have been introduced so far by the Universal Reduction Rule. Schematically,

$$\frac{\delta}{\delta(f(x_1, \ldots, x_n))}$$

Finally, the presence of free variables must be taken into account. The most generous way of doing this is to add a Substitution Rule. For a substitution σ we use the notation $B\sigma$, where B is a block, to denote the result of applying σ to every formula in B. We say σ is *free* for a formula Φ provided, for each variable x in the domain of σ, $x\sigma$ is free for x in Φ. Finally, we say σ is *free* for a block provided it is free for every formula in it.

Substitution Rule A block B may be replaced by a block $B\sigma$ provided σ is free for B.

Adding these rules to any of the propositional systems considered earlier gives a first order version for which soundness is not hard to establish. We remark again that Kripke models with varying (but monotonic) domains are appropriate here.

Of course the Substitution Rule is problematic when it comes to implementation, because the choice of substitution to apply is left open. In fact, the systems are complete if applications of the Substitution Rule are restricted to most general (free) substitutions that enable a Resolution Rule application, and thus unification can be used in the usual way, combined with Resolution. Note,

however, that substitutions must be applied to the entire block. Thus if we compute a most general unifier σ whose application will allow us to resolve two clauses in one clause list in a block, all other clause lists will be affected as well. This means that we can not try to produce closed clause lists from the various clause lists in a block in an independent way; communication is necessary.

Reduction Rule applications are analogous to the steps involved in the classical reduction to clause form. And we still have completeness if all applications of the Resolution Rule are restricted to the literal level. If one takes advantage of this, freeness of substitutions is guaranteed, because the only variables present at the literal level will be free variables introduced by the Universal Rule, and these were all new, hence most general unifiers can not interact badly with quantifiers.

Completeness, with the restrictions discussed above, can be shown using appropriate first order consistency properties [9] together with a straightforward generalization of the Lifting Lemma.

Example. The following is a proof in the first order $S4$ system of $\Diamond(\exists x)(P(x) \supset \Box(\forall y)P(y))$

1. $[\langle[\neg\Diamond(\exists x)(P(x) \supset \Box(\forall y)P(y))]\rangle]$
2. $[\langle[\neg(\exists x)(P(x) \supset \Box(\forall y)P(y))], [\neg\Diamond(\exists x)(P(x) \supset \Box(\forall y)P(y))]\rangle]$
3. $[\langle[\neg(P(v_1) \supset \Box(\forall y)P(y))], [\neg(\exists x)(P(x) \supset \Box(\forall y)P(y))], [\neg\Diamond(\exists x)(P(x) \supset \Box(\forall y)P(y))]\rangle]$
4. $[\langle[P(v_1)], [\neg\Box(\forall y)P(y))], [\neg(\exists x)(P(x) \supset \Box(\forall y)P(y))], [\neg\Diamond(\exists x)(P(x) \supset \Box(\forall y)P(y))]\rangle]$
5. $[\langle[\neg(\forall y)P(y)], [\neg\Diamond(\exists x)(P(x) \supset \Box(\forall y)P(y))]\rangle]$
6. $[\langle[\neg P(f(v_1))], [\neg\Diamond(\exists x)(P(x) \supset \Box(\forall y)P(y))]\rangle]$
7. $[\langle[\neg(\exists x)(P(x) \supset \Box(\forall y)P(y))], [\neg P(f(v_1))], [\neg\Diamond(\exists x)(P(x) \supset \Box(\forall y)P(y))]\rangle]$
8. $[\langle[\neg(P(v_2) \supset \Box(\forall y)P(y))], [\neg(\exists x)(P(x) \supset \Box(\forall y)P(y))], [\neg P(f(v_1))],$
   $[\neg\Diamond(\exists x)(P(x) \supset \Box(\forall y)P(y))]\rangle]$
9. $[\langle[P(v_2)], [\neg\Box(\forall y)P(y))], [\neg(\exists x)(P(x) \supset \Box(\forall y)P(y))], [\neg P(f(v_1))],$
   $[\neg\Diamond(\exists x)(P(x) \supset \Box(\forall y)P(y))]\rangle]$
10. $[\langle[\,], [P(f(v_1))], [\neg\Box(\forall y)P(y))], [\neg(\exists x)(P(x) \supset \Box(\forall y)P(y))], [\neg P(f(v_1))],$
    $[\neg\Diamond(\exists x)(P(x) \supset \Box(\forall y)P(y))]\rangle]$

In this 3) is from 2) by the Universal Rule; 6) is from 5) by the Existential Rule; 8) is from 7) by the Universal Rule; and 10) is from 9) by Substitution and Resolution.

The reader may wish to attempt a proof in the first-order system for K of

$$(\forall x)(\forall y)(\forall z)(\Box P(x,y) \vee \Box Q(y,z)) \supset ((\exists x)(\exists y)\neg\Diamond\neg P(x,y) \vee (\exists y)(\exists z)\neg\Diamond\neg Q(y,z))$$

which is a theorem, and also of the non-theorem

$$(\forall x)(\forall y)(\forall z)(\Box P(x,y) \vee \Box Q(y,z)) \supset ((\forall x)(\forall y)\neg\Diamond\neg P(x,y) \vee (\forall y)(\forall z)\neg\Diamond\neg Q(y,z)).$$

7 Conclusion

We have given, in detail, destructive resolution systems for four normal modal logics. Other normal logics, such as D are easily treatable this way. Small changes allow the treatment of several regular, but non-normal logics, such as C. Then interpretation into these logics allows the treatment of the quasi-regular logics $S2$ and $S3$. Finally one can also treat logics of knowledge, thought of as modal logics with many modal operators.

It is by design that the systems here bear a strong resemblance to the tableau based systems of [11]. There is a kind of duality between tableau systems and these resolution systems. Resolution has been extensively investigated in classical logic, while tableaux have been more or less neglected. Perhaps with the rich field of non-classical logics to work with, the relative merits and disadvantages of the two approaches can be better assessed.

References

[1] M. Abadi and Z. Manna, Non clausal temporal deduction, *Logics of Programs*, R. Parikh ed., Springer Lecture Notes in Computer Science, vol 173, pp 1–15 (1985).

[2] M. Abadi and Z. Manna, Modal theorem proving, 8CAD, LNCS, Springer-Verlag, pp 172–186 (1986).

[3] Y. Auffray and P. Enjalbert, Strategies for modal resolution, to appear.

[4] M. Davis and H. Putnam, A computing procedure for quantification theory, *JACM*, vol 7, pp 201–215 (1960).

[5] P. Enjalbert and L. Farinas, Modal resolution in clausal form, to appear in *Theoretical Computer Science*.

[6] L. Farinas del Cerro, Resolution modal logic, *Logique et analyse*, no 110, 111, pp 152–172 (1985).

[7] M. C. Fitting, Tableau methods of proof for modal logics, *Notre Dame Journal of Formal Logic*, vol 13, pp 237–247 (1972).

[8] M. C. Fitting, Model existence theorems for modal and intuitionistic logics, *Journal of Symbolic Logic*, vol 38, pp 613–627 (1973).

[9] M. C. Fitting, *Proof Methods for Modal and Intuitionistic Logics*, D. Reidel Publishing Co., Dordrecht (1983).

[10] M. C. Fitting, Resolution for intuitionistic logic, *Methodologies for Intelligent Systems*, Z. W. Ras and M. Zemankova eds., pp 400–407, North Holland Publishing Co., New York (1987).

[11] M. C. Fitting, First-order modal tableaux, *Journal of Automated Reasoning*, vol 4, pp 191–213 (1988).

[12] M. C. Fitting, *Logical Foundations of Automated Theorem Proving*, forthcoming, Springer-Verlag.

[13] M. C. Fitting, Modal logic should say more than it does, submitted for publication.

[14] C. Geissler and K. Konolige, A Resolution method for quantified modal logics of knowledge and belief, *Theoretical Aspects of Reasoning About Knowledge*, J. Y. Halpern ed., pp 309–324, Morgan Kaufmann Publishers, Los Altos, CA (1986).

[15] R. E. Ladner, The computational complexity of provability in systems of modal propositional logic, *SIAM J. Comput.*, vol 6, pp 467–480 (1977).

[16] D. W. Loveland, *Automated Theorem Proving: A Logical Basis*, North-Holland Publishing Co., Amsterdam (1978).

[17] G. E. Mints, Resolution calculi for modal logics, *Eesti NSV Tead. Akad. Toimetised Füüs-Mat.* vol 35, pp 279–290 (1986).

[18] G. E. Mints, review of "Some remarks on the possibility of extending resolution proof procedures to intuitionistic logic," by M. Cialdea, *Math. Reviews*, review 87j:03016, p 5407 (1987).

[19] G. E. Mints, Cutfree formalizations and resolution method for propositional modal logics, *Proceedings of the LMPS Congress*, Moscow (1987).

[20] G. E. Mints and E. Tyugu, The programming system PRIZ, *Journal of Symbolic Computation* (1987).

[21] H. J. Ohlbach, A Resolution Calculus for modal logics, *CADE-9*, pp 500–516 (1988).

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

[23] M. Y. Vardi, On the complexity of epistemic reasoning, *Fourth Annual Symposium on Logic in Computer Science (LICS)*, IEEE Computer Society, pp 243–252 (1989).

[24] L. Wallen, Matrix proof methods for modal logics, *Proc of the 10th ICALP* (1987).