

Acknowledgements

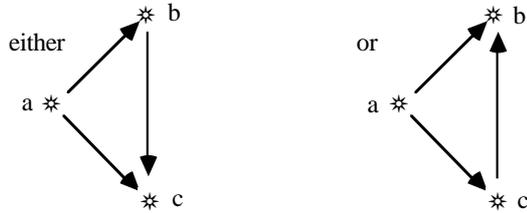
I would like to thank Andreas Nonnengart for many useful discussions and contributions, and Lincoln A. Wallen whose talk about his modal Matrix method, given in spring 1987 in Munich, inspired me to this work. Norbert Eisinger and Jörg Siekmann read drafts of this paper and gave many useful hints.

References

- AM86 M. Abadi, Z. Manna. *Modal Theorem Proving*.
In Proc. of CADE 86, pp. 172-189, 1986.
- Ch87 Chan, M. *The Recursive Resolution Method for Modal Logic*.
New Generation Computing, 5, 1987, pp 155-183.
- Ei85 N. Eisinger. What you always wanted to know about connection graphs.
Proc. of 8th Conference on Automated Deduction, Oxford, 1985.
- Fa85 L. Fariñas del Cerro. *Resolution modal logics*.
In Logics and Models of Concurrent Systems, (K.R. Apt, ed.), Springer 1985, pp 27-55.
- Fit72 M.C. Fitting. *Tableau methods of proof for modal logics*.
Notre Dame Journal of Formal Logic, XIII:237-247,1972.
- Fit83 M.C. Fitting. *Proof methods for modal and intuitionistic logics*.
Vol. 169 of Synthese Library, D. Reidel Publishing Company, Dordrecht, 1983.
- HC68 G.E.Hughes, M.J.Cresswell, *An Introduction to Modal Logics*.
Methuen &Co., London, 1968.
- Ko75 R. Kowalski: *A Proof Procedure Using Connection Graphs*.
J.ACM Vol. 22, No.4, 1975.
- Kr 59 S. Kripke. *A Completeness Theorem in Modal Logic*.
J. of Symbolic Logic, Vol 24, 1959, pp 1-14.
- Kr 63 S. Kripke. *Semantical analysis of modal logic I, normal propositional calculi*.
Zeitschrift für mathematische Logik und Grundlagen der Mathematik, Vol. 9, 1963, pp 67-96.
- Oh88 H.J. Ohlbach, *A Resolution Calculus for Modal Logics*.
Thesis, FB Informatik, Univ. of Kaiserslautern (forthcoming 1988).
- Ro65 Robinson, J.A. *A Machine Oriented Logic Based on the Resolution Principle*.
J.ACM Vol. 12, No. 1, pp 23-41, 1965.
- RW69 Robinson, G., Wos,L., *Paramodulation and theorem proving in first order theories with equality*.
Machine Intelligence 4, American Elsevier, New York, pp. 135-150, 1969.
- SS 85 M. Schmidt-Schauß. *A Many-Sorted Calculus with Polymorphic Functions Based on Resolution and Paramodulation*. Proc. of 9th IJCAI, Los Angeles, 1985, 1162-1168.
- Wal87 L.A.Wallen *Matrix proof methods for modal logics*.
In Proc. of 10th IJCAI, 1987.
- Wa87 C. Walther: *A Many-sorted Calculus Based on Resolution and Paramodulation*.
Research Notes in Artificial Intelligence, Pitman Ltd., London, M. Kaufmann Inc., Los Altos, 1987.

a world a two worlds b and c are accessible, then either b is accessible from c or vice versa:

Euclidian
accessibility
relation



The unification of the two world-paths [0abv] and [0ac] for instance is possible, provided c is accessible from b. The unifier must therefore be a tuple consisting of a substitution and the negation of the condition, i.e. the relations that must hold if the unification was wrong. In our example it could look like $(\{[bv] \mapsto c\}, R(c,b))$. The resolution operation must insert these literals as a residue into the resolvent, similar to the residue in non-serial interpretations. Furthermore resolution between such literals $R(s,t)$ and literals containing partial world-paths [t s] must be allowed.

Accessibility Relations with finite R-chains.

One of the models for the modal system G that is characterized by Löb's axioms $\Box(\Box \mathcal{F} \Rightarrow \mathcal{F}) \Rightarrow \Box \mathcal{F}$ has a transitive accessibility relation where all chains $R(a_1, a_2), R(a_2, a_3), \dots$ terminate. In order to demonstrate where the methods developed in this work fail for the system G, let us try to prove Löb's axiom for the predicate P assuming only transitivity of R. The P-logic clause form for the negated theorem $\neg \Box(\Box P \Rightarrow P) \Rightarrow \Box P$ is:

$$C1: \neg P[0va], P[0v]$$

$$C2: \neg P[0b]$$

Resolution between C2 and C1,2 produces $\neg P[0ba]$ which is not very helpful. The second possibility for resolution is the self resolution of C1 with a unifier $\{v \mapsto [v'a]\}$ giving the resolvent $\neg P[0v'aa], P[0v']$. Now there is another self resolution possibility giving $\neg P[0v''aaaa], P[0v'']$ and a third time self resolution becomes possible etc: C1 carries the potential to generate a world-path with a sequence of infinitely many non-variable CW-terms, and that contradicts the assumption that there are no infinite R-chains. Simply eliminating the first literal and generating $P[0v]$ as a new clause would be a sound deduction in this example. The new clause could then be resolved with C2 giving the desired refutation. The potential to generate infinitely long world-paths with C1 has been indicated by a weak unifier (without renaming) $\{v \mapsto [va]\}$ that puts a non-variable term behind the substituted variable itself. Adding a rule for detecting these situations and removing literals from clauses that can produce infinitely long world-paths might lead to a feasible calculus for G.

paramodulation operation with an equation whose two sides have different sorts does not increase the sort of the paramodulated term. Adapting these ideas to P-logic should be no problem when the sort structure and the sort declarations for the function symbols do not depend on the modal context. This is the case in most applications where only fixed sorts like “Integer”, “Real” etc. occur.

Varying-Domain Modal Logics

In varying-domain interpretations there is no universal domain, but each world has its own domain which may or may not intersect with the domain of other worlds. That means universally quantified domain variables depend on the modal context. The idea is now to modify the translation function Π such that the W -term that characterizes the modal context is attached to the domain variables as well. Unification of such a world-depending D -variable $x[p]$ with a D -term $f([q], t_1, \dots, t_n)$ is only possible when the world-paths p and q are unifiable. To demonstrate this, let us try to prove the Barcan formula $\forall x \Box Px \Rightarrow \Box \forall x Px$ which does not hold in varying-domain models. If the proof fails, we have some evidence that the idea is sufficient. The P-logic clause form of the negated Barcan formula $\forall x \Box Px \wedge \Diamond \exists x \neg Px$ is:

$$C1: P([0u] x[0]) \quad C2: \neg P([0a] f[0a])$$

In fact, the two world-paths $[0]$ and $[0a]$ of the variable x and the symbol f are not unifiable and no refutation is possible. In constant-domain interpretations on the other hand, where x has no world-path, there is the unifier $\{u \mapsto a, x \mapsto f[0a]\}$.

Multiple Operator Modal Logics

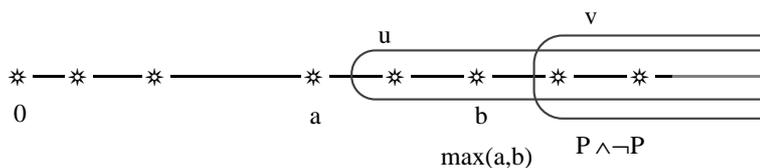
In the most simple multiple operator modal logics the two operators \Box and \Diamond are just multiplied and each copy is associated with its own accessibility relation. In such a logic we can index \Box and \Diamond with an integer and transfer this index to the P-logic syntax to obtain indexed W -variables and W -valued function symbols. The unification of a W -variable with index i and a partial world-path $[p_1 \dots p_n]$ with indices i_1, \dots, i_n is then possible only when all indices equal i . Even operators indexed with a set of integers should be easy to handle. If the semantics of an operator $\Box_{\{i, \dots, k\}}$ is disjunctive, i.e. $\Box_{\{i, \dots, k\}} \Leftrightarrow \Box_i \vee \dots \vee \Box_k$ then the unification of the corresponding CW-terms with index sets s_I and t_K must compute just the intersection $I \cap K$. If the semantics is conjunctive, i.e. $\Box_{\{i, \dots, k\}} \Leftrightarrow \Box_i \wedge \dots \wedge \Box_k$ then two non-variable CW-terms are unifiable only when their index sets are equal and a W -variable is unifiable with a CW-term only when the index set of the variable is a subset of the index set of the CW-term.

Other Properties of the Accessibility Relation

Let us come back to modal logics with one pair of operators \Box and \Diamond and consider some accessibility relations with properties other than reflexivity, symmetry and transitivity.

Linear Accessibility Relations.

Linear means that there is just one sequence of worlds. The interesting case, where the interpretation of the two modal operators is not identical is when the accessibility relation R is transitive, i.e. a total ordering. In this case for two given worlds it can always be determined which one is farther away from the initial world. The consequence is that for example a formula like $\Diamond \Box P \wedge \Diamond \Box \neg P$ is unsatisfiable when in addition R is serial. The reason is that for the two worlds denoted by the two \Diamond -operators, all worlds “behind” that one which is farthest away from the initial world, are also accessible from the other world. In other words there is no linear and serial interpretation where the intersection of the worlds denoted by the two \Box -operators is empty, and the formula requires P and $\neg P$ to hold in these worlds. In order to find this contradiction in the P-logic version of the formula: $\forall u P[0au] \wedge \forall v \neg P[0bv]$, we must be able to unify $[au]$ and $[bv]$. A plausible unifier could be $\{[au] \mapsto [\max(a,b)w], [bv] \mapsto [\max(a,b)w]\}$ where $\max(a,b) = a$ if $R(b,a)$ holds, otherwise $\max(a,b) = b$. The figure below shows an interpretation falsifying the formula.



Euclidian Accessibility Relations.

The accessibility relation of the modal system S 4.3 is reflexive, transitive and has the additional property that, if from

develop specialized theorem provers for modal logics. Only a slight modification of an existing predicate logic resolution based theorem prover is sufficient, in other words most of the sophisticated implementation and search control techniques, for instance the connection graph idea [Ko75, Ei85], which have been developed for predicate logic can immediately be applied to modal logics as well. This is an indirect advantage which, however, should not be underestimated: almost 25 years of experience with the resolution principle are now available to modal logic theorem proving as well.

What else can be done?

My hope is that the basic ideas presented in this work are powerful enough to open the door to efficient theorem proving in a much larger class of non-standard logics than the relatively simple modal systems I have examined so far. Let me therefore sketch some ideas for further work in this area.

Resolution in Modal Logics with Non-Serial Accessibility Relations.

A serious complication arises in non-serial models where a world may have no accessible world at all. A formula $\Box P$ for example may become true in such an interpretation, not because P is true, but because the quantification “for every accessible world” is empty. This phenomenon has two consequences for formulae in the P-logic translation:

1. A formula $\forall u P[0u]$ must not be straightforwardly instantiated with a non-variable term t because a world-path $[0t]$ in a formula $P[0t]$ requires the existence of an accessible world.
2. The two literals $P[0u]$ and $\neg P[0u]$ in a formula $\forall u P[0u] \wedge \neg P[0u]$ are not necessarily complementary because the quantification may be empty.

The introduction of an artificial predicate “End(w)” which is true in an interpretation \mathcal{X} iff $\mathcal{X}(w)$ is the last world, i.e. $\mathcal{X}(w)$ exists and there is no further world accessible from $\mathcal{X}(w)$, provides a solution to both problems:

1. The instantiation operation can now introduce a condition expressing its admissibility. For example $P[0u]$ instantiated with $\{u \mapsto a\}$ yields $\text{End}[0] \vee P[0a]$, i.e. either 0 is the last world or P holds in the world denoted by $[0a]$. The resolution operation therefore inserts such “End-literals” as a residue into the resolvent.
2. Further End-literals are to be inserted into the resolvent in order to handle the case that the resolution literals are not contradictory. For example the resolvent of $P[0u] \wedge \neg P[0u]$ would be $\text{End}[0]$. This literal can for instance be resolved with a literal $Q[0a]$ where a is not a variable because $\text{End}[0]$ expresses that 0 is the last world whereas $[0a]$ postulates the existence of an accessible world a .

A sound and complete calculus based on this idea is given in [Oh88]. The subjects mentioned below, however, are still open research problems.

Equality Reasoning in Modal Logics

Consider the formula $\mathcal{F}: a = b \wedge \Box P(a)$. Since the second occurrence of a is in the scope of the \Box -operator and may therefore be interpreted differently to the first occurrence, it is not possible to replace a by b and to deduce $\Box P(b)$. Thus, an unrestricted application of a replacement operation like the paramodulation rule [RW69] is not sound in the modal logic syntax. In P-logic syntax, the modal context is available at each term and can be used to influence a deduction operation. The translated formula $\Pi(\mathcal{F}): a[0] = b[0] \wedge \forall u P([0u] a[0u])$ for instance can safely be paramodulated when the accessibility relation is reflexive, the unifier for $a[0]$ and $a[0u]$ is $\{u \mapsto []\}$, and the paramodulant is $P([0] b[0])$. (We assume the equality predicate to be rigid!) Thus, equality reasoning by paramodulation should be no problem in P-logic. The paramodulation rule need not be changed, just the accessibility relation dependent unification algorithms must be applied for unifying one side of an equation, which is always a D-term, with the subterm of the literal to be paramodulated. More work, however, needs to be done for adapting term rewriting and Knuth-Bendix completion techniques to P-logic. They must be able to handle more than one most general unifier and the term orderings must take the world-path strings into account.

Many-Sorted Modal Logics

Resolution and paramodulation calculi for sorted first order predicate logic have been published for instance in [Wa87] and [SS85]. It has been shown that only two slight modifications of the unification algorithm and one modification of the paramodulation rule are necessary for handling hierarchical sort structures: A variable x of sort $S1$ and a variable y of sort $S2$ can only be unified when there is a common subsort of $S1$ and $S2$. A variable x of sort $S1$ can only be unified with a term t of sort $S2$ if $S2 = S1$ or $S2$ is a subsort of $S1$. The paramodulation rule must take care that a

11. The Resolution Rule

There is no significant difference to the resolution rule for predicate logic.

Definition: Let $C = Pt_1 \vee \dots \vee Pt_n \vee C'$ and $D = \neg Psl_1 \vee \dots \vee \neg Psl_m \vee D'$ be two clauses with no variables in common and let σ be a most general prefix-preserving and R-compatible unifier for the termlists t_1, \dots, t_n and sl_1, \dots, sl_m , i.e. $t_1\sigma = \dots = t_n\sigma = sl_1\sigma = \dots = sl_m\sigma$. Then the clause $(C' \vee D')\sigma$ is called a *resolvent* of C and D. ■

Theorem The resolution rule is sound and complete.

The completeness proof follows the ideas of the completeness proof of the resolution rule for predicate logic using term interpretations (an analogy to Herbrand interpretations) and semantic trees. ■

A **final example** shall illustrate the whole procedure. The example proves that in the modal system D (serial accessibility relation with no special properties) Löb's Axioms $\Box(\Box G \Rightarrow G) \Rightarrow \Box G$ imply the formula $\Box P \Rightarrow \Box \Box P$ that characterizes transitive accessibility relations. Let $G := P \wedge \Box P$. The theorem to be proved is

$$\mathcal{F} := (\Box(\Box(P \wedge \Box P) \Rightarrow (P \wedge \Box P)) \Rightarrow \Box(P \wedge \Box P)) \Rightarrow (\Box P \Rightarrow \Box \Box P).$$

Step 1: Negation of \mathcal{F} yields:

$$\neg((\Box(\Box(P \wedge \Box P) \Rightarrow (P \wedge \Box P)) \Rightarrow \Box(P \wedge \Box P)) \Rightarrow (\Box P \Rightarrow \Box \Box P)).$$

Step 2: The negation normal form of $\neg \mathcal{F}$ is:

$$(\Diamond(\Box(P \wedge \Box P) \wedge (\neg P \vee \Diamond \neg P)) \vee \Box(P \wedge \Box P)) \wedge (\Box P \wedge \Diamond \neg \Box P)$$

Step 3: Translation into P-logic syntax:

$$((\forall u (P[0au] \wedge \forall v P[0auv]) \wedge (\neg P[0a] \vee \neg P[0ab])) \vee \forall w (P[0w] \wedge \forall x P[0wx])) \wedge (\forall y P[0y] \wedge P[0cd])$$

Step 4: Conjunctive normal form:

$$\begin{aligned} & \forall u (P[0au] \wedge \forall v P[0auv]) \wedge \\ & (\neg P[0a] \vee \neg P[0ab] \vee \forall w (P[0w])) \wedge \\ & (\neg P[0a] \vee \neg P[0ab] \vee \forall x P[0wx]) \wedge \\ & \wedge \forall y P[0y] \wedge P[0cd] \end{aligned}$$

Eliminating the universal quantifiers gives the usual clause notation:

C1: P[0au]	C2: P[0auv]
C3: $\neg P[0a], \neg P[0ab], P[0w]$	C4: $\neg P[0a], \neg P[0ab], P[0wx]$
C5: P[0y]	C6: P[0cd]

Step 5: A resolution refutation:

C4,1 & C5	→	R1: $\neg P[0ab], P[0w'x']$	(unifier = $\{y \mapsto a\}$)
R1,1 & C6	→	R2: $\neg P[0ab]$	(unifier = $\{w' \mapsto c, x' \mapsto d\}$)
R2 & C1	→	R3: empty	(unifier = $\{u \mapsto b\}$)

Conclusion

A clause based resolution calculus has been developed for several first order modal logics. The two most significant advantages of this calculus are:

- Instantiation and inference across modal operators can be controlled by a uniform and deterministic unification algorithm. Thus the large search space generated by the usual instantiation rules and operator shifting rules in tableau based systems is eliminated.
- The method is absolutely compatible with the paradigm of the standard resolution principle, i.e. there is no need to

Function Unify-world-paths-equivalence (s, t, R)

Input: Two world-paths s and t. $R = \{\text{reflexive, symmetric, transitive}\}$.

Output: A complete set of unifiers for s and t.

Return Case (s, t) = ([0], [0]) then $\{\emptyset\}$
 ([0], [0w]) or ([0w], [0]) where w is a variable then $\{\{w \mapsto []\}\}$
 ([0], [0r]) or ([0r], [0]) then \emptyset
 ([0r], [0q]) then Unify-terms (r, q, R).

Function Unify-instantiated-wps (Ξ , s, t, R)

Input: Ξ is a set of substitutions, s and t are two world-paths.

Output: A complete set of unifiers for s and t which are smaller than some element of Ξ .

Return $\bigcup_{\xi \in \Xi} \{\xi\theta\}_{\text{Var}(s,t)} \mid \theta \in \text{Unify-world-paths}(s\xi, t\xi, R)$.

Function Unify-collapse(t, R)

Input: $t = [t_1, \dots, t_n]$ is a world-path. $R \in \{\{\text{reflexive}\}, \{\text{symmetric}\}, \{\text{reflexive, symmetric}\}\}$.

Output: A complete set of unifiers which collapse t into [].

Case $n = 0$ **Return** $\{\emptyset\}$.

$n > 0$ **Let** $\Lambda := \emptyset$

If R is reflexive and t_1 is a variable then

$\tau := \{t_1 \mapsto []\}; \quad \Lambda := \Lambda \cup \{\tau\theta \mid \theta \in \text{Unify-collapse}([t_2, \dots, t_n], \tau, R)\}$

If R is symmetric and $t_1 \neq 0$ and $n > 1$ then

For $i = 2, \dots, n$

If t_i is a variable then

$\tau := \{t_i \mapsto [t_i^{-1}]\}$

$\Xi := \{\tau\theta \mid \theta \in \text{Unify-collapse}([t_2, \dots, t_{i-1}], R)\}$

$\Lambda := \Lambda \cup \bigcup_{\xi \in \Xi} \{\xi\theta \mid \theta \in \text{Unify-collapse}([t_{i+1}, \dots, t_n], \xi, R)\}$

Return Λ .

Function Unify-prefix (s_1, t, R)

Input: A CW-term s_1 , a world-path $t = [t_1 \dots t_n]$. R is transitive (and reflexive).

Output: If either s_1 is a variable or $n = 1$: A complete set of unifiers for $[s_1]$ and $[t_1 \dots t_n]$.

Return **If** $n = 1$ then Unify-terms (s_1, t_1, R).

elseif s_1 is a variable and $s_1 \notin t$ and either $n > 0$ or R is reflexive

then $\{\{s_1 \mapsto t\}\}$.

otherwise \emptyset .

Function Unify-split (s, t, i, R)

Input: Two world-paths $s = [s_1 \dots s_n]$ and $t = [t_1 \dots t_m]$, a positive integer and a transitive (and possibly reflexive) accessibility relation R.

Output: A complete set of unifiers for s and t.

If $i > 1$ and t_i is a variable and $s_1 \notin [t_1 \dots t_i]$ then

Let $\xi := \{s_1 \mapsto [t_1 \dots t_{i-1}u], t_i \mapsto [u v]\}$ where u and v are new variables

Return $\{\theta\}_{\text{Var}(s,t)} \mid \theta \in \text{Unify-instantiated-wps}(\{\xi\}, [s_2 \dots s_n], [v t_{i+1} \dots t_m], R)$

else **Return** \emptyset .

Theorem The unification algorithm terminates and it is sound and complete, i.e. it computes a complete set of prefix-preserving idempotent most general unifiers. ■

Function Unify-terms (s, t, R)

Input: s and t are either empty lists or two prefix-stable terms or atoms.

Output: A complete set of idempotent and prefix-preserving unifiers for s and t.

If s = t then Return { \emptyset } (\emptyset is the empty substitution, i.e. the identity.)If s is a variable then Return If s \in t then \emptyset else {{s \mapsto t}}If t is a variable then Return If t \in s then \emptyset else {{t \mapsto s}}If Var(s, t) = \emptyset or s = () or t = () or topsymbol(s) \neq topsymbol(t) then Return \emptyset .If s and t are CW-terms then Return Unify-termlists(arguments(s), arguments(t), R)Let s =: f(v, s₁, ..., s_n) and t =: f(w, t₁, ..., t_n)Let Ξ := Unify-world-paths (v, w, R)Return $\bigcup_{\xi \in \Xi} \{ \xi \theta \mid \theta \in \text{Unify-termlists}((s_1, \dots, s_n)^\xi, (t_1, \dots, t_n)^\xi, R) \}$ **Function** Unify-termlists (s, t, R)Input: Two prefix-stable termlists s =: (s₁...s_n) and t =: (t₁...t_m).

Output: A complete set of prefix-preserving idempotent unifiers for s and t.

If s = t then Return { \emptyset }.Let Ξ := Unify-terms (s₁, t₁, R)Return $\bigcup_{\xi \in \Xi} \{ (\xi \theta)_{\text{Var}(s,t)} \mid \theta \in \text{Unify-termlists}((s_2 \dots s_n)^\xi, (t_2 \dots t_m)^\xi, R) \}$.**Function** Unify-world-paths (s, t, R)

Input: Two world-paths.

Output: A complete set unifiers for s and t.

If s = t then Return { \emptyset }.If s = () or t = () then Return \emptyset .Return Case R denotes an accessibility relation which iswithout special properties then Unify-termlists (s, t, R)reflexive and (or) symmetric then Unify-world-paths-reflexive-or-symmetric (s, t, R)transitive then Unify-world-paths-transitive (s, t, R)reflexive and transitive then Unify-world-paths-transitive (s, t, R)reflexive, symmetric and transitive then Unify-world-paths-equivalence (s, t, R).**Function** Unify-world-paths-reflexive-or-symmetric (s, t, R)Input: Two world-paths s =: [s₁ ... s_n] and t =: [t₁ ... t_m].R \in { {reflexive}, {symmetric}, {reflexive, symmetric} }.

Output: A complete set of unifiers for s and t.

Let Λ := Unify-instantiated-wps (Unify-terms (s₁, t₁, R), [s₂...s_n], [t₂...t_m], R)If R is reflexive only then n' := 1, m' := 1 else n' := n, m' := m.For i = 1, ..., n' Λ := $\Lambda \cup$ Unify-instantiated-wps (Unify-collapse ([s₁...s_i], R), [s_{i+1}...s_n], t, R).For i = 1, ..., m' Λ := $\Lambda \cup$ Unify-instantiated-wps (Unify-collapse ([t₁...t_i], R), s, [t_{i+1}...t_m], R).Return Λ .**Function** Unify-world-paths-transitive (s, t, R)Input: Two world-paths s =: [s₁ ... s_n] and t =: [t₁ ... t_m]. R \in {transitive}, {reflexive, transitive}.

Output: A complete set of unifiers for s and t.

Let Λ := \emptyset For i = 0, ..., m (i = 0 is the collapsing case when R is reflexive.)Let Ξ := Unify-prefix (s₁, [t₁...t_i], R) Λ := $\Lambda \cup$ Unify-instantiated-wps (Ξ , [s₂...s_n], [t_{i+1}...t_m], R). Λ := $\Lambda \cup$ Unify-split (s, t, i, R).Repeat the For loop with s and t exchanged.Return { $\lambda_{\text{Var}(s,t)} \mid \lambda \in \Lambda$ }.

function whose inverse exists in symmetric interpretations. For example the two world-paths $[0 \ v \ w]$ and $[0]$ are unifiable with a unifier $\{w \mapsto v^{-1}\}$. The unification algorithm must consider all possibilities to collapse a W-variable w and its predecessor t in the world-path by the substitution component $w \mapsto t^{-1}$ to the empty path $[]$ and to unify the CW-terms in the reduced world-paths pairwise. Since there are only finitely many variables to be collapsed, there are *at most finitely many most general unifiers* for each unification problem.

The symmetric case is the first case where the prefix-stability of terms can be exploited: When a W-variable w and its predecessor t in a world-path have been collapsed with the substitution component $w \mapsto t^{-1}$, we know that in all other terms occurring in the current clause set, t is the predecessor of w . The application of $w \mapsto t^{-1}$ to an arbitrary term containing w in the clause set will therefore collapse $[t \ w]$ to $[]$. No inverse CW-term will ever occur in an instantiated term.

Unification when the accessibility relation is reflexive and symmetric only.

The two basic ideas for reflexivity and symmetry can simply be joined. The unification algorithm must consider all possibilities to remove W-variables w by the substitution component $w \mapsto []$ and to collapse a W-variable w and its predecessor t in the world-path by the substitution component $w \mapsto t^{-1}$ into the empty path $[]$ and to unify the CW-terms in the reduced world-paths pairwise. Since there are only finitely many variables to be collapsed or removed, there are *at most finitely many most general unifiers* for each unification problem.

For example the two world-paths $[0 \ a \ u \ v]$ and $[0 \ w]$ are unifiable with two independent unifiers $\{u \mapsto a^{-1}, v \mapsto w\}$ and $\{v \mapsto u^{-1}, w \mapsto a\}$.

Unification when the accessibility relation is transitive only.

R-compatible substitutions may substitute arbitrary partial world-paths for a W-variable. A unifier for the two world-paths $[0 \ v \ c \ d]$ and $[0 \ a \ b \ w \ d]$ is $\{v \mapsto [a \ b], w \mapsto c\}$, but the substitution $\{v \mapsto [a \ b \ w'], w \mapsto [w' \ c]\}$ with a new W-variable w' is also a unifier. Thus, we must consider variable splitting like in the unification algorithm for associative function symbols, which is in general infinitary. Fortunately it turns out that the toplevel linearity of the world-paths is sufficient to *keep the unification finitary*.

The unification algorithm for two world-paths $s = [s_1 \dots s_n]$ and $t = [t_1 \dots t_m]$ works from left to right.

Roughly speaking it consists of the three main steps:

1. Unify the term s_1 with t_1 and call the world-path unification algorithm recursively for $[s_2 \dots s_n]$ and $[t_2 \dots t_m]$.
- 2.a When s_1 is a W-variable then for $i = 2, \dots, m$ create the substitution component $s_1 \mapsto [t_1 \dots t_i]$ and call the world-path unification algorithm recursively for $[s_2 \dots s_n]$ and $[t_{i+1} \dots t_m]$.
- 2.b When t_1 is a W-variable, then split t_1 into the two new W-variables $[u \ v]$, create the substitution component $\{s_1 \mapsto [t_1 \dots t_{i-1} \ u], t_1 \mapsto [u \ v]\}$ and call the world-path unification algorithm again for $[s_2 \dots s_n]$ and $[v \ t_{i+1} \dots t_m]$.

Unification when the accessibility relation is reflexive and transitive only.

The ideas for the reflexive case and transitive case can be joined without further problems. The algorithm for the transitive case must simply be augmented with a step that removes W-variables w with a substitution component $w \mapsto []$. There are still *at most finitely many most general unifiers* for each unification problem.

Unification when the accessibility relation is an equivalence relation (modal logic S5)

World-paths for S5 interpretations in normal form of modal degree one consist of at most two CW-terms, i.e. they look like $[0]$ or $[0 \ t]$. Two world-paths $[0]$ and $[0 \ t]$ can only be unified when t is a variable, the unifier is $t \mapsto []$. Two world-paths $[0 \ s]$ and $[0 \ t]$ can be unified when s and t are unifiable. Therefore there is *at most one most general unifier* for each unification problem.

The Unification Algorithm

Only one algorithm is defined that takes the accessibility relation *type* R as an additional parameter and branches internally to the R -depending algorithm for world-paths. (R may be just a list of key words like 'reflexive', 'symmetric' and 'transitive'.) The main control loop of the algorithm is similar to the Robinson algorithm for first-order terms. ($s \in t$ is true if s is a subterm of t . $\text{Var}(s_1 \dots s_n)$ denotes the set of variables occurring in s_1, \dots, s_n .)

and only transitivity guarantees that Σ' is directly accessible from Σ . In non-transitive interpretations, substitution components like $v \mapsto \lambda(w) g(w)$ with only one nested function in the codomain can be interpreted as W-variable assignments. Therefore we introduce the notion of *R-compatible* substitutions with a syntactic condition that guarantees that R-compatible substitutions can be interpreted as W-variable assignments.

Definition: Given an accessibility relation R , a substitution σ is called *R-compatible*

iff either R is transitive (there is no restriction for transitive interpretations) or

every substitution component for W-variables has the form:

$$\begin{array}{ll} u \mapsto \lambda(w) g(w, s_1, \dots, s_k) & \text{or} \\ u \mapsto \lambda(w) g^{-1}(w, s_1, \dots, s_k) & \text{in case } R \text{ is symmetric} \quad \text{or} \\ u \mapsto \lambda(w) w & \text{in case } R \text{ is reflexive.} \end{array} \quad \blacksquare$$

9. Prefix-Stability - An Invariant on the Structure of Terms.

Terms of a translated modal logic formula are “prefix-stable”, i.e. they have the property that all occurrences of a W-variable have identical subterms. A term like $f(w(0), g(w(v(0))))$ for example never occurs. This property prevents the unification algorithms for transitive interpretations from becoming infinitary, and should be preserved also for resolvents. Unifiers must therefore be “prefix-preserving”, i.e. the unified instances must again be prefix-stable.

An important property of prefix-stable terms is that they have no double occurrences of a W-variable at the toplevel of a world-path (toplevel linearity). Furthermore a W-variable in a prefix-stable term cannot occur in its own prefix (prefix-linearity). Nevertheless it is possible, that a W-variable occurs a second time behind its first occurrence in a prefix-stable world-path. An example is: $[0 w g(a[0 w v])]$.

10. Unification

The unification algorithms for P-logic terms, we are going to define work on the world-path syntax. The unification of world-paths that must produce R-compatible substitutions is therefore the only difference to unification of first order predicate logic terms.

Unification when the accessibility relation has no special properties.

R-compatible substitutions are allowed to substitute a partial world-path with *exactly* one CW-term for a W-variable.

World-paths like $[0 v a]$ and $[0 b w]$ are therefore unifiable with a unifier $\{v \mapsto b, w \mapsto a\}$, whereas the two terms $[0 v a]$ and $[0 b u w]$ would require a non-R-compatible substitution $\{v \mapsto [b c], w \mapsto a\}$. They are not unifiable.

In general two world-paths are unifiable when they have equal length and the CW-terms are pairwise unifiable with compatible unifiers. Thus, the world-paths can be treated like ordinary terms and with the exception that the argument lists may be of different length, there is no difference from unification of first-order predicate logic terms. *There is at most one most general unifier* that is unique up to renaming for every unification problem.

Unification when the accessibility relation is reflexive only.

The substitution component $w \mapsto []$ represents the assignment of the identity mapping to a W-variable. It is R-compatible because in reflexive interpretations a world is accessible from itself. Therefore R-compatible substitutions are allowed to substitute a partial world-path with *at most* one CW-term for a W-variable. The substitution components $w \mapsto []$ remove a variable completely from a world-path such that the world-paths $[0 v a]$ and $[0 b u w]$ are unifiable with the two independent unifiers $\{v \mapsto b, u \mapsto [], w \mapsto a\}$ and $\{v \mapsto b, u \mapsto a, w \mapsto []\}$. The unification algorithm must consider all possibilities to remove W-variables w by the substitution component $w \mapsto []$ and to unify the CW-terms in the reduced world-paths pairwise. Since there are only finitely many variables to be removed, there are *at most finitely many most general unifiers* for each unification problem.

Unification when the accessibility relation is symmetric only.

In symmetric interpretations, each W-valued function symbol has an associated inverse function. A substitution component $w \mapsto a^{-1}$ is therefore suitable for collapsing the partial world-path $[a w]$ into $[a a^{-1}] = []$. R-compatible substitutions are allowed to substitute a partial world-path with *exactly* one CW-term or an “inverse” CW-term for a W-variable. The “inverse” v^{-1} of a W-variable is also allowed, because the interpretation of a W-variable is also a

Examples of the different syntactic versions:

Modal Logic	P-logic, W-term Syntax	World-path Syntax
$\Box P$	$\forall w P(w(0))$	$\forall w P[0 w]$
$\Diamond P$	$P(g(0))$	$P[0 g]$
$\forall x \Diamond Q(x, a)$	$\forall x Q(h(0, x), x, a(h(0, x)))$	$Q([0 h(x)], x, a[0 h(x)])$
$\Box \forall x (R(x) \wedge \Box \exists y \Diamond R(y))$	$\forall w \forall x (R(w(0), x) \wedge \forall v R(k(v(w(0)), x), r(v(w(0)), x)))$	$\forall w \forall x (R([0 w], x) \wedge \forall v R([0 w v k(x)], r[0 w v]))$
$\Diamond \Box P$	$\forall w P(h(w(g(0))))$	$\forall w P[0 g w h]$

8. Substitutions

One of the most important notions is that of a substitution as a mapping from terms to terms. Since we deal with functional variables, we must use the notions of the λ -calculus for correlating syntactic objects with functions.

Definition: (Substitutions)

- ▶ A *substitution* (in W-term syntax) is a finite set $\{x_1 \mapsto t_1, \dots, x_k \mapsto t_k\}$ of assignments of D-terms to D-variables and λ -expressions $\lambda(w) g_n(g_{n-1}(\dots g_1(w, s_{1,1}, \dots, s_{1,k_1}), s_{2,1}, \dots, s_{2,k_2}), \dots)$ to W-variables where the g_i are $(1, k_i)$ -place W-valued function symbols and the $s_{i,j}$ are D-terms which do not contain w as a subterm. Here n may be 0 in which case $\lambda(w) w$ is just the identity mapping.
- ▶ A substitution component $v \mapsto \lambda(w) g_n(g_{n-1}(\dots g_1(w, s_{1,1}, \dots, s_{1,k_1}), s_{2,1}, \dots, s_{2,k_2}), \dots)$ for a W-variable has in world-path syntax the following structure: $v \mapsto [g_1(s_{1,1}, \dots, s_{1,k_1}) \dots g_n(s_{n,1}, \dots, s_{n,k_n})]$. $v \mapsto []$ is the representation of the identity $\lambda(w) w$.
- ▶ Substitutions can be turned into mappings from terms to terms, literals to literals and clauses to clauses using the inductive definition of terms and the β -reduction rule of the λ -calculus such that the following equation for all $(1, n)$ -place function and predicate symbols f and terms w and t_i hold: $f(w, t_1, \dots, t_n)\sigma = f(w\sigma, t_1\sigma, \dots, t_n\sigma)$. An application of a substitution $\sigma = \{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$ to a term t has the intuitive meaning that all occurrences of D-variables x_i are simultaneously replaced by t_i and all occurrences of W-variables x_j in a subterm $x_j(s)$ (in W-term syntax) are simultaneously replaced by $(\lambda(w) g(\dots(w, \dots)\dots)(s))$ and immediately reduced with the β -reduction rule to $g(\dots(s, \dots)\dots)$. Furthermore, an application of a substitution with an inverse function in symmetric interpretations like $\lambda(w) g^{-1}(w)$ to a term $g(s)$ immediately reduces the term to $g^{-1}(g(s)) = s$. Thus, the rewrite rule $g^{-1}(g(v, a_1, \dots, a_n), a_1, \dots, a_n) \rightarrow v$ will implicitly be applied whenever it is possible.
- ▶ The application of a substitution $\{x_i \mapsto [g_1 \dots g_n]\}$ in world-path syntax splices the partial world-path $[g_1 \dots g_n]$ into each world-path at the place of an occurrence of x_i , i.e. $[0 \dots a x_i b \dots] \{x_i \mapsto [g_1 \dots g_n]\} = [0 \dots a g_1 \dots g_n b \dots]$. The corresponding rewrite rule that reduces world-paths with inverse functions in symmetric interpretations is: $[g(a_1, \dots, a_n) g^{-1}(a_1, \dots, a_n)] \rightarrow []$.
- ▶ The composition $\sigma\tau$ of two substitutions σ and τ is denoted as: $t(\sigma\tau) = (t\sigma)\tau$ for a term t .
- ▶ σ_V is the restriction of a substitution σ to a set V of variables.
- ▶ A substitution σ is *idempotent* iff $\sigma\sigma = \sigma$.

Examples for substitutions:

W-term syntax:	$\sigma = \{x \mapsto f(y), v \mapsto \lambda(w) g(u(h(w, a(0))))\}$	(u is a W-variable)
World-path syntax:	$\sigma = \{x \mapsto f(y), v \mapsto [h(a[0]) u g]\}$.	
$f(k(v(l(0))), x)\sigma$	$= f(k(\lambda(w) g(u(h(w, a(0)))) (l(0)), f(y))$	(W-term syntax)
	$= f(k(g(u(h(l(0), a(0)))) (l(0)), f(y))$	(β -reduction)
$f([0 l v k], x)\sigma$	$= f([0 l h(a[0]) u g k], f(y))$	(World-path syntax)

Substitutions are usually the syntactic equivalents to the variable assignments that are used in the definition for the semantics of variables. Therefore we must find syntactic conditions for substitutions that reflect the restrictions imposed on the semantic objects assigned to variables. The definition of W-variable assignments in P-interpretations has the restriction, that only functions which map worlds to accessible worlds can be assigned to a W-variable. A substitution $\sigma = \{v \mapsto \lambda(w) h(g(w))\}$, for instance, has no meaningful interpretation as a W-variable assignment unless the accessibility relation is transitive. Due to the interpretation of the W-valued function symbols f and g , the corresponding semantic function $\lambda(\Sigma)h'(g'(\Sigma))$ would map a world Σ to a world Σ' which is accessible in two steps,

The transformation rules are:

1. The toplevel call is: $\Pi(\mathcal{F}) := \pi(\mathcal{F}, 0, ())$ where $()$ is the empty list.
2. $\pi(\mathcal{F} \wedge \mathcal{G}, w, D\text{-vars}) := \pi(\mathcal{F}, w, D\text{-vars}) \wedge \pi(\mathcal{G}, w, D\text{-vars})$
3. $\pi(\mathcal{F} \vee \mathcal{G}, w, D\text{-vars}) := \pi(\mathcal{F}, w, D\text{-vars}) \vee \pi(\mathcal{G}, w, D\text{-vars})$
4. $\pi(\forall x \mathcal{F}, w, D\text{-vars}) := \forall x \pi(\mathcal{F}, w, D\text{-vars} + x)$
5. $\pi(\Box \mathcal{F}, w, D\text{-vars}) := \forall u \pi(\mathcal{F}, u(w), D\text{-vars})$
 u is added as a new W -variable to Σ_p .
6. $\pi(\exists x \mathcal{F}, w, D\text{-vars}) := \pi(\mathcal{F}, w, D\text{-vars})[x \leftarrow f(w+D\text{-vars})]$ (i.e. replace x by $f(w+D\text{-vars})$)
 f is added as a new $(1, |D\text{-vars}|)$ -place D -valued function symbol to Σ_p .
7. $\pi(\Diamond \mathcal{F}, w, D\text{-vars}) := \pi(\mathcal{F}, g(w+D\text{-vars}), D\text{-vars})$
 g is added as a new $(1, |D\text{-vars}|)$ -place W -valued function symbol to Σ_p .

Let P be an n -place predicate symbol and let f be an n -place function symbol.

8. $\pi(\neg P(t_1, \dots, t_n), w, D\text{-vars}) := \neg P(w, \pi(t_1, w, D\text{-vars}), \dots, \pi(t_n, w, D\text{-vars}))$
9. $\pi(P(t_1, \dots, t_n), w, D\text{-vars}) := P(w, \pi(t_1, w, D\text{-vars}), \dots, \pi(t_n, w, D\text{-vars}))$
10. $\pi(f(t_1, \dots, t_n), w, D\text{-vars}) := f(w, \pi(t_1, w, D\text{-vars}), \dots, \pi(t_n, w, D\text{-vars}))$
11. $\pi(x, w, D\text{-vars}) := x$ where x is a D -variable ■

Theorem: (Soundness and Completeness of the Translation Algorithm)

\mathcal{H} is a satisfiable modal formula if and only if $\Pi(\mathcal{H})$ is a P -satisfiable P -formula.

The proof follows the recursion of π and assures that the information about the modal context is correctly shifted from the nesting of the modal operators to the W -terms. ■

These results are the basis for a complete proof procedure: In order to prove that a modal logic formula is unsatisfiable, it is sufficient to prove that the translated P -logic formula is P -unsatisfiable.

6. Conjunctive Normal Form

Since the P -logic syntax contains no existential quantifier and no modal operators, a transformation of an arbitrary formula to an equivalent set of clauses is essentially the same as in predicate logic, but without the need to skolemize.

7. World-Paths - An Alternative Syntax for W -terms.

W -terms contain W -variable symbols in a functional position and are therefore higher order terms. For many purposes, especially for the definition of the unification algorithms, the purely first order “world-path” syntax that has been used in the introductory examples is much more convenient. The transition from W -terms to world-paths can be easily explained on the semantic level, where the function symbols and W -variable symbols are interpreted as functions, such that the currying operation is applicable. The currying operation transforms an n -place function f into an $n-1$ -place function f^c that produces a one-place function which, when applied to the remaining argument returns the same value as f when applied to all n arguments at once, i.e. $f(s_1, \dots, s_n) = s_1 f^c(s_2, \dots, s_n)$. Currying can be used to remove the “world argument” from the interpretation of an $(1, n)$ -place W -valued function symbol, leaving a function that is applicable to domain elements only. For a nested function call like $f(g(w, s_1, \dots, s_n), t_1, \dots, t_m)$ the equivalent curried function call looks like $g(w, s_1, \dots, s_n) f^c(t_1, \dots, t_m) = (w g^c(s_1, \dots, s_n)) f^c(t_1, \dots, t_m) = w (g^c(s_1, \dots, s_n) \circ f^c(t_1, \dots, t_m))$ where \circ is the composition of functions. A corresponding term that can be interpreted in such a way is (apply $(\circ g^c(s_1, \dots, s_n) f^c(t_1, \dots, t_m)) w$) or still simpler a list $[w g^c(s_1, \dots, s_n) f^c(t_1, \dots, t_m)]$, where the associativity of \circ can be exploited to remove the parentheses of nested terms.

A term $f^c(t_1, \dots, t_m)$ without a W -term as a first argument will be called a *CW-term* (curried world term) in the sequel. The interpretation of a CW -term $f(t_1, \dots, t_n)$ occurring in a world-path is that of a curried function mapping the corresponding domain elements $\mathcal{A}(t_1), \dots, \mathcal{A}(t_n)$ to a function which accepts a world and returns another world. The interpretation of W -variable symbols remains unchanged. A world-path evaluates under an interpretation \mathcal{A} to the same world as the corresponding W -term. Since there is a unique correspondence between a W -term and a world-path, we can use freely these two syntactic versions for denoting worlds.

When R is symmetric, two additional assumptions are necessary:

1. We assume that \mathbb{F}_W contains for every (1,n)-place W-valued function symbol f a corresponding (1,n)-place “inverse” function symbol f^{-1} , and when Θ assigns an injective world access function g to f, then Θ must assign a corresponding inverse world access function (see below) g^{-1} to f^{-1} .
2. We assume that \mathbb{V}_W contains for every W-variable symbol w an associated “inverse” W-variable symbol w^{-1} , and when μ assigns an injective world access function g to u then μ must also assign an associated inverse world access function g^{-1} to w^{-1} .

(Such “inverse symbols” never occur in formulae, however they may occur in substitutions! Their interpretations will be used only in situations where world access functions are injective.)

- b) An (1,n)-place world access function g is a mapping $g: \Sigma \times \mathbb{D}^n \rightarrow \Sigma$ such that for every $\Sigma \in \Sigma$ and $a_1, \dots, a_n \in \mathbb{D}$: $R(\Sigma, g(\Sigma, a_1, \dots, a_n))$ holds. (g maps worlds to accessible worlds.)
- c) An associated inverse world access function g^{-1} for the world access function g satisfies for every world w and domain elements s_1, \dots, s_n the following equation: $g^{-1}(g(w, s_1, \dots, s_n), s_1, \dots, s_n) = w$. ■

The next step is to say how a term is to be evaluated in a given interpretation. The idea is to evaluate the W-term w of a D-term $f(w, t_1, \dots, t_n)$ first, giving a signature interpretation Σ which determines the actual interpretation of f.

Definition: (P-Evaluation of Terms)

A P-interpretation $\mathcal{X} = (\mathbb{D}, \Sigma, \sigma, R, \Theta, \mu)$ can be turned into an interpreter for terms:

$\sigma(t)$	when t is a D-variable symbol.	
$\mathcal{X}(f(w, t_1, \dots, t_n))$	when $t = f(w, t_1, \dots, t_n)$ and f is a D-valued function symbol.	
$\mathcal{X}(0) := \Theta(0)$	when $t = 0$	
$\mu(u)(\mathcal{X}(w))$	when $t = u(w)$ and u is a W-variable symbol.	
$\Theta(g)(\mathcal{X}(w), \mathcal{X}(t_1), \dots, \mathcal{X}(t_n))$	when $t = g(w, t_1, \dots, t_n)$ and g is a W-valued function symbol.	■

Let $\mathcal{X}[x/a]$ denote the P-interpretation that differs from \mathcal{X} only in that the variable x is assigned the value a.

Definition: (The P-Satisfiability Relation \models_P)

Let $\mathcal{X} := (\mathbb{D}, \Sigma, \sigma, R, \Theta, \mu)$ be a P-interpretation for the signature Σ_P .

$\mathcal{X} \models_P P(w, t_1, \dots, t_n)$	iff $\mathcal{X}(w)(P)(\mathcal{X}(t_1), \dots, \mathcal{X}(t_n))$.	
$\mathcal{X} \models_P \neg P(w, t_1, \dots, t_n)$	iff not $\mathcal{X}(w)(P)(\mathcal{X}(t_1), \dots, \mathcal{X}(t_n))$.	
$\mathcal{X} \models_P (\mathcal{F} \wedge \mathcal{G})$	iff $\mathcal{X} \models_P \mathcal{F}$ and $\mathcal{X} \models_P \mathcal{G}$.	
$\mathcal{X} \models_P (\mathcal{F} \vee \mathcal{G})$	iff $\mathcal{X} \models_P \mathcal{F}$ or $\mathcal{X} \models_P \mathcal{G}$.	
$\mathcal{X} \models_P \forall x \mathcal{F}$ where $x \in \mathbb{V}_D$	iff for every $a \in \mathbb{D}$: $\mathcal{X}[x/a] \models_P \mathcal{F}$.	
$\mathcal{X} \models_P \forall u \mathcal{F}$ where $u \in \mathbb{V}_W$	iff for every world access function g: $\mathcal{X}[u/g] \models_P \mathcal{F}$.	■

Definition: (P-Models)

A P-interpretation \mathcal{X} is a P-model for a P-formula \mathcal{F} iff $\mathcal{X} \models_P \mathcal{F}$ (\mathcal{X} P-satisfies \mathcal{F}).

A P-formula \mathcal{F} is P-satisfiable if a P-model for \mathcal{F} exists. It is P-unsatisfiable if no P-model for \mathcal{F} exists. ■

5. Translation from Modal Logic Syntax to P-Logic Syntax.

We must define how to translate the modal logic signature into P-logic signature and the modal logic formulae into P-logic formulae.

1. Translation of the signature: We construct an initial P-signature $\Sigma_P := (\mathbb{V}_D, \mathbb{F}_D, \mathbb{P}, \{0\}, \emptyset)$ where $\mathbb{V}_D, \mathbb{F}_D, \mathbb{P}$ are the same variable, function and predicate symbols as in the modal logic formula.
2. Translation of terms and formulae: We define a translation function Π that takes a modal logic formula \mathcal{F} , translates it into a P-logic formula $\Pi(\mathcal{F})$ and updates the signature Σ_P with the generated W-variables that replace the \Box -operator and the skolem functions for the \exists -quantifier and the $\hat{\diamond}$ -operator. Π needs an auxiliary function π that makes the recursive descent into the modal formulae and terms. π records as a second argument the modal context in form of a W-term w and as a third argument the universally quantified variables D-vars. Take Σ_P to be a “global variable” that is updated during the recursive descent.

D-vars + x means the concatenation of a list D-vars = $(x_1 \dots x_n)$ with x, the result is $(x_1 \dots x_n x)$.
 $f(w + \text{D-vars})$ denotes the term $f(w, x_1, \dots, x_n)$ where D-vars = $(x_1 \dots x_n)$

Definition: (Terms, Atoms, Literals and Formulae)

Given a P-signature $\Sigma_P := (\mathbb{V}_D, \mathbb{F}_D, \mathbb{P}, 0, \mathbb{V}_W, \mathbb{F}_W)$,

- ▶ the set of *D-terms* \mathbb{T}_D over Σ_P is defined as the least set such that:
 - (i) $\mathbb{V}_D \subseteq \mathbb{T}_D$,
 - (ii) if $f \in \mathbb{F}_{D,n}$, $t_1, \dots, t_n \in \mathbb{T}_D$ and w is a *W-term* then $f(w, t_1, \dots, t_n) \in \mathbb{T}_D$.
 - ▶ The set of *W-terms* \mathbb{T}_W over Σ_P is defined as the least set such that:
 - (i) 0 is a *W-term*.
 - (ii) if $f \in \mathbb{F}_{W,n}$, $t_1, \dots, t_n \in \mathbb{T}_D$ and w is a *W-term* then $f(w, t_1, \dots, t_n) \in \mathbb{T}_W$.
 - (iii) if u is a *W-variable symbol* and t is a *W-term* then $u(t)$ is a *W-term*.
 - ▶ If $P \in \mathbb{P}_n$, $t_1, \dots, t_n \in \mathbb{T}_D$ and w is a *W-term* then $P(w, t_1, \dots, t_n)$ is a *P-atom*.
 - ▶ A *P-literal* is either a *P-atom* or a negated *P-atom*.
 - ▶ *P-formulae* are built with *P-literals*, the connectives \wedge, \vee and the universal quantifier \forall as usual.
- For convenience we assume that quantified variables are standardized apart, i.e. formulae like $\forall x(\exists x Px) \wedge Qx$ are not allowed and should be rewritten as $\forall x(\exists y Py) \wedge Qx$. ■

Examples for P-logic terms and formulae and their modal logic counterparts:

Modal Logic	P-logic	Signature
$\Box P$	$\forall w P(w(0))$	$P \in \mathbb{P}_0, w \in \mathbb{V}_W$
$\Diamond P$	$P(g(0))$	$P \in \mathbb{P}_0, g \in \mathbb{F}_{W,0}$
$\forall x \Diamond Q(x,a)$	$\forall x Q(h(0,x), x, a(h(0,x)))$	$Q \in \mathbb{P}_2, x \in \mathbb{V}_W, a \in \mathbb{F}_{D,0}, h \in \mathbb{F}_{W,1}$
$\Box \forall x (R(x) \wedge \Box \exists y \Diamond R(y))$	$\forall w \forall x (R(w(0), x) \wedge \forall v R(k(v(w(0)), x), r(v(w(0)), x)))$	
\uparrow	$\uparrow \uparrow \uparrow$	$R \in \mathbb{P}_1, v, w \in \mathbb{V}_W, k \in \mathbb{F}_{W,1}, r \in \mathbb{F}_{D,1}$.
w	$v \quad r \quad k$	

4.2 Semantics of P-Logic

The semantics of P-logics consists of three components, the interpretation of the signature, an evaluator for terms and the satisfiability relation. First the meaning of the symbols has to be defined. The intended meaning of *W-valued function symbols* is to convey the meaning of the \Diamond -operator whose interpretation is: From a given world Σ there exists an accessible world Σ' (possibly depending on some surrounding \forall -quantifiers) such that Thus, the object that is to be assigned to a *W-valued function symbol* must be a function - possibly depending on some domain arguments - which maps worlds to accessible worlds.

The *W-variables* are intended for conveying the meaning of the \Box -operator whose interpretation is: For all worlds Σ' which are accessible from a given world Σ A quantification $\forall w \dots$ over a *W-variable* must therefore be restricted on some worlds; moreover the restriction has a dynamic character - it depends on the modal context. The only way to incorporate this restriction is to assign to *W-variable symbols* functions which map worlds to accessible worlds. On the first glance, this gives a higher order character to P-logic, but this higher order character is so weak that it can be eliminated with the standard function currying trick (see below).

Definition: (P-Interpretation)

Given a signature $\Sigma_P := (\mathbb{V}_D, \mathbb{F}_D, \mathbb{P}, 0, \mathbb{V}_W, \mathbb{F}_W)$:

- a) By a *P-interpretation* for Σ_P we understand any tuple of the following form $(\mathbb{D}, \Sigma, \sigma, R, \Theta, \mu)$ where
- ▶ \mathbb{D} is a nonempty set, the *domain* of discourse.
 - ▶ Σ is a nonempty set of *D-signature interpretations*. Each *D-signature interpretation* is an assignment of “values” to each *D-valued function symbol* and *predicate symbol* in Σ_P as follows:
 - To each $(1,n)$ -place *D-valued function symbol* a mapping from \mathbb{D}^n to \mathbb{D} is assigned.
 - To each $(1,n)$ -place *predicate symbol* an n -place relation over \mathbb{D} is assigned.
 (The elements of Σ are the “worlds”.)
 - ▶ σ is a *D-variable assignment*, i.e. a mapping $\mathbb{V}_D \rightarrow \mathbb{D}$.
 - ▶ R is a relation on $\Sigma \times \Sigma$. (R is the accessibility relation.)
 - ▶ Θ is a *W-signature interpretation* that assigns values to 0 and the *W-valued function symbols*:
 - Θ assigns an element of Σ to 0 and to each $(1,n)$ -place *W-valued function symbol* g an $(1,n)$ -place world access function (see below).
 - ▶ μ is a *W-variable assignment* that assigns to each *W-variable* a one place world access function.

3. Modal Logic

Now we characterize briefly the particular modal logics we shall be considering:

3.1 Syntax

The formulae are those of first order predicate logic with two additional modal operators \Box (necessity) and \Diamond (possibility). In order to limit the syntactic variety of modal logic (without losing expressiveness), we only consider modal logic formulae in *negation normal form* without the implication and equivalence sign, that is all negation signs are moved in front of the atoms. Arbitrary formulae can be brought into this form using the appropriate transformation rules of predicate logic and the two additional rules $\neg\Box\mathcal{F} \rightarrow \Diamond\neg\mathcal{F}$ and $\neg\Diamond\mathcal{F} \rightarrow \Box\neg\mathcal{F}$.

3.2 Semantics

A common model theory for modal logics is S. Kripke's "possible worlds semantics" [Kr59, Kr63]. A "possible world" determines how the function and predicate symbols are to be interpreted in that world, i.e. it is an interpretation in the predicate logic sense. Different possible worlds may assign different meanings to the same symbol, but the universe is the same in all interpretations (constant-domain assumption!). There is one initial world and possibly infinitely many others which are "accessible" by an "accessibility relation" R . Two major classes of accessibility relations can be distinguished: serial and non-serial ones. We shall consider only serial accessibility relations, however in combination with the following properties of the relation R : reflexivity, symmetry and transitivity. In serial interpretations symmetric and transitive relations are also reflexive. Furthermore there is a special normal form for modal logic formulae when R is in fact an equivalence relation (modal logic S5) [HC68]. All formulae have at most one nested modal operator ("modal degree" one). In the sequel we shall assume for S5 interpretations that all formulae have been reduced to this normal form.

4. P-Logic

P-logics ("P" for Predicate logic style) as defined in this paper are syntactic variants of modal logics where the modal operators are replaced by "world-terms". A world-term represents the modal context, i.e. the sequence of nested modal operators, and is attached to the terms and atoms as an additional argument. It holds the information in which world the term or formula is to be interpreted. In order to preserve the possible worlds semantics for P-logic formulae, a world-term must denote a world and not a domain element. This suggests to formulate P-logic as a two-sorted logic with the two disjoint sorts D (for Domain) and W (for Worlds).

The world-paths that have been used in the introductory examples are syntactic variants of the world-terms and will be introduced below. World-terms are more appropriate for understanding the semantics of this structure, whereas world-paths are a suitable datastructure for the unification algorithms.

4.1 Syntax of P-Logic

As usual we begin with the definition of a signature which consists of a D -part and a W -part. The D -part, i.e. D -variable symbols, D -valued function symbols and predicate symbols, is the same as in modal logics. What is new is the W -part that is used for building world-terms. It consists of W -valued function symbols which are something like skolem functions and replace the \Diamond -operator, and W -variable symbols that replace the \Box -operator.

Definition: (Signature of P-Logic)

The alphabet for building P-logic terms and formulae consists of the logical connectives and the following symbols:

\mathbb{V}_D is a set of D -variable symbols.	\mathbb{V}_W is a set of W -variable symbols.
$\mathbb{F}_{D,n}$ is a set of $(1,n)$ -place D -valued function symbols.	\mathbb{F}_D is the union of all $\mathbb{F}_{D,n}$.
\mathbb{P}_n is a set of $(1,n)$ -place predicate symbols.	\mathbb{P} is the union of all \mathbb{P}_n .
0 is a W -valued constant symbol (denoting the initial world).	
$\mathbb{F}_{W,n}$ is a set of $(1,n)$ -place W -valued function symbols.	\mathbb{F}_W is the union of all $\mathbb{F}_{W,n}$.
$\Sigma_P := (\mathbb{V}_D, \mathbb{F}_D, \mathbb{P}, 0, \mathbb{V}_W, \mathbb{F}_W)$ is a P -signature .	

■

The premises $\diamond\diamond\forall x (\diamond Px \wedge \Box Qx)$ of the formula (*) can therefore be expressed in words as:

- \diamond From the initial world (say “0”) there is an accessible world a,
- \diamond from a there is an accessible world b,
- $\forall x$ such that for all x
 - (\diamond there is a world c, accessible from b (but depending on x, therefore c(x))
 - Px such that Px holds, where P is interpreted in world c(x)
 - $\wedge \Box$ and for all worlds u which are accessible from b
 - Qx) Qx holds, where Q is interpreted in world u.

The syntax transformation we are going to present in this paper records these worlds a, b, c(x) and u explicitly and attaches them as an additional “world-path” argument to the predicate and function symbols. Hence the context in which terms and atoms are to be interpreted can be recognized directly from this argument.

The above formula $\diamond\diamond\forall x (\diamond Px \wedge \Box Qx)$, for instance, is translated into $\forall x(P[0abc(x)]x \wedge \forall u Q[0abu]x)$ with the intuitive meaning:

- $\forall x (P[0abc(x)]x$ | For all x Px holds in all worlds which are accessible via the paths 0 a b c(x)
- $\wedge \forall u$ | and for all admissible worlds u (which worlds are actually admissible depends on the
- | world-paths in the subformulae of the quantifier.)
- $Q[0abu]x$ | Qx holds in all worlds which are accessible via the paths 0 a b u.

In order to prove the formula (*) by contradiction the consequence of the implication must be negated, i.e.:

$\neg\diamond(\forall yPy \wedge \forall zQz)$, and moving the negation sign inside yields: $\Box(\exists y\neg Py \vee \exists z\neg Qz)$. The transformed version is $\forall v(\neg P[0v]f[0v] \vee \neg Q[0v]g[0v])$ with the intuitive meaning:

- $\forall v$ | For all admissible worlds v
- $(\neg P[0v]f[0v]$ | $\neg Pf[0v]$ holds in all worlds which are accessible from 0.
- | f is a skolem function that denotes the original y “that must exist”,
- | but in each world v, there may exist another y. (f has no “ordinary” argument.)
- $\vee \neg Q[0v]g[0v])$ | or $\neg Qg[0v]$ holds in all worlds which are accessible from 0.
- | g is the second skolem function that depends also on v.

Eliminating the universal quantifiers the transformed negated formula (*) can be written in clause form as:

$$C1: P[0abc(x)]x \quad C2: Q[0abu]x \quad C3: \neg P[0v]f[0v], \neg Q[0v]g[0v]$$

There are two candidates for resolution operations, C2 with C3,2 and C1 with C3,1. Consider the first candidate, C2 with C3,2. Before generating a resolvent the atoms $Q[0abu]x$ and $Q[0v]g[0v]$ must be unified, i.e. the problem of unifying the two world-paths [0abu] and [0v] has to be solved. It is easy to see that unification is impossible unless the accessibility relation of the underlying logic is transitive. In this case $\{v \mapsto [abu]\}$ is a (most general) unifier of [0abu] and [0v]. Combining this substitution with the unifier for x and $g[0v]$, we obtain the final unifier $\{v \mapsto [abu], x \mapsto g[0abu]\}$. In case the accessibility relation is transitive and *serial* (i.e. from every world there exists an accessible world), the resolvent $\neg P[0abu]f[0abu]$ can be generated. Consider now the second candidate for resolution, C1 with C3,1, which gives rise to the unification problem $P[0abc(x)]x$ and $P[0v]f[0v]$. Again we can unify the world-paths [0abc(x)] and [0v] only in logics with a transitive accessibility relation. The unifier is $\{v \mapsto [abc(x)]\}$. This substitution can be applied to the remaining terms before the unification proceeds and the second unification problem: unify x with $f[0abc(x)]$ now fails with an occur check failure. The atoms $P[0abc(x)]x$ and $P[0v]f[0v]$ are not unifiable. Since there is no other possibility for resolution, the proof fails; and in fact, the formula (*) is not a theorem.

A Resolution Calculus for Modal Logics

Hans Jürgen Ohlbach

FB Informatik, University of Kaiserslautern, W-Germany

uucp: ...!seismo!unido!uklirb!ohlbach

Abstract A syntax transformation is presented that eliminates the modal logic operators from modal logic formulae by shifting the modal context information to the term level. The formulae in the transformed syntax can be brought into conjunctive normal form such that a clause based resolution calculus without any additional inference rule, but with special modal unification algorithms, can be defined. The method works for first-order modal logics with the two operators \Box and \Diamond and with constant-domain Kripke semantics where the accessibility relation is serial and may have any combination of the following properties: reflexivity, symmetry, transitivity. In particular the quantified versions of the modal systems T, S4, S5, B, D, D4 and DB can be treated. Extensions to non-serial and varying-domain systems are possible, but not presented here.

Key words: modal logic, resolution principle, unification

1. Introduction

A straightforward application of the resolution principle [Ro65] to modal logic fails because two syntactically complementary literals may be in different modal contexts and are therefore not necessarily semantically contradictory. Consequently most of the proposed deduction calculi for modal logics are based on tableaux (eg. [Fit72], [Fit83]). Some attempts to adapt the ideas of the standard resolution principle to modal logics have stopped at the half-way point. The method proposed by Fariñas del Cerro [Fa85] for example cannot treat full modal logics, since quantifiers are not allowed in the scope of modalities. Abadi and Manna's systems [AM86] restrict the application of the resolution rule to modal contexts in which it is sound. Inference across modal operators is performed by additional Hilbert-style deduction rules which generate a very redundant search space. Chan's method [Ch87], based on an idea which is in fact very similar to the one presented in this paper, is restricted to the propositional S4 system. The most elaborated work on efficient proof systems currently seems to be L.A. Wallen's matrix proof methods for modal logics [Wal87]. It is the first system that computes the modal context that allows an inference operation with a deterministic unification algorithm, and not by nondeterministic search. In this paper a formalism is presented that fits into the paradigm of the resolution principle for predicate logic and can therefore easily be integrated into an existing clause based resolution theorem prover. The soundness and completeness proofs as well as an extension of the method to logics with non-serial accessibility relations can be found in [Oh88].

2. An Example that Demonstrates the Basic Ideas.

Consider the formula $\Diamond \Diamond \forall x (\Diamond Px \wedge \Box Qx) \Rightarrow \Diamond (\forall y Py \wedge \forall z Qz)$ (*)

The meaning of such a formula can be described in terms of possible worlds which are connected by an accessibility relation [HC68]. A possible world is an interpretation in the predicate logic sense. It determines how the function and predicate symbols are to be interpreted in that world. The nesting of the modal operators in a formula determines which world or which interpretation respectively is actually meant. $\Diamond \mathcal{F}$ for instance means "there is a world b which is accessible from the current world a, such that \mathcal{F} holds under the interpretation of b". $\Box \mathcal{F}$ means "for every world b which is accessible from the current world a, \mathcal{F} holds under the interpretation of b".

This work was supported by the Sonderforschungsbereich 314 of the Deutsche Forschungsgemeinschaft and by the ESPRIT project 1033 of the European Community.