A Decision and Minimization Procedure for Modal Logic

by

Wanda B. K. Boyer
B.Sc., University of Victoria, 2012

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

A Decision and Minimization Procedure for Modal Logic

by

Wanda B. K. Boyer
B.Sc., University of Victoria, 2012

ABSTRACT

This thesis describes a decision and minimization procedure for modal logic. The decision procedure answers the question of whether there exists a satisfying pointed model for a formula which obeys user-specified first-order conditions on the underlying frame. Then the minimization procedure produces a minimal model with respect to the number of worlds that satisfies the desired formula while obeying the requisite conditions on the underlying frame. A proof of correctness for the decision and minimization procedures is supplied, as well as a description of an implementation built upon the Enfragmo model expansion solver.

Table of Contents

List of Figures

List of Definitions

Acknowledgements

First, I would like to thank my supervisors, Dr. Bruce Kapron and Dr. Audrey Yap, and my external member Dr. David Mitchell, for their unparalleled technical knowledge and leadership, as well as their unceasing support in writing this thesis. I would also like to thank Wendy Beggs, the Graduate Secretary for Computer Science, for her kindness and grace in helping me to surmount major obstacles that have arisen in my pursuing this degree.

To my Mother, Dr. Wanda Boyer; my father, Dr. John Boyer; and my father-in-law, David Palmer-Stone: thank you for applying your deep mastery of technical writing to the proofreading of my work, and for your extremely deep, involved, and thought-provoking questions.

To my mother-in-law, Brenda Palmer-Stone; my Grandmother, Barbara Rumson; my uncle, Gordon A. Rumson; and my Grandfather, Gordon E. Rumson: thank you for pushing me to strive for excellence as I have completed this degree. Your love has sustained me.

To my friends Charles Leitz, Mohammad Hajiabadi, and Reid Howard, I wish to extend my sincerest thanks for discussing crucial parts of my thesis, and for thoroughly proof-reading the results. I am so thankful for how generous you have been with your time in helping me! Thanks as well to Matt Hemmings for your time in hearing me present and insight into further avenues of research; you and Jen are positively wonderful people!

To Rachel Caulfield, Chelsey Hennessey, and Sumaiya Maria, our study dates were invaluable in my journey. You kept me company through the dark nights and grim days of poring over Blackburn, bug-hunting and subsequent squashing, and LaTeX mayhem. To all my other friends, thank you for your support and eternal optimism! I wouldn't have been able to do this without all of you.

Finally, to my sweetheart, Devin Palmer-Stone: thank you for your love and support. Here's to all the adventures we've had together, and to all that shall come!

Chapter 1

Introduction

1.1 Problem Statement and Motivation

Modal logics can be thought of as extending propositional logic with new operators, such as $\Box$ (necessity) and $\Diamond$ (possibility), which act as qualifiers for statements. For example, in Schrödinger's thought experiment, the sentence "the cat is alive" is possibly true, as is "the cat is not alive". Hence, "it is not necessarily true that the cat is dead", since a propositional state in which the cat is alive is considered possible, and likewise, "it is not necessarily true that the cat is alive" also holds because we consider it possible that the cat is dead.

Interest in modal logics stems not only from studies in model theory, but also from application to new domains, such as in the investigation of interactions between epistemic agents or in the verification of software [4, 5, 8]. Propositional logic is too weak to express linguistic modalities and falls prey to the paradoxes of material implication, and first-order logic, although sufficiently strong to express the necessary concepts in these problem spaces, is undecidable. The modal logics considered in this thesis possess the properties of soundness and completeness along with decidability, which make them suitable for application in these novel domains [3, 4].

Solvers that produce modal proofs of satisfiability by applying the rules of inference may make unexpected deductive steps that fail to be explanatory and do not provide techniques applicable to other formulas with similar features. The proofs produced through this syntactic approach often include inefficient choices in deductive steps, and therefore fail to evince the elegance of human inventiveness in their style.

An alternative to constructing a deductive modal proof is to build a relational structure called a pointed model in which the formula holds. Soundness and completeness results afford a different perspective of the the relationship between provability from axioms and satisfiability in a pointed model with certain characteristics: as opposed to applying rules of inference to propositional tautologies and axiom schemes to demonstrate that a formula is a theorem of the modal language, the frame validity of axioms directly impose conditions on the relational structure of the model in which a formula is satisfied.

Of the existing semantic approaches by which a model is produced, a common thread emerges: that a decision procedure only answers the question of whether a pointed model for the formula exists, but does not furnish the user with such a model in an immediate form. Explicitly constructing a mathematical entity in the course of demonstrating satisfiability is interesting because it allows for further scrutiny, shedding light on the object represented by the formula.

With these considerations in mind, this thesis will describe a process that answers the following questions:

1. Given φ and the first-order correspondent α of an axiom schema ψ, can we find $\mathfrak{M}, w$ such that $\mathfrak{M}, w \models \varphi$ and $\mathfrak{F} \models \psi$?
2. Can we correctly perform minimization to find $\mathfrak{M}', w'$ with the minimum number of worlds necessary for $\mathfrak{M}', w' \models \varphi$ with $\mathfrak{F}' \models \psi$?
3. Can we enable the visualization of this data to facilitate higher-level analysis?

1.2 Overture

Chapter 1 presents an overview of the basics of modal logic and the tools necessary to understand the foundation upon which this decision and minimization procedure is based are presented.

Chapter 2 first details the core of the decision procedure, then describes the doubling mechanism by which the question of satisfaction is answered, applying a variant of binary search to finding a satisfying pointed model with the minimum number of worlds, and the mechanism by which we produce visualizations of pointed models.

Chapter 3 expounds a proof of correctness of the doubling and halving procedure to find a satisfying pointed model for φ that is minimal with respect to the number of worlds. This proof dictates that if we have a model for φ of size n that obeys an arbitrary modal axiom schema ψ, then there exists a model of size $n+1$ that satisfies φ which obeys that same set of conditions on its accessibility relation.

To give a concrete example of the functioning of the system, Chapter 4 presents a sample formula, the corresponding Enfragmo problem instance file, and the effects of different axiom schemata on the resulting models. The raw XML output is given first to demonstrate the necessity of parsing the output to the .dot graph representation format for increased clarity, and the benefits of having a concrete visualization for analysis of the model produced.

Finally, Chapter 5 at first mirrors this overture, summarizing the contributions made within this thesis and significance to applications. Then, avenues for future investigation are suggested, with some first clues as to how to pursue them.

1.3 Background Information

1.3.1 Basic syntax and semantics of Modal Logic

This review takes highlights from *Modal Logic for Open Minds* [17], *Modal Logic* [3], and *The Handbook of Modal Logic* [4].

The syntactic structure of the modal language is defined inductively from proposition letters $PROP = \{p, q, r, \ldots\}$ and propositional constants $\top, \bot$ with the following operators:

$$\varphi := PROP \mid \bot \mid \top \mid \neg\varphi \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\varphi \to \psi) \mid (\varphi \leftrightarrow \psi) \mid \Diamond\varphi \mid \Box\varphi$$

Strictly speaking, we only require the functionally complete set of operators $\{\neg, \vee, \Box\}$ to define our modal language; however, adding symbols serves to simplify expressions without affecting equivalence. These additional operators and relevant equivalences are as follows:

$$\begin{aligned} \varphi \wedge \psi &\equiv \neg(\neg\varphi \vee \neg\psi) \\ \varphi \to \psi &\equiv \neg\varphi \vee \psi \\ \varphi \leftrightarrow \psi &\equiv (\varphi \to \psi) \wedge (\psi \to \varphi) \\ \Diamond\varphi &\equiv \neg\Box\neg\varphi \end{aligned}$$

Normal modal logics stem from the Hilbert-style deductive system $\mathbb{K}$. The language of $\mathbb{K}$ is comprised of all formulas derivable through the application of a finite sequence of rules of proof to the axioms of the system. The axioms of $\mathbb{K}$ are all propositional tautologies, plus:

K axiom	$\Box(p \to q) \to (\Box p \to \Box q)$
Duality	$\Box p \leftrightarrow \neg\Diamond\neg p$

The rules of proof are:

Modus Ponens	If φ holds and we know that $\varphi \to \psi$, then ψ must hold.
Uniform Substitution	Allows us to consider axioms as schema, where we may substitute any formula φ for proposition letter p, as long as it is done consistently throughout the schema.
Generalization	Given φ, $\Box\varphi$ is a consequence.

Therefore, a formula is provable in $\mathbb{K}$ if it is the final member of a sequence of axioms and rules of proof applied to those axioms [3, p. 33].

One of the foremost interpretations of the modal operators is that of necessity ($\Box$) and possibility ($\Diamond$). The duality of $\Box$ and $\Diamond$, as given above, can be taken to mean "It is *necessary* that φ if and only if it is not *possible* that $\neg\varphi$." Different interpretations of these modal operators require different axiomatizations. For example, the $\mathbb{T}$ axiom of epistemic logic (that whatever an agent knows must be true: $K_a\varphi \to \varphi$) is used to model the knowledge of agents, whereas doxastic operators, using the weaker D axiom, $\neg B_a\bot$, model agents' beliefs.

The semantic framework of modal logic is defined in terms of so-called possible worlds models, whose relationship to the modal deductive system was touched upon in § 1.1.

Definition 1.3.1 (Kripke Structure). In our basic modal framework, Kripke structure is a 3-tuple $\mathfrak{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle$, where $\mathcal{W} \neq \emptyset$ is the set of possible worlds, $\mathcal{R}$ is the binary accessibility relation between worlds, and $\mathcal{V} : \mathcal{A} \times \mathcal{W} \to \{0, 1\}$ is the valuation map (i.e. $\mathcal{V}$ tells us whether proposition letter p holds at world w). ♦

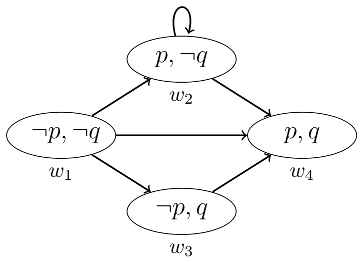
Truth in a Kripke structure is evaluated at a particular world $w \in \mathcal{W}$, where the statement "φ holds at w in $\mathfrak{M}$" is represented by the metastatement $\mathfrak{M}, w \models \varphi$.

Definition 1.3.2 (Kripke Semantics in a Pointed Model)**.** The truth of a modal formula φ at a world w of Kripke structure $\mathfrak{M}$, written $\mathfrak{M}, w \models \varphi$ for pointed model $\mathfrak{M}, w$, is defined inductively:

$$\begin{array}{lcr}
\mathfrak{M}, w \models p & \text{iff} & V(p, w) = 1 \\
\mathfrak{M}, w \models \neg\varphi & \text{iff} & \text{not } \mathfrak{M}, w \models \varphi \\
\mathfrak{M}, w \models \varphi \wedge \psi & \text{iff} & \mathfrak{M}, w \models \varphi \text{ and } \mathfrak{M}, w \models \psi \\
\mathfrak{M}, w \models \Box\varphi & \text{iff} & \forall v \text{ such that } wRv,\ \mathfrak{M}, v \models \varphi \\
\mathfrak{M}, w \models \Diamond\varphi & \text{iff} & \exists v \text{ such that } wRv,\ \mathfrak{M}, v \models \varphi
\end{array}$$

As we can see, the accessibility relation $\mathcal{R}$ restricts quantification over possible worlds as we unpack modal operators. ♦

Example 1.3.1. Consider the following model:



The pointed model $\mathfrak{M}, w_1$ satisfies the formula: $\mathfrak{M}, w_1 \models \Diamond\Box p$, since the edge (w_1, w_2) brings us to world w_2, where all worlds accessible by $\mathcal{R}$ from w_2 satisfy proposition letter p (namely worlds w_2 and w_4). ◀

To return to the duality between $\Box$ and $\Diamond$, the semantic perspective reveals the connection to the duality of universal ($\forall$) and existential ($\exists$) quantification: if *all* worlds accessible from the current state make φ true, then it is *not possible* that there is an accessible world where φ is false. These semantics also capture the notion of uncertainty, where if we can access two different worlds from the current state where the valuation for some formula φ differs at those two worlds, then we are uncertain as to whether the formula holds; it is *possible* that the formula is true, but it is also possible that the formula is false.

1.3.2 Frame Semantics

Frames are an important abstraction in modal semantics, allowing us to divorce the structure of a model from the valuation, and to provide a means for clearly expressing properties of families of models sharing the same underlying frame.

Definition 1.3.3 (Frame). A frame is a pair $\mathfrak{F} = \langle \mathcal{W}, \ \mathcal{R} \rangle$ where $\mathcal{W}$ is a nonempty set of worlds or states, and $\mathcal{R}$ is a binary relation on $\mathcal{W}$, referred to as the accessibility relation. ⧫

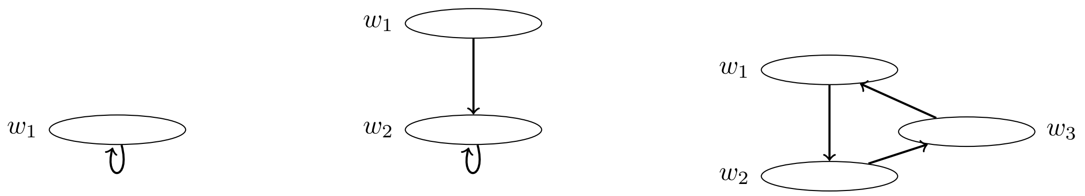
When a formula φ is *valid* in a frame $\mathfrak{F}$, written $\mathfrak{F} \models \varphi$, then it holds at every state in the frame regardless of the valuation, and therefore must hold everywhere in any model $\mathfrak{M} = \langle \mathfrak{F}, \mathcal{V} \rangle$ with arbitrary valuation map $\mathcal{V}$. We can use modal formulas to define classes of frames with unique relational structures through the concept of frame validity.

Definition 1.3.4 (Definability). A formula φ is said to characterize a class of frames $\mathbb{F}$ relative to a superclass $\mathbb{C}$ of frames if for all $\mathfrak{F} \in \mathbb{C}$, we have $\mathfrak{F} \in \mathbb{F} \Leftrightarrow \mathfrak{F} \models \varphi$. ⧫

A formula, when taken as an axiom schema, immediately has second-order semantics when viewed from the perspective of frame validity. The arbitrariness of valuation maps in this context translates to the notion of quantifying over predicates in the second-order language. However, there are classes of modal formulas for which these second-order correspondents collapse down into global first-order frame correspondents.

Definition 1.3.5 (Frame Correspondence). If a class of frames $\mathbb{F}$ can be defined by a modal formula φ as well as by a formula α from a first-order frame language (i.e. sentences involving $\mathcal{R}$ with equality), in which we have predicates for each propositional variable and binary relations over the set of worlds, then φ and α are global frame correspondents, providing two different perspectives on the same property. This correspondence is unique up to logical equivalence. ♦

Example 1.3.2 (A modal formula and its global first-order correspondent)**.** The formula $\Box\varphi \to \Diamond\varphi$ has global first-order correspondent $\forall x \exists y \; s.t. \; \mathcal{R}(x, y)$, and defines the class of serial frames, which have no terminal worlds. Some members of the class of serial frames include:



◀

1.3.3 Modal axioms with first-order correspondents

Because the Enfragmo specification language is an *existential second order language* (Abbreviated as $\exists SO$), it is desirable to be able to constrain the accessibility relation in either $\exists SO$ or first-order (FO) terms.

When considering an axiom characterization for an application, it is necessary to determine what features are required of the accessibility relation, and if these axioms have a first-order correspondent. When axioms are taken as schema, they are considered valid in the underlying frame of a model; therefore, there is an implicit translation to second order logic, where we quantify over predicates corresponding to the propositional atoms (i.e. quantify over valuation maps, since the formula must hold at all worlds regardless of valuation). For example, the McKinsey axiom $\Box\Diamond p \to \Diamond\Box p$ does not have a first-order correspondent, as it violates the Löwenheim-Skolem theorem [3, p. 134].

Modal axioms of an appropriate form, namely Sahlqvist formulas, are guaranteed to have first-order correspondents, and are subject to a translation algorithm to obtain their global first-order correspondents. These conditions and a translation procedure are given in detail in Blackburn [3, pp. 157-167, 168-179], as well as in van Benthem's *Modal Logic and Classical Logic*[16].

For modal formulas outside of the class of Sahlqvist formulas, the prospects for finding their first-order correspondents are grim: Chagrova's result tells us that the problem of determining if a modal formula has a first-order correspondent is undecidable [3, p. 168].

For a more detailed list of common modal axiom schemes and their first-order correspondents, refer to Tables 25.1 and 25.2 in the chapter "SAT Techniques for Modal and Description Logics" in *The Handbook of Satisfiability* [15]. The following table gives common normal modal axiom schema and their semantic characterizations:

Axiom Schema	**Property**	
B. $\neg\varphi \to \Box\neg\Box\varphi$	Symmetric	$\forall u, v\Big(\mathcal{R}(u,v) \to \mathcal{R}(v,u)\Big)$
D. $\neg\Box\bot$	Serial	$\forall u \exists v\Big(\mathcal{R}(u,v)\Big)$
T. $\Box\varphi \to \varphi$	Reflexive	$\forall u\Big(\mathcal{R}(u,u)\Big)$
4. $\Box\varphi \to \Box\Box\varphi$	Transitive	$\forall u, v, w\Big(\Big(\mathcal{R}(u,v) \wedge \mathcal{R}(v,w)\Big) \to \mathcal{R}(u,w)\Big)$
5. $\neg\Box\varphi \to \Box\neg\Box\varphi$	Euclidean	$\forall u, v, w\Big(\Big(\mathcal{R}(u,v) \wedge \mathcal{R}(u,w)\Big) \to \mathcal{R}(v,w)\Big)$

1.3.4 Constructions that preserve truth

Certain constructions have been defined to further illuminate the relationship between the current model theoretic system and systems that subsume it. Of particular interest are those constructions which preserve the truth of formulas, or those for which validity in the old structure implies validity in the new construction.

Bisimulation

Bisimulations between models are one variety of truth-preserving relations that can be constructed between models, thereby demonstrating underlying similarities in the models' structures. To recall § 1.1, the *theories* (i.e. sets of formulas satisfied by) bisimilar models are identical: any formula that holds in one of the models must hold in the other. In a sense, the theory is unable to "distinguish" between these bisimilar models [3, p. 68].

Definition 1.3.6 (Bisimulation). A *bisimulation* is a relation $\mathcal{E} \subseteq \mathcal{W} \times \mathcal{W}'$ between two models $\mathfrak{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle$ and $\mathfrak{M}' = \langle \mathcal{W}', \mathcal{R}', \mathcal{V}' \rangle$ with the following properties:

1. $\forall w \in \mathcal{W},\ w' \in \mathcal{W}' :\ w\mathcal{E}w' \Leftrightarrow \mathcal{V}(w) = \mathcal{V}'(w')$

2. (Forth) if $w\ \mathcal{E}\ w'$ and $w\ \mathcal{R}\ v$, then $\exists v' \in \mathcal{W}'$ such that $w'\ \mathcal{R}'\ v'$ and $v\ \mathcal{E}\ v'$

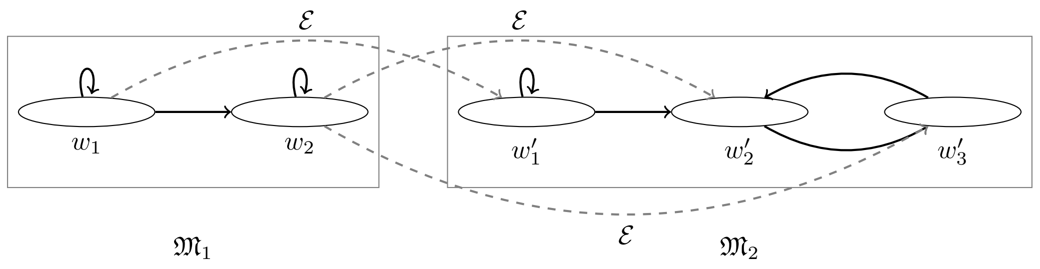
3. (Back) if $w\ \mathcal{E}\ w'$ and $w'\ \mathcal{R}'\ v'$, then $\exists v \in \mathcal{W}$ such that $w\ \mathcal{R}\ v$ and $v\ \mathcal{E}\ v'$

That is, the valuation maps must correspond between worlds w and w', and any relational "step" that can be taken in model $\mathfrak{M}$ must have a corresponding step in $\mathfrak{M}'$. ♦

Theorem 2.20 in Blackburn [3, p. 67] gives an inductive proof that the truth of modal formulas is preserved. The converse of this result comes from the Hennessey-Milner Theorem, which states that if two models verify the same formulas, there must exist a bisimulation between them.

Interestingly enough, it is possible to construct bisimulations between models with disparate frame conditions:

Example 1.3.3. Consider the figure below. First, note that the left model $\mathfrak{M}_1$ is reflexive, and that the right model $\mathfrak{M}_2$ is not. The bisimulation $\mathcal{E}$ is drawn in dashed arcs between bisimilar worlds.



Since the valuation map is trivial (i.e. all propositional atoms are false at all worlds, so their assignments correspond automatically), we focus on the forth and back conditions between $\mathfrak{M}_1$ and $\mathfrak{M}_2$.

For $w_1\mathcal{E}w'_1$:

$w_1\mathcal{R}w_1$ and $w'_1\mathcal{R}'w'_1$ $\qquad$ $w_1\mathcal{E}w'_1$

$w_1\mathcal{R}w_2$ and $w'_1\mathcal{R}'w'_2$ $\qquad$ $w_2\mathcal{E}w'_2$

So all successors of w_1 have corresponding successors in $\mathfrak{M}_2$.

Similarly, $w_2\mathcal{E}w'_2$:

$w_2\mathcal{R}w_2$ and $w'_2\mathcal{R}'w'_3$ $\qquad$ $w_2\mathcal{E}w'_3$

And finally, for $w_2\mathcal{E}w'_3$:

$w_2\mathcal{R}w_2$ and $w'_3\mathcal{R}'w'_2$ $\qquad$ $w_2\mathcal{E}w'_2$

And therefore all successors of w_2 have corresponding successors in $\mathfrak{M}_2$ that are appropriately linked by bisimulation $\mathcal{E}$.

Note that even if a number of worlds can "see" themselves, if there exists even one non-reflexive world in the model, the axiom $\Box\varphi \to \varphi$ corresponding to the reflexive frame condition cannot possibly be valid in the underlying frame. ◀

Finally, it is of note that an algebra over relations is induced by bisimulations, with the operator being the composition of functions: if you have a bisimulation $\mathcal{E}$ between $\mathfrak{M}$ and $\mathfrak{M}'$, and another bisimulation $\mathcal{E}'$ between $\mathfrak{M}'$ and $\mathfrak{M}''$, then the composition $E'' = \mathcal{E} \circ \mathcal{E}'$ is a bisimulation between $\mathfrak{M}$ and $\mathfrak{M}''$.

Bounded morphisms

Bounded morphisms are a particular variety of bisimulations, where we make the further requirement that the relation be a function.

Definition 1.3.7 (Bounded morphism)**.** A bounded morphism from model $\mathfrak{M}$ to $\mathfrak{M}'$ is a function $f : \mathfrak{M} \to \mathfrak{M}'$ which obeys the following conditions:

1. $\forall w \in \mathcal{W},\ \ \mathcal{V}(w) = \mathcal{V}'(f(w))$
2. If $a\mathcal{R}b$ in $\mathfrak{M}$, then $f(a)\mathcal{R}'f(b)$ in $\mathfrak{M}'$.
3. Suppose $f(a)\mathcal{R}'z$. Then $\exists b \in \mathcal{W}$ such that $a\mathcal{R}b$ and $f(b) = z$.

♦

From *The Handbook of Modal Logic*, in Corollary 16 [4, p. 259], we have the following results regarding the desired truth-preserving properties. Given that $f : \mathfrak{M} \to \mathfrak{M}'$ is a bounded morphism:

1. $\forall u \in \mathcal{W}$, $\mathfrak{M}, u \models \varphi \Leftrightarrow \mathfrak{M}', f(u) \models \varphi$.
2. If f is surjective, then $\mathfrak{M} \models \varphi \Leftrightarrow \mathfrak{M}' \models \varphi$.
3. $\forall u \in \mathcal{W}$, if $\mathfrak{F}, u \models \varphi$, then $\mathfrak{F}', f(u) \models \varphi$.
4. If f is surjective, then $\mathfrak{F} \models \varphi \Rightarrow \mathfrak{F}' \models \varphi$.

Of particular interest to this thesis in Chapter 3 are parts 1 and 4. When we consider actual world w in the context of part 1, the existence of a bounded morphism $f : \mathcal{W} \to \mathcal{W}'$ dictates that models $\mathfrak{M}$ and $\mathfrak{M}'$ satisfy the same formulas. The existence of surjective bounded morphisms $f : \mathcal{W} \to \mathcal{W}'$ and $g : \mathcal{W}' \to \mathcal{W}$ in conjunction with part 4 implies that the underlying frames $\mathfrak{F}$ and $\mathfrak{F}'$ validate the same formulas; in particular, they share the same axiom schemata, and therefore their relations $\mathcal{R}$ and $\mathcal{R}'$ have the same properties imposed by the axiom schemata.

Disjoint unions

Another truth-preserving construction is that of the disjoint union of models, where a copy of each of the world-disjoint models over which the disjoint union is taken is included as an "island" in the new structure. Of particular interest to the proof of correctness in Chapter 3 is that we may use this construction to grow the domain of a given model with frame $\mathfrak{F}$, resulting in a new model with frame $\mathfrak{F} \uplus \mathfrak{F}$. This allows us to build a bounded morphism onto another frame, $\mathfrak{F}'$, which was built to possess precisely one more world than $\mathfrak{F}$, and would therefore otherwise not be a feasible domain for a surjective function.

Definition 1.3.8 (Disjoint union of models)**.** For disjoint models $\mathfrak{M}_i = \langle \mathcal{W}_i, \mathcal{R}_i, \mathcal{V}_i \rangle$ with $i \in \mathbb{I}$, then:

$$\biguplus_{i \in \mathbb{I}} \mathfrak{M}_i = \langle \{(x,i) \mid x \in \mathcal{W}_i\},\ \{\big((x,i),(y,i)\big) \mid (x,y) \in \mathcal{R}_i\},\ \bigcup_{i \in \mathbb{I}} \Big\{\{(w,i) \mid w \in \mathcal{V}_i(p)\}\Big\} \rangle$$

♦

The important truth preserving properties of the disjoint union are presented in Proposition 26 of *The Handbook of Modal Logic*[4, p. 262]. Given a family of frames $\{\mathfrak{F}_i\}$ where $\mathfrak{F}_i = \langle \mathcal{W}_i, \mathcal{R}_i \rangle$ and models $\{\mathfrak{M}_i\}$ where each $\mathfrak{M}_i = \langle \mathfrak{F}_i, \mathcal{V}_i \rangle$:

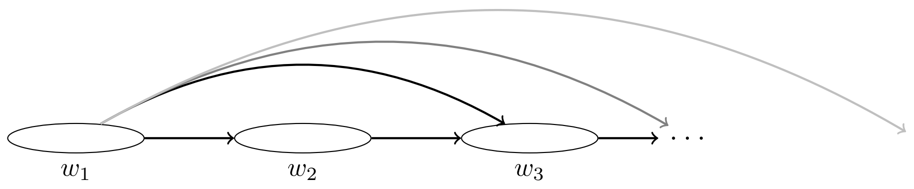
1. $\forall i \in \mathbb{I}$ and $w \in \mathcal{W}_i$: $\mathfrak{M}_i, w \models \varphi \Leftrightarrow \biguplus_{i \in \mathbb{I}} \mathfrak{M}_i, (w,i) \models \varphi$
2. $\forall i \in \mathbb{I}$ and $w \in \mathcal{W}_i$: $\mathfrak{F}_i, w \models \varphi \Leftrightarrow \biguplus_{i \in \mathbb{I}} \mathfrak{F}_i, (w,i) \models \varphi$
3. $\biguplus_{i \in \mathbb{I}} \mathfrak{M}_i \models \varphi \Leftrightarrow \mathfrak{M}_i \models \varphi$ for every $i \in \mathbb{I}$
4. $\biguplus_{i \in \mathbb{I}} \mathfrak{F}_i \models \varphi \Leftrightarrow \mathfrak{F}_i \models \varphi$ for every $i \in \mathbb{I}$

Specifically, the validity preserving properties of part 4 are used to show that since $\mathfrak{F}$ and $\mathfrak{F} \uplus \mathfrak{F}$ validate the same axiom schema, we are able to construct a bounded morphism from $\mathfrak{F} \uplus \mathfrak{F}$ to $\mathfrak{F}'$ to complete one direction of the proof.

1.3.5 Finite Frames and the Finite Model Property

The modal logic $\mathbb{K}$ fails to have the expressive power to dictate that a formula must only have infinite models [3, p. 93]. However, when we allow for different axiom schemata, possibly of first-order without modal correspondents, we may not necessarily be able to find a finite model for a formula.

Example 1.3.4. An example of a characterization that forces an infinite chain of modal states with no terminating world would be an irreflexive, transitive, and antisymmetric frame in which $\neg\Box\bot$ holds. Since no world w may satisfy $\bot$, a world with no successors vacuously satisfies the formula $\Box\bot$; therefore, if $\neg\Box\bot$ holds, then by duality, $\Diamond\neg\bot \Leftrightarrow \Diamond\top$ holds, and so every world must have a successor that was not a previous member of the chain.



Note especially that transitivity works in concert with the conditions of irreflexivity and antisymmetry to ensure that the underlying frame is acyclic. Irreflexivity and antisymmetry alone do not preclude the existence of cycles, therefore allowing for models with finite cycles in the system, which contradicts our goal of obtaining only infinite models. ◀

There are techniques for proving that an axiom characterization that extends the basic modal logic $\mathbb{K}$ possesses the finite model property, the foremost of which is filtration.

Definition 1.3.9 (Subformula-closed set)**.** A set Σ is said to be *subformula-closed* when:

1. $\varphi \wedge \psi \in \Sigma \Rightarrow \varphi, \psi \in \Sigma$.
2. $\neg\varphi \in \Sigma \Rightarrow \varphi \in \Sigma$.
3. $\Diamond\varphi \in \Sigma \Rightarrow \varphi \in \Sigma$

♦

Definition 1.3.10 (Equivalence relation under subformula-closed set Σ)**.** Define the equivalence relation $\leftrightsquigarrow_\Sigma$ on worlds in $\mathfrak{M}$ as follows:

$$w \leftrightsquigarrow_\Sigma v \Leftrightarrow \Big(\forall\varphi \in \Sigma\ (\mathfrak{M}, w \models \varphi \Leftrightarrow \mathfrak{M}, v \models \varphi)\Big)$$

This induces equivalence classes $\mid w \mid_\Sigma = \Big\{v \ \Big|\ w \leftrightsquigarrow_\Sigma v\Big\}$ on the domain $\mathcal{W}$ of $\mathfrak{M}$. ♦

Definition 1.3.11 (Filtration)**.** The *filtration* of a model $\mathfrak{M}$, written $\mathfrak{M}^f$, under subformula-closed set Σ is the transformation of the domain $\mathcal{W}$ to the set of equivalence classes induced by Σ, according to Definition 1.3.10.

1. $\mathcal{W}^f = \mathcal{W}_\Sigma = \Big\{ \mid w \mid_\Sigma \ \Big|\ w \in \mathcal{W}\Big\}$
2. $\mathcal{V}^f(p) = \Big\{ \mid w \mid \ \Big|\ \mathfrak{M}, w \models p\Big\}$ for each atom $p \in \Sigma$.
3. If $u\mathcal{R}v$, then $\mid u \mid \mathcal{R}^f \mid v \mid$
4. When $\mid u \mid \mathcal{R}^f \mid v \mid$, then for $\Diamond\varphi \in \Sigma$, if $\mathfrak{M}, v \models \varphi$, then $\mathfrak{M}, u \models \Diamond\varphi$. That is to say, it must be the case that for the representative worlds for the equivalence classes, the semantics of $\Diamond$ are preserved.

♦

Given a suitable choice of $\mathcal{R}^f$, the conditions imposed by frame validity on the relational structure are preserved. [3, pp. 79-81] However, some axiom schemata are resistant to this technique, such as one of the confluence axioms, $\Diamond\Box p \to \Box\Diamond p$, which is used in branching time logics [12, p. 108].

Therefore, one possible approach to modal filtration is to compute the subformula closed set for the finite modal formula φ for which we are trying to determine satisfiability, and to perform a filtration of a satisfying pointed model $\mathfrak{M}, w \models \varphi$ through this set. But note that this potentially gives us a model of size $\mid \mathcal{W} \mid = 2^{|\varphi|}$ where a smaller model would suffice, and does not address the problem of finding a minimal model with respect to the number of worlds (henceforth referred to as *minimal model* for brevity) [3, p.79]. We conclude that filtration is not intended for practical application in the minimization of models, but rather is a tool to demonstrate that an axiom characterization has the finite model property.

1.4 Existing Modal Decision Procedures

Contemporary researchers have summarized a broad set of tools for deciding satisfiability that are applicable to various members of the hierarchy of modal languages [4, 15].

In this section, we outline examples of existing decision procedures for modal logics with the finite model property and operating with a single modality, since that is the focus of this thesis. We restrict our attention to logics with no ability to construct new accessibility relations from existing ones using relational algebras.

1.4.1 Tableaux approach

One of the first techniques taught for generating semantic representations of modal concepts is that of constructing a modal tableaux [17, pp. 42-45]. The text written by Gasquet et. al in *Kripke's Worlds: An Introduction to Modal Logics via Tableaux* introduces LoTREC, which begins execution by applying tableau rules to the formula for which the user wishes to determine satisfiability; this approach breaks down the formula into its subformula components and creates nodes according to the semantic rules, adding edges between these nodes when required. Based on the prioritization of the application of rules, different strategies can lead to different saturated premodels, which affects overall performance of the procedure. [9, pp. 53-77]. Constraints on the accessibility relation may be enforced by adding additional potentially multi-part rules to the solution strategy using, for example, the `isLinked` keyword [9, pp. 87-109]. An open premodel is then transformed into a model with the desired constraints by performing an additional pass with the requisite rules.

1.4.2 SAT based approach

In his paper, "Building Decision Procedures for Modal Logics from Propositional Decision Procedures: The Case Study of Modal K(m)", Giunchiglia envisions modal reasoning as a form of nested propositional reasoning [10]. The advantage of this approach is that such propositional reasoning can be handled efficiently by a DPLL-like procedure (i.e. the standard approach for propositional satisfiability solvers), but requires that the modal formulas be converted to CNF; however, this can be done while preserving satisfiability [7, pp. 7].

Definition 1.4.1 (Atom (propositional or modal))**.** An *atom* is anything that cannot be decomposed propositionally. That is, any subformula whose main connective is modal (i.e. $\Box$ or $\Diamond$), or a subformula corresponding to a propositional atom. ⧫

Example 1.4.1. P_1 and $\Box(P_1 \vee \neg P_2)$ are atoms. ◀

Definition 1.4.2 (Top level atom)**.** A *top level atom* is an atom that doesn't appear under the scope of a box operator. Both of the atoms in Example 1.4.1 are top level atoms, P_1 being a propositional atom and $\Box(P_1 \vee \neg P_2)$ being a modal atom. However, P_1 as it appears within $\Box(P_1 \vee \neg P_2)$ is not a top level atom, as it is under the scope of a box operator. ⧫

Definition 1.4.3 (Truth assignment)**.** A *truth assignment* μ for a modal formula φ is a truth assignment to all top level atoms of φ, and can be written as:

$$\mu = \bigwedge_m \bigwedge_i \Box_m \alpha_{mi} \wedge \bigwedge_n \bigwedge_j \neg \Box_n \beta_{nj}$$

Where $\Box_m \alpha_{mi}$ are the positively appearing top-level modal atoms, and $\neg\Box_n \beta_{nj}$ appear negatively. ⧫

Definition 1.4.4 (Restricted truth assignment)**.** A **restricted truth assignment** μ^r is one which restricts assignments to atoms only involving $\Box_r$ as the main connective, and can be written as:

$$\mu^r = \bigwedge_i \Box_r \alpha_{ri} \wedge \bigwedge_j \neg \Box_r \beta_{ri}$$

⧫

The substance of Giunchiglia's paper is contained in these three subsequent theorems. With them, he constructs a recursive algorithm to decompose the problem of satisfaction for a modal formula.

Theorem 1.4.1. A modal formula φ is $K(m)$ satisfiable iff there exists a $K(m)$ satisfiable truth assignment μ such that $\mu \vDash \varphi$. ■

Theorem 1.4.2. A truth assignment μ is $K(m)$ satisfiable iff the restricted truth assignment μ^r is $K(m)$ satisfiable for all $\Box_r$. ■

Theorem 1.4.3. The restricted truth assignment μ^r is $K(m)$ satisfiable iff for every $\neg\Box_r\beta_{rj}$ occurring in μ^r, $\varphi^{rj} = \bigwedge_i \alpha_{ri} \wedge \neg\beta_{rj}$ is $K(m)$ satisfiable. ■

1.4.3 First-order Resolution-based approach

"Using Resolution for Testing Modal Satisfiability" by Hustadt and Schmidt details a syntactic approach for answering the question of decidability for a formula φ of multi-modal logic $\mathbb{K}$ with relational operators: the standard translation of φ to first-order logic is then pre-processed so as to be in negation normal form (NNF) through logical equivalence while maintaining the relationship with the original modal formula, and the first-order correspondents of the desired axiom schema are added as conjuncts to φ. This approach also encompasses stronger systems in which algebras over relations are allowed; however, these systems subsume the normal modal logics without relational algebras.

The expansion rules of deduction, deletion, and splitting are used to expand formulas so that resolution and factoring may be applied to form a proof by contradiction; the end of the resolution procedure yields the empty clause only if the formula is unsatisfiable [11, pp. 210-211].

The ability to polynomially simulate tableaux calculi using resolution suggests a means for producing models from this deductive procedure, where a satisfying pointed model is extracted from the set of positive ground clauses [11, p. 219].

1.4.4 Tree-automaton approach

Pan et. al's 2006 paper "BDD-based decision procedures for the modal logic K" describes how the implicit construction of a tree automaton can be used to decide whether a modal formula is satisfiable in the normal modal logic $\mathbb{K}$ [13]. These approaches use the tree-model property to attack the decision problem in $\mathbb{K}$, which ensures decidability [18]; additionally, the cost of the fixpoint computation for the non-emptiness test of the implicit tree automaton in stronger axiom systems is prohibitively high. Basic top-down and bottom-up approaches are given, which operate over *types* - subformula-closed subsets of the closure of the formula ψ for which we wish to answer the question of decidability. A satisfying pointed model is induced by the construction of these ψ-types: $\langle AP(\psi), A, \Delta, L\rangle$ where $AP(\psi)$ is the set of atoms of ψ, A is the set of types which corresponds to our set of worlds, Δ is the maximal accessibility relation defined as $\Delta(a, a') \Leftrightarrow \forall \Box\varphi' \in a \rightarrow \varphi' \in a'$, and L is the labelling (valuation map) where if $q \in a$, then $a \in L(q)$ [13, p. 174]. In the basic top-down approach, the set of all ψ-types is incrementally pared-down to exclude counterexamples to negated box formulas (i.e. $\forall \neg\Box\varphi \in a$ and $\forall b \in A$ such that $\Delta(a, b)$, $\varphi \in b$). The bottom-up approach builds upon an initial set which only contains propositional subformulas and positive box formulas (which may be vacuously satisfied without the requirement of a "witness" in A); the ψ-types a added in the update operation are such that $\forall \neg\Box\varphi \in a$, $\exists b \in A$ where $\Delta(a, b)$ and $\varphi \notin b$.

Pan et. al proceed to present optimizations to the basic approaches, such as replacing ψ-types with *particles*, and changing the criterion for the respective contraction and expansion of A in the top-down and bottom-up approaches to reduce redundant information [13, p. 178]; using level-based evaluation, which relies upon the finite tree-model property of $\mathbb{K}$ [13, p. 182]; and using modal pure-literal simplification with other equivalence- and satisfiability-preserving transformations to reduce the modal depth and overall length of the formula [13, p. 184].

Experimental results for the implementations of these basic techniques and subsequently optimized versions were obtained by running them on the modal benchmark formulas for the logic $\mathbb{K}$ from the TANCS2000 benchmark set. It was determined that applying simplification rules on the formulas and using a greedy approach led to the swiftest runtime and highest case completion [13, pp. 185, 199].

1.5 Addressing issues of Applicability

An examination of the literature demonstrates that some desirable features are absent from existing approaches, an account of which is given below. This supplies motivation for their implementation and proof of correctness provided by this thesis.

1.5.1 Constraining the Accessibility Relation

When applying a decision procedure to practical problems, it is necessary to consider the axiom characterization that best describes the way in which we wish modal operators to behave. The existing approaches outlined in § 1.4 all provide a framework for fixed axiom characterizations, ranging from the basic modal logic $\mathbb{K}$, to the higher systems of $\mathbb{T}$, $\mathbb{S}4$, and $\mathbb{S}5$. However, given the existence of the lattice of modal axiom characterizations with the finite model property, as well as how practical applications may call for arbitrary restrictions that do not have modal correspondents (e.g. the conditions of trichotomy or irreflexivity), there is the need for a procedure which enables users to enforce their desired conditions on the structure of the satisfying pointed model, or to show that a model satisfying these conditions does not exist.

1.5.2 Finding a Minimal Model

Following from the results of § 1.3.5, we know that there exists a finite upper bound on the number of worlds of a satisfying model for logics possessing the finite model property, namely $| \mathcal{W} | \leq 2^{|\varphi|}$, where $| \varphi |$ is the number of subformulas for formula φ [14, p. 52]. Existing semantic solvers focus on the question of the existence of satisfying models, but do not focus on the features of those models other than the properties of the accessibility relation, and only then in a restricted fashion.

1.5.3 Visualizations

Since the semantic techniques eschew the production of models, there is also a lack of concrete visualizations of the pointed models that satisfy a formula. This pursuit may seem superfluous, but from a pedagogical perspective, examining a satisfying model lends a deeper understanding of the system being worked in and of the formula itself.

Chapter 2

A Flexible Decision Procedure for Modal Logic

2.1 A new approach

The underlying principle of this decision and minimization procedure is that in order to determine the satisfiability of a formula φ, one must refer to its subformulas to reason about the relationships between their truth values. If one views a modal formula in terms of its syntactic tree, one can "peel away" layers of operators, considering a subformula as a main operator presiding over either one (for unary operators such as $\Box$ or $\neg$) or two (for binary operators such as $\wedge$) yet smaller subformulas.

The Enfragmo model expansion solver is the "engine" which requires the semantic rules of the language as well as the syntactic description of the formula to produce a pointed model with a certain number of worlds, or to indicate that no model of that size exists. The semantic rules for modal logic are encoded in the Enfragmo theory file, while a representation of the formula is contained within the problem instance file. An additional feature of the problem instance file is that the number of worlds for a potential satisfying model is pre-specified; therefore, this base procedure answers the more specific question of whether there exists a pointed model with a certain number of worlds for the formula encoded in the instance file. The output of this procedure is contained within an `XML` file, either giving the valuation map and relational structure representing the model produced, or returning `Unsatisfiable` if no model of the size specified in the problem instance file exists. This is outlined in Figure 2.1.

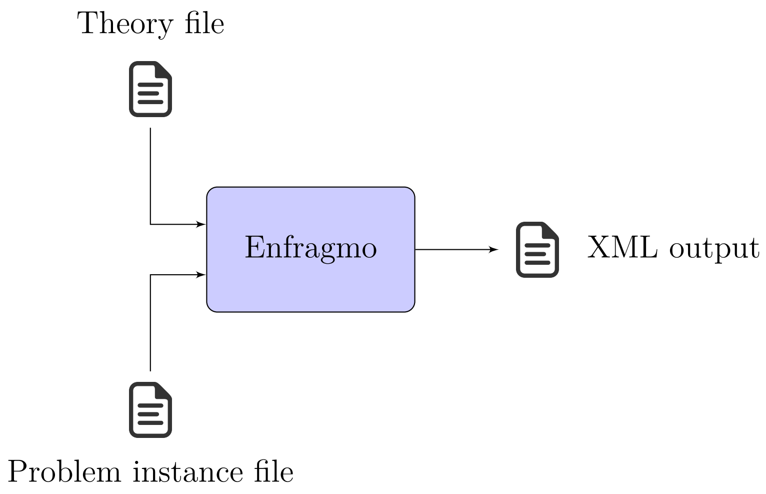


Figure 2.1: Basic solver employing Enfragmo.

This becomes the core of the decision and minimization procedure, which is described in Figure 2.2. The Modal Solver Suite first solves the problem of whether there exists a finite model for the formula by invoking Enfragmo with $\mid \mathcal{W} \mid = 1$, and then, if that is unsatisfiable, by repeatedly doubling the number of worlds and invoking Enfragmo until either some $\mid \mathcal{W} \mid = 2^k$ fails but $\mid \mathcal{W} \mid = 2^{k+1}$ succeeds, or the ceiling of $\mid \mathcal{W} \mid = 2^{|\varphi|}$ is reached. Subsequently, the user may optionally choose to find a satisfying pointed model with the minimum number of worlds; this is done by using a variant of binary search on the interval $(2^k, 2^{k+1}]$.

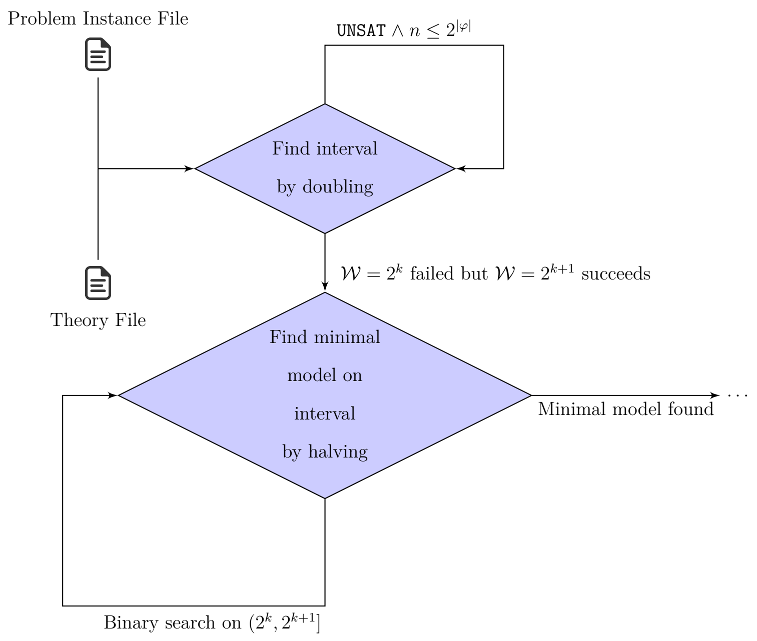

Problem Instance File
$\texttt{UNSAT} \wedge n \leq 2^{|\varphi|}$
Find interval by doubling
$\mathcal{W} = 2^k$ failed but $\mathcal{W} = 2^{k+1}$ succeeds
Theory File
Find minimal model on interval by halving
Minimal model found
...
Binary search on $(2^k, 2^{k+1}]$

Figure 2.2: Finding the minimal model

Figure 2.3 gives an idea of the overall architecture of the Modal Solver Suite, which will be covered in finer detail in the following sections. The user may either run the decision and minimization procedures on a single problem instance file, or on a directory of such files; in either case, the document sequencer sends one file at a time to the decision and minimization subroutine. Additionally, users may specify optional conditions for the accessibility relation; these are inserted into a copy of the theory file for the modal logic $\mathbb{K}$, and sent to the decision and minimization procedure. The resulting XML file will either indicate that no finite model exists, or will give the relational structure representing a pointed model. This is sent to the Kripke Model Constructor, which produces a single .dot representation of the pointed model and an .svg image for each file processed by the document sequencer.

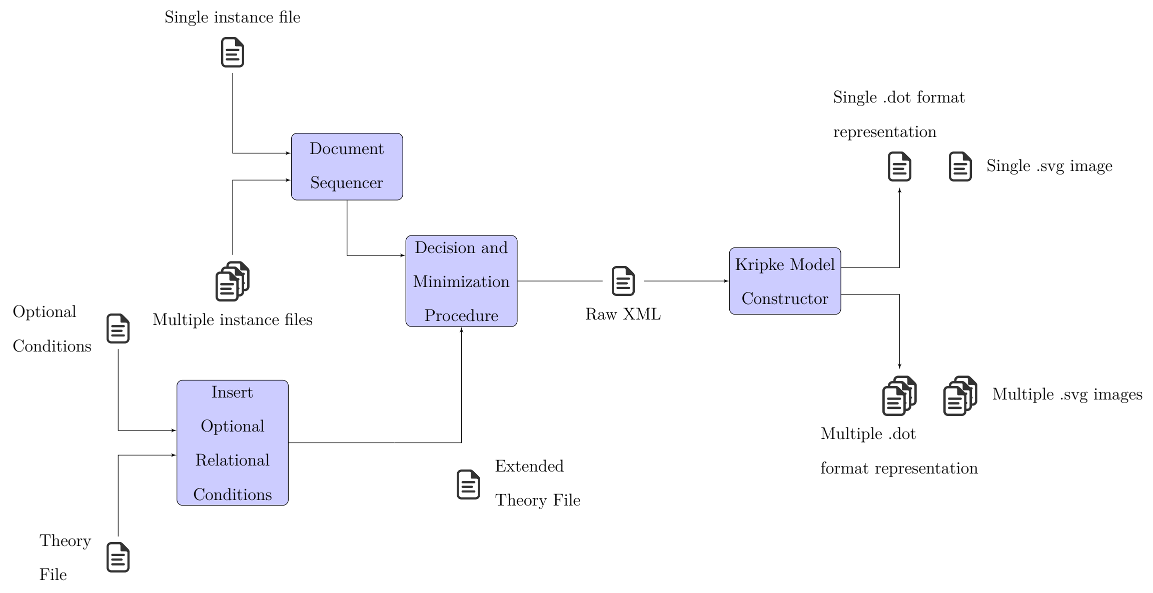


Figure 2.3: Architecture diagram of Modal Solver Suite.

2.1.1 Syntactic trees for modal formulas

When considering a modal formula φ, we construct its syntactic tree, which is unique. This is done by splitting the formula into its subformulas based on the main connective of the formula. The binary operators, $\wedge$ and $\vee$, dominate two subformulas, while the unary operators, $\neg$, $\Box$, and $\Diamond$, only dominate one subformula.

For this tree, the internal nodes are in correspondence with the subformulas of φ, and are labelled with the main connective of the subformula. Each of these internal nodes has children corresponding to the subformulas that are the arguments to the operator that labels the node. Leaf nodes represent the propositional atoms of the formula. Finally, the nodes of the tree are named according to a preorder depth-first search (DFS) traversal.

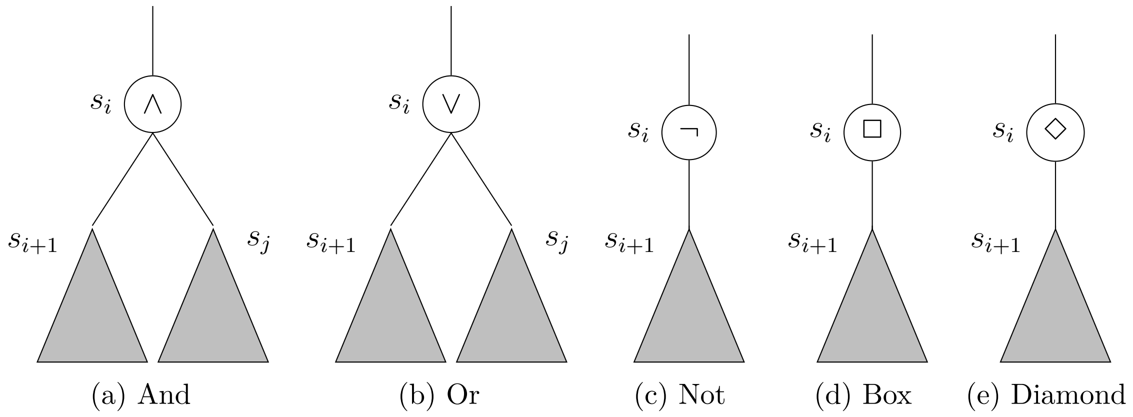
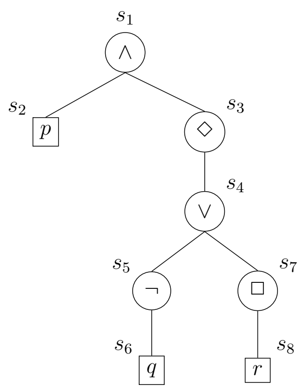


Figure 2.4: Constructing the syntactic tree of a modal formula φ

This labelling of the nodes of the tree gives us a means to refer to the subformulas by name when specifying the problem instance and when dictating the requirements of satisfaction of a modal formula.

Example 2.1.1. Consider formula $\varphi = (p \wedge \Diamond(\neg q \vee \Box r))$, which has the syntactic tree:



If we have a satisfying model for φ, then the following relationships between the truth value of the formula and its subformulas must hold:

$\wedge$	If a subformula whose main connective is conjunction holds at a world, then both conjuncts must hold at the world.
$\vee$	If a subformula whose main connective is disjunction holds at a world, then either the first, the second, or both disjuncts hold at that world.
$\neg$	If a subformula whose main connective is negation holds at a world, then the contents of the negation cannot be true at the world.
$\Box$	If a subformula whose main connective is a box operator is true at the current world, then the subformula dominated by the box must be true at all worlds accessible from the current world.
$\Diamond$	If a subformula whose main connective is a diamond operator holds at the current world, then the subformula dominated by the diamond must be true at at least one world that is accessible from the current world.

2.1.2 Enfragmo implementation

The method is to encode the formula as an Enfragmo problem instance, with the theory file expressing in first-order logic what it means for subformulas to be true at a world in a model, based on the main connective of the subformula and the truth values of the sub-subformulas.

Problem Instance Files

The problem instance file describes a modal formula in terms of its syntax tree, with labels assigned to subformulas as detailed in Figure 2.4. First, the number of subformulas of φ is specified. Then we give a bound on the number of worlds the model produced by Enfragmo should have.

The first predicate, '`Atom`', represents the propositional atoms; these are our most basic operands, the subformulas labelling the leaves of the syntax tree.

The subsequent list of predicates represent the operators in the grammar, where the list of tuples following each predicate declaration has either two or three arguments. The first argument is the label of the subformula for which the operator is main connective, while the second (unary operator) and third (binary operator) arguments are subformula labels, which appear either as a singleton under `Atom`, or as the first argument for a tuple satisfying another predicate.

The final predicate '`SameAtom`' is related to the first, in that it dictates which subformulas at the leaf-level of the syntax tree correspond to the same propositional atom.

```
TYPE  Subformula [ 1.. n]
TYPE World [1..x]
PREDICATE Atom
(a)
...
PREDICATE <OperatorName>
(b, c)
...
PREDICATE SameAtom
(a, d)
```

Figure 2.5: The contents of a problem instance file

The problem instance file describes a modal formula in terms of its syntactic tree. First, the number of subformulas are specified: the formula φ is a subformula of itself, and is therefore the first subformula, with the remaining subformulas arising from the decomposition based on the main connective. The propositional atoms comprise the remaining subformulas.

Then, assuming that we are working with conditions on the accessibility relation that allow finite models, we give a bound on the number of worlds the desired model should have: if the formula φ does have a satisfying pointed model, then we know it will have a pointed model with less than or equal to $2^{|\varphi|}$ worlds; this means that we can halt our search when we reach this bound in axiom systems with the finite model property.

Finally, we have the list of predicates. Given the preorder DFS labelling of the nodes of the syntactic tree, we indicate which subformulas are atoms, and then show the relationships between other subformulas involved with each other via various operators.

Example 2.1.2. Continuing from Example 2.1.1, this is the problem instance file constructed for the formula $\varphi = (p \wedge \Diamond(\neg q \vee \Box r))$:

```
TYPE  Subformula [ 1.. 8]
TYPE World [1..x]
PREDICATE Atom
(2)
(6)
(8)
PREDICATE And
(1, 2, 3)
PREDICATE Diamond
(3, 4)
PREDICATE Or
(4, 5, 7)
PREDICATE Not
(5,6)
PREDICATE Box
(7,8)
```

◀

Theory File

Keeping in mind the formal semantics for modal logic given in § 1.3.1, new predicates are created to capture the ideas inherent in Kripke semantics: the structural predicates, $TrueAt$, and *Accessible*.

The structural predicates, such as *And* and Box, are used to indicate that the first argument is a subformula whose main connective is that operator, with the remaining arguments being the subformulas bound by that operator, whereas *SameAtom* is used to indicate that two subformulas actually refer to the same atom. The predicate $TrueAt$ is used to indicate that a subformula is true at a world. Finally, the predicate *Accessible* is an encoding of the accessibility relation that we are aiming to build as part of the model.

The formulas that we used to represent these constraints are as follows:

$\bot$	$\forall s \in Subformula, w \in World\ (\mathbf{Falsum}(s) \rightarrow \neg(TrueAt(s, w)))$
$\wedge$	$\forall s_1, s_2, s_3 \in Subformula\ (\mathbf{And}(s_1, s_2, s_3) \rightarrow$
	$\forall w \in World\ (TrueAt(s_1, w) \leftrightarrow (TrueAt(s_2, w) \wedge TrueAt(s_3, w))))$
$\vee$	$\forall s_1, s_2, s_3 \in Subformula\ (\mathbf{Or}(s_1, s_2, s_3) \rightarrow$
	$\forall w \in World\ (TrueAt(s_1, w) \leftrightarrow (TrueAt(s_2, w) \vee TrueAt(s_3, w))))$
$\neg$	$\forall s_1, s_2 \in Subformula\ (\mathbf{Not}(s_1, s_2) \rightarrow$
	$\forall w \in World\ (TrueAt(s_1, w) \leftrightarrow \neg TrueAt(s_2, w)))$
$\rightarrow$	$\forall s_1, s_2, s_3 \in Subformula\ (\mathbf{Implication}(s_1, s_2, s_3) \rightarrow$
	$\forall w \in World\ (TrueAt(s_1, w) \leftrightarrow (TrueAt(s_2, w) \rightarrow TrueAt(s_3, w))))$
$\leftrightarrow$	$\forall s_1, s_2, s_3 \in Subformula\ (\mathbf{Biconditional}(s_1, s_2, s_3) \rightarrow$
	$\forall w \in World\ (TrueAt(s_1, w) \leftrightarrow (TrueAt(s_2, w) \leftrightarrow TrueAt(s3, w))))$
$\Box$	$\forall s_1, s_2 \in Subformula\ (\mathbf{Box}(s_1, s_2) \rightarrow$
	$\forall w1 \in World\ (TrueAt(s_1, w1)$
	$\leftrightarrow \forall w2 \in World$ s.t. $(Accessible(w_1, w_2) \rightarrow TrueAt(s_2, w_2))))$

$\Diamond$ $\forall s_1, s_2 \in Subformula\ (\mathbf{Diamond}(s_1, s_2) \rightarrow$

$\forall w_1 \in World\ (TrueAt(s_1, w_1)$

$\leftrightarrow \exists w_2 \in World$ s.t. $(Accessible(w_1, w_2) \wedge TrueAt(s_2, w_2))))$

These are encoded in the Enfragmo theory file in the following way:

```
⊥   !  s:Subformula w:World :
     ((Falsum (s) ) =>
     ( ~ ( TrueAt (s,w) )));
∧   !  s1:Subformula s2:Subformula s3:Subformula :
    ( And (s1,s2,s3) =>
    ( !  w:World :  ( TrueAt (s1,w) <=> ( TrueAt (s2,w) & TrueAt (s3,w) ))));
∨   !  s1:Subformula s2:Subformula s3:Subformula :
    ( Or (s1,s2,s3) =>
    ( !  w:World :  ( TrueAt (s1,w) <=> ( TrueAt (s2,w) | TrueAt (s3,w)))));
¬   !  s1:Subformula s2:Subformula :
     ( Not (s1,s2) =>
    ( !  w:World :( TrueAt (s1,w) <=> ~ TrueAt (s2,w))));
→   !  s1:Subformula s2:Subformula s3:Subformula :
    ( Implication (s1,s2,s3) =>
     ( !  w:World :  ( TrueAt (s1,w) <=> ( TrueAt (s2,w) => TrueAt (s3,w)))));
↔   !  s1:Subformula s2:Subformula s3:Subformula :
    ( Biconditional (s1,s2,s3) =>
     ( !  w:World :  ( TrueAt (s1,w) <=> ( TrueAt (s2,w) <=> TrueAt (s3,w)))));
□   !  s1:Subformula s2:Subformula :
    ( Box (s1,s2) =>
    ( !  w1:World :  TrueAt (s1,w1) <=> !  w2:World :  ( Accessible (w1,w2) =>
    TrueAt (s2,w2))));
◇   !  s1:Subformula s2:Subformula :
    ( Diamond (s1,s2) => ( !  w1:World :  TrueAt (s1,w1) <=>
    ?  w2:World :  ( Accessible (w1,w2) & TrueAt (s2,w2))));
```

Note the grammar of the Enfragmo specification language: `!` is used for universal quantification, `?` for existential, `&` for $\wedge$, `|` for $\vee$, `~` for $\neg$, `=>` for $\rightarrow$, and `<=>` for $\leftrightarrow$. The use of these symbols give the structural predicates their intended meaning.

The following two clauses are necessary to ensure that for any world in the model, subformulas corresponding to the same atom must share the same truth value at the same world.

```
! s1:Subformula s2:Subformula w:World :
( SameAtom (s1, s2)  =>
( TrueAt (s1, w) <=> TrueAt (s2, w) ) );


! s1:Subformula s3:Subformula w:World : ( ? s2:Subformula :
( ( Not (s2,s3) & SameAtom (s1, s3) )  =>
( TrueAt (s1, w) <=> ( ~ TrueAt (s2, w) & TrueAt(s3, w) ) ) ) );
```

The first line simply reads that if two subformulas s1 and s2 correspond to the same atom, then they must share the same truth value at world w.

The second line reads "At world `w`, if `s1` and `s3` correspond to the same atom and `s3` is under negation in subformula `s2`, then `s1` is true at `w` if and only if `s2` is false at `w` and `s3` is true at `w`". The truth values of `s1` and `s3` must correspond when evaluated at the same world.

2.2 The Modal Solver Suite

When used in isolation, sending the theory file and problem instance to Enfragmo gives us the solution to an individual problem: "does φ have a satisfying pointed model of size k?". This extends to the general decision problem of whether there exists a finite pointed model for a formula within a given axiom system. Finally, one might wish to produce a minimal model for the formula with the same frame conditions. There is also the issue of ease of use: how can one solve the same type of problem for a sequence of modal formulas, and then obtain human-readable output?

To accomplish these goals, supplementary packages in Python were written (available on GitHub; see Appendix A), which enable the core procedure to enforce different user-specified constraints on the accessibility relation, as well as to tackle potentially large numbers of problem instances in sequence. The Modal Solver Suite consists of a driving procedure, which repeatedly invokes the decision procedure, followed by the minimization procedure for either a single or multiple problem instance files with the option to constrain the accessibility relation. Finally, the Kripke model constructor subroutine is run to build human-readable output.

2.2.1 Decision Procedure

The number of worlds required for a satisfying model is not always apparent following inspection of the formula, especially for formulas with high modal depth. Therefore, a mechanism is necessary to first determine if a formula possesses a finite model with $1 \leq |\mathcal{W}| \leq 2^{|\varphi|}$, where $|\varphi|$ is the number of subformulas of φ.

The Modal Solver Suite repeatedly invokes Enfragmo with $|\mathcal{W}| = 2^i,\ i \in [0, |\varphi|]$, until either some $|\mathcal{W}| = 2^k$ fails but $|\mathcal{W}| = 2^{k+1}$ succeeds, or the theoretical maximum of $|\mathcal{W}| = 2^{|\varphi|}$ is reached and no satisfying pointed model is found. This is done by changing the second line of the problem instance file, `TYPE World [1..x]`, where the next highest power of two to be tested replaces the value at placeholder `x`.

The variable `isUnSAT` keeps track of whether or not we have found a pointed model with $|\mathcal{W}| =$ `currNumWorld`. At every loop iteration, we make sure a model still has not been found, and that we are not testing beyond the theoretical maximum, before executing the loop body. Within the loop body, we attempt to produce a model with $|\mathcal{W}| =$ `currNumWorld`. If successful, a model is produced and output to file, and `isUnSAT` is set to `FALSE`; the next loop condition test will fail, and we will have a satisfying pointed model for φ. If unsuccessful, the output must contain a tag labelled `UNSATISFIABLE`, and so `isUnSAT` is set to `TRUE` and the variable `currNumWorld` is doubled for the next iteration of the loop.

Algorithm 1 Decision procedure via doubling

```
1:  procedure DECISION PROCEDURE($startingNumWorlds$, $maxNumWorlds$)
2:      $isUnSAT \leftarrow True$
3:      $currNumWorld \leftarrow startingNumWorlds$
4:      while $isUnSAT$ and $currNumWorld \leq maxNumWorlds$ do
5:          $isUnSAT \leftarrow makeModel(currNumWorld)$
6:          if $isUnSAT$ then
7:              $currNumWorld \leftarrow currNumWorld * 2$        ▷ At most $\mid \varphi \mid$ iterations.
8:          end if
9:      end while
10:     return $isUnSAT$
11: end procedure
```

2.2.2 Multiple axiom characterizations

Suppose we wish to find whether a formula has a model which adheres to a specific axiom characterization. This means that we wish to add requirements to the accessibility relation, which is done by adding clauses to the theory file of the specification. Currently, this is done by adding the first-order correspondent of an axiom schema - provided that the axiom possesses a first-order correspondent - or by adding first-order formulas for which there exists no corresponding modal axiom. The latter should be done with caution, for the proof of correctness in Chapter 3 only guarantees invariance for first-order formulas which are the correspondent of a modal axiom schema.

For example, if we wanted to represent the conditions imposed on $\mathcal{R}$ by some common axiom characterizations, such as:

K	$\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$	(Holds in all normal frames)
T	$\Box\varphi \to \varphi$	Reflexivity
$S4$	$\Box\varphi \to \Box\Box\varphi$	Transitivity
$S5$	$\Diamond\varphi \to \Box\Diamond\varphi$	Euclidity

With their equivalent first-order frame conditions:

T	$\forall w \in W, R(w, w)$
$S4$	$T + \forall w, v, u \in W, R(w, v) \wedge R(v, u) \Rightarrow R(w, u)$
$S5$	$S4 + \forall w, v, u \in W, R(w, u) \wedge R(w, v) \Rightarrow R(u, v)$

We would add the following to our Enfragmo theory file:

T	!w: World : Accessible(w, w)
4	!w: World v: World u: World : ((Accessible(w,v) & Accessible(v,u)) => Accessible(w,u))
5	!w: World v: World u: World : ((Accessible(w,u) & Accessible(w,v)) => Accessible(u,v))

These clauses can be modified to enforce restrictions on the accessibility relation as well, dictating, for example, that the relation should *not* be reflexive by including the clause `?w:  World :  ~ Accessible (w, w)`. Note that this is not the same as dictating that the accessibility relation be irreflexive, as in Example 1.3.4:

```
!w1:  World w2 :  World :  ( Accessible (w1, w2) => ( ~ ( w1=w2 ) ) )
```

This might be applicable when representing the knowledge of agents, thereby restricting their capabilities. If we do not want an agent capable of positive or negative introspection, we would negate the first-order correspondents for those axiom schemas and add them as conditions on the accessibility relation.

A key feature of the Modal Solver Suite is the ability for users to choose a file containing the encoding of required features of the accessibility relation in the Enfragmo specification language, which are inserted into a new theory file at the start of a run. If no file is specified, then satisfiability is determined in modal logic $\mathbb{K}$. As mentioned in § 1.3.2, no recursively enumerable procedure can exist for determining the first-order frame correspondent of a modal axiom in general. Therefore, any translation procedure to obtain a first-order correspondent from even modal axioms of a form that guarantees the existence of such a correspondent is outside of the scope of this thesis; thus, users must provide their own translation, should one exist. For example, despite the existence of a translation mechanism to determine the first-order correspondent of a Sahlqvist formula as outlined in Blackburn [3, pp. 157-167], no implementation is given. We feel that this is not a shortcoming, since we allow users to enforce arbitrary first-order conditions on the relation.

2.2.3 Finding a minimal model

We extend the concept of a binary search to the problem of finding a minimal model for a formula φ with respect to the number of worlds, while still respecting the desired frame conditions. This avoids the shortcomings of the method of filtration, which relies on the creative construction of and proof of correctness for a specialized filtrated relation. If an axiom characterization is resistant to definition in this way, then the method fails. Furthermore, given a complete axiom schema comprised of multiple axioms, deriving a relation $\mathcal{R}^f$ becomes increasingly involved. Another problem is that given a poor choice for a starting model, the equivalence classes are not guaranteed to be maximal.

In § 1.5.2, we outlined the theory of finite models, which brought to light the existence of an upper bound on the number of worlds for a formula in an axiom system with the finite model property. Recall that this bound is in terms of the number of subformulas of φ, namely `maxWorlds` $= 2^{|\varphi|}$ where $\mid \varphi \mid$ is the notation for the number of subformulas of φ. Conveniently, the problem instance file for a formula contains the total number of subformulas without additional processing. Here, we describe the approach implemented to find a minimal satisfying pointed model for a formula.

Algorithm 2 Minimization procedure using a variant of binary search

```
1: procedure MINIMIZATION PROCEDURE($lower$, $upper$)
2:     while $lower \leq upper$ do
3:         $midpoint \leftarrow \lfloor (upper + lower)/2 \rfloor$
4:         $isUnSAT \leftarrow makeModel(midpoint)$
5:         if isUnSAT then $lowerBound \leftarrow midpoint + 1$
6:         else
7:             $upper \leftarrow midpoint - 1$
8:             $found \leftarrow midpoint$
9:         end if
10:    end while
11:    $makeModel(found)$
12: end procedure
```

Note that we must use the floor function to calculate `midpoint` due to the proof of Chapter 3: if $\lfloor$(`upper + lower`)$/2\rfloor$ was the actual lower bound, then $\lceil$(`upper + lower`)$/2\rceil$ will also yield a satisfying pointed model, which would not be minimal. We see if there exists a model with $\mid \mathcal{W} \mid =$ `midpoint`. If not, then we exclude `midpoint` from the next interval of consideration and set the new `lower = midpoint + 1`. If such a model does exist, we first save `midpoint` as our current minimal value in `found`, and then set `upper = midpoint - 1`, diving into the lower part of the interval; `found` will eventually contain the minimal value for $\mid \mathcal{W} \mid$.

Since the minimization procedure is performed on an interval of size $2^{k+1} - (2^k + 1) = 2^k - 1$, at most k tests are performed to find a minimal model.

It is especially important to realize that these techniques require that the desired axiom characterization possesses the finite model property; if this is not the case, then there is no guarantee that the procedure will find a model before halting at the upper bound of $|\,\mathcal{W}\,| = 2^{|\varphi|}$.

2.2.4 Visualization of Kripke Structures

In order to make the results produced by Enfragmo easier to work with, it is necessary to first extract the relevant information from the raw XML file and to encode it in a more accessible format. The Graphviz .dot graph representation language is used due to the number of programs that can process this format, as well as how accessible it is to human comprehension.

Although constructing a visual representation of this data is desirable, it is not necessarily feasible for larger structures; having a succinct representation of the relational structure is a major step in improving usability, which facilitates the creation of visualizations for smaller structures.

First, we determine if the formula was satisfiable. If so, we parse the output to obtain the valuation map and accessibility relation; if not, then no .dot format representation is created. After reading the valuation and accessibility relation, we extract the number of worlds in the model from the problem instance file. Finally, a Graphviz Digraph object is created, where we give the object the number of nodes (the worlds), set the valuation map (a dictionary object), and set the accessibility relation (another dictionary object). When we display the Kripke structure, the source is written to a file, and then the graph is rendered; the code may be modified so that the rendering phase may be performed by a different processing engine.

2.2.5 Automation of multiple runs

Users may either specify a single problem instance corresponding to a formula for which the problem of decidability in the given axiom characterization must be answered, or, if no filename is given, to run the procedure on the entire directory structure of the current folder. This was implemented using the Python `os.walk` method with keyname parameter `topdown=False` so that the original directory structure would be respected when generating the solutions [1].

Chapter 3

Proof of Correctness

3.1 Motivation and Statement of the Theorem

This chapter provides validation for the application of a doubling technique to find a satisfying pointed model, and for the application of a variant of binary search to find a minimal model after a satisfying model has been found. The decision procedure first determines that no satisfying pointed model for φ exists with $\mid \mathcal{W} \mid \leq 2^k$, but that a model exists with $\mid \mathcal{W} \mid = 2^{k+1}$. We then know that a minimal satisfying pointed model must have $\mid \mathcal{W} \mid \in \left(2^k, 2^{k+1}\right]$. The $O(log\ n)$ complexity of a variant of binary search is attractive in finding a minimal pointed model $\mathfrak{M}$ with $\mid \mathcal{W} \mid = \ell$, but requires some notion of continuity on the interval of application.

Theorem 3.1.1 (Monotonicity of Modal Satisfaction)**.** For a modal formula φ and axiom schema ψ, if there exists a satisfying pointed model $\mathfrak{M}, w$ for φ where $\mid \mathcal{W} \mid = n > 0$, written $\mathfrak{M}, w \models \varphi$, and axiom schema ψ is valid on the underlying frame $\mathfrak{F}$ of $\mathfrak{M}$, written $\mathfrak{F} \models \psi$, then there exists a pointed model $\mathfrak{M}', w'$ with $\mid \mathcal{W}' \mid = n+1$ such that $\mathfrak{M}', w' \models \varphi$ and $\mathfrak{F}' \models \psi$. Furthermore, there exists no modal formula ψ' such that $\mathfrak{F}' \models \psi'$ and $\mathfrak{F} \not\models \psi'$. ■

This theorem guarantees the existence of models of each size $n + i$ for $n, i \in \mathbb{Z}^+$, given the existence of a satisfying pointed model $\mathfrak{M}, w$ for φ with $\mid \mathcal{W} \mid = n$, with this inflation due to the addition of i bisimilar worlds. Additionally, this proof ensures that modal axioms (and therefore their first-order correspondents) hold in the larger model if and only if they hold in the original model. The validity of arbitrary first-order conditions is not invariant under bounded morphisms, and so only the validity of α which are the first-order correspondents of axiom schema ψ are preserved under this construction.

Note that this proof is not a guarantee that the model produced by the variant of binary search presented in Chapter 2 is bisimilar to the model that Enfragmo found with $\mid \mathcal{W} \mid = 2^{k+1}$; rather, it simply serves to show that there are no "gaps" in the interval, thereby proving that a variant of binary search is applicable.

Unlike the use of bisimulation contraction, this procedure is guaranteed to find some minimal model, ignoring the possibly superfluous relational links that the Enfragmo solver adds, since it doesn't hinge upon some requirement on the internal structure of the model (i.e. non-trivial equivalence classes of worlds). Furthermore, the method of bisimulation contraction requires a computationally expensive fixpoint calculation [6]. As discussed in § 1.3.5 Definition 1.3.11, the methods of selection and filtration also fail, because in the former, the intermediate steps of producing a tree-like model may obliterate symmetric or reflexive edges, and in the latter, some axiom characterizations remain resistant to defining an appropriate $\mathcal{R}^f$. Additionally, unlike filtration, the doubling and halving procedure is guaranteed to be applicable to any modally expressible axiom characterization without requiring specialized machinery for each characterization.

The proof of this theorem appeals to the definitions of bounded morphisms and disjoint unions given in § 1.3.4, and the subsequent theorems regarding their truth- and validity-preserving results. In order to prove that not only $\mathfrak{M}, w \models \varphi \Leftrightarrow \mathfrak{M}', \hat{w} \models \varphi$, where $\hat{w}$ is the copy of w in the extended model $\mathfrak{M}'$, but that all frame conditions imposed by axioms ψ are preserved through equivalent frame validity $\mathfrak{F} \models \psi \Leftrightarrow \mathfrak{F}' \models \psi$, we must produce the appropriate constructions. First, we construct the surjective bounded morphism $f : \mathfrak{F}' \to \mathfrak{F}$ to obtain $\mathfrak{F}' \models \psi \Rightarrow \mathfrak{F} \models \psi$. Then, we construct the disjoint union $\mathfrak{F} \uplus \mathfrak{F}$, for which we have $\mathfrak{F} \models \psi \Leftrightarrow \mathfrak{F} \uplus \mathfrak{F} \models \psi$, so that we might construct the surjective bounded morphism $g : \mathfrak{F} \uplus \mathfrak{F} \to \mathfrak{F}'$ to demonstrate that $\mathfrak{F} \uplus \mathfrak{F} \models \psi \Rightarrow \mathfrak{F}' \models \psi$, and therefore that $\mathfrak{F} \models \psi \Rightarrow \mathfrak{F}' \models \psi$. This completes both directions of the biconditional $\mathfrak{F} \models \psi \Leftrightarrow \mathfrak{F}' \models \psi$, and concludes the proof of the theorem.

3.2 Constructing the larger model $\mathfrak{M}'$

Let $\mathfrak{M}, w$ be a pointed model with $\mathfrak{M} = \langle \mathcal{W},\ \mathcal{R},\ V^{\mathfrak{M}} \rangle$ and $\mid \mathcal{W} \mid =\ n$.

Definition 3.2.1 (Extended model $\mathfrak{M}'$)**.** To construct model $\mathfrak{M}'$ with $\mid \mathcal{W}' \mid = n+1$, create bisimilar world $w' \cong w$ with $w' \neq \hat{w}$, the copy of w in $\mathfrak{M}'$, so that $\mathcal{W}' = \left\{\hat{w} \mid w \in \mathcal{W}\right\} \cup \left\{w'\right\}$.

The valuation map $\mathcal{V}$ is extended to $\mathcal{V}'$ as follows:

- $\forall s \in \mathcal{W},\ \forall p \in \mathcal{A}\quad \mathcal{V}'(\hat{s},\ p) = \mathcal{V}(s,\ p)$
- For $w' \in \mathcal{W}',\ \forall p \in \mathcal{A}\quad \mathcal{V}'(w',\ p) = \mathcal{V}(w,\ p)$

The accessibility relation $\mathcal{R}$ is extended so that $\mathcal{R} \subset \mathcal{R}'$:

$$\begin{aligned} \mathcal{R}' = &\Big\{(\hat{x}, \hat{y}) \mid (x, y) \in \mathcal{R}\Big\} && (3.1)\\ &\cup \Big\{\langle w',\ \hat{x}\rangle \mid \langle w,\ x\rangle \in \mathcal{R}\Big\} && (3.2)\\ &\cup \Big\{\langle \hat{x},\ w'\rangle \mid \langle x,\ w\rangle \in \mathcal{R}\Big\} && (3.3)\\ &\cup \Big\{\langle w',\ w'\rangle \mid \langle w,\ w\rangle \in \mathcal{R}\Big\} && (3.4) \end{aligned}$$

♦

Recall from § 1.3.2 that model $\mathfrak{M}$ has frame $\mathfrak{F} = \langle \mathcal{W}, \mathcal{R}\rangle$ and $\mathfrak{M}'$ has frame $\mathfrak{F}' = \langle \mathcal{W}', \mathcal{R}'\rangle$, where we consider the set of worlds and accessibility relation in isolation from the valuation map.

3.3 Construction of the bounded morphisms

3.3.1 Bounded morphism from $\mathfrak{F}'$ to $\mathfrak{F}$

Definition 3.3.1 (Bounded morphism f)**.** Define bounded morphism $f : \mathfrak{F}' \to \mathfrak{F}$ as follows:

$$f(y) = \left\{ \begin{array}{ll} x & \text{if } y = \hat{x} \text{ for } x \in \mathcal{W} \\ w & \text{if } y = w' \end{array} \right\}$$

♦

We now show that f satisfies properties 1 to 3 of Definition 1.3.7.

1. To prove that f is surjective, let $b \in \mathcal{W}$ be arbitrary:

   (a) $b \neq w$: $\wedge$ gives us a unique image in $\mathcal{W}'$: $f(\hat{b}) = b$.

   (b) $b = w$: Both $f(\hat{w}) = b$ and $f(w') = b$.

Therefore, each member of $\mathcal{W}$ is mapped to at least once under f, so f is surjective. ■

2. Suppose $a\mathcal{R}'b$ in $\mathfrak{F}'$. We must show that steps taken in the domain are mirrored by steps taken in the codomain: $a\mathcal{R}'b \Rightarrow f(a)\mathcal{R}f(b)$.

   (a) If $a \neq w'$, $b \neq w'$, then a and b must be the respective unique images of some elements in $\mathcal{W}$ under $\wedge$: $a = \hat{y}$, $b = \hat{z}$. So if $\hat{y}\mathcal{R}'\hat{z}$ in $\mathfrak{F}'$, then by Definition line 3.1, it must have been the case that $y\mathcal{R}z$ in $\mathfrak{F}$. By Definition 3.3.1, $f(\hat{y}) = y$ and $f(\hat{z}) = z$. ■

   (b) If $a = w'$, $b \neq w'$, then b must be the image of an element in $\mathcal{W}$ under $\wedge$: $b = \hat{z}$. If $w'\mathcal{R}'\hat{z}$ in $\mathfrak{F}'$, then by Definition line 3.2, it must have been the case that $w\mathcal{R}z$ in $\mathfrak{F}$. By Definition 3.3.1, $f(w') = w$ and $f(\hat{z}) = z$. ■

   (c) If $a \neq w'$, $b = w'$, then a must be the image of an element in $\mathcal{W}$ under $\wedge$: $a = \hat{y}$. If $\hat{y}\mathcal{R}'w'$ in $\mathfrak{F}'$, then by Definition line 3.3, it must have been the case that $y\mathcal{R}w$ in $\mathfrak{F}$. By Definition 3.3.1, $f(\hat{y}) = y$ and $f(w') = w$. ■

   (d) If $a = w'$, $b = w'$, then $w'\mathcal{R}'w'$ in $\mathfrak{F}'$ must mean that $w\mathcal{R}w$ in $\mathfrak{F}$, by Definition line 3.4. Definition 3.3.1 dictates that $f(w') = w$. ■

3. Suppose $f(a)\mathcal{R}z$ in $\mathfrak{F}$. We must show that z will be the image of some element of $\mathcal{W}'$ under f: $f(a)\mathcal{R}z \Rightarrow \exists x \in \mathcal{W}'$ such that $a\mathcal{R}'x \wedge f(x) = z$.

   (a) If $a \neq w'$, then a must be the image of some $y \in \mathcal{W}$: $a = \hat{y}$. Therefore, $f(a) = f(\hat{y}) = y$ by Definition 3.3.1.

      i. If $z \neq w$, then z is some element of $\mathcal{W}$ with image $\hat{z} \in \mathcal{W}'$ under $\wedge$. So if $y\mathcal{R}z$ in $\mathfrak{F}$, by Definition line 3.1 we must have $\hat{y}\mathcal{R}'\hat{z}$. Definition 3.3.1 gives us $f(\hat{z}) = z$. ■

ii. If $z = w$, then $y\mathcal{R}w$ in $\mathfrak{F}$ means that $\hat{y}\mathcal{R}'w'$ in $\mathfrak{F}'$ by Definition line 3.3. Definition 3.3.1 gives us $f(w') = w$. ■

(b) If $a = w'$, then by Definition 3.3.1, we have $f(w') = w$.

i. If $z \neq w$, then z is some element of $\mathcal{W}$ with image $\hat{z} \in \mathcal{W}'$ under $\wedge$. So if $w\mathcal{R}z$ in $\mathfrak{F}$, by Definition line 3.2 we must have $w'\mathcal{R}'\hat{z}$. Definition 3.3.1 gives us $f(\hat{z}) = z$,. ■

ii. If $z = w$, then $w\mathcal{R}w$ in $\mathfrak{F}$ means that $w'\mathcal{R}'w'$ in $\mathfrak{F}'$ by Definition line 3.4. Definition 3.3.1 gives us $f(w') = w$. ■

Since $f : \mathfrak{F}' \to \mathfrak{F}$ is a surjective bounded morphism from $\mathfrak{F}'$ to $\mathfrak{F}$, we can conclude by Corollary 16 of the *Handbook of Modal Logic* [4, p. 259] that condition iv gives us:

$$\text{If } f \text{ is an onto bounded morphism, } \mathfrak{F}' \models \psi \Rightarrow \mathfrak{F} \models \psi$$

Hence one direction is complete [4, p. 259]. □

3.3.2 Validity-preserving expansion to $\mathfrak{F} \uplus \mathfrak{F}$

Definition 3.3.2 ($\mathfrak{F}\uplus\mathfrak{F}$). Recalling the theory presented in § 1.3.4, if we have frame $\mathfrak{F} = \langle \mathcal{W}, \mathcal{R} \rangle$, the intermediate structure $\mathfrak{F} \uplus \mathfrak{F}$ is constructed as follows:

$$\mathcal{W}^{\uplus} = \mathcal{W} \times \{0\} \,\cup\, \mathcal{W} \times \{1\} = \Big\{(x, i) \mid x \in \mathcal{W}, i \in \{0, 1\}\Big\} \tag{3.5}$$

$$\mathcal{R}^{\uplus} = \mathcal{R}^0 \cup \mathcal{R}^1 = \Big\{\Big((x, i), (y, i)\Big) \mid (x, y) \in \mathcal{R}, i \in \{0, 1\}\Big\} \tag{3.6}$$

♦

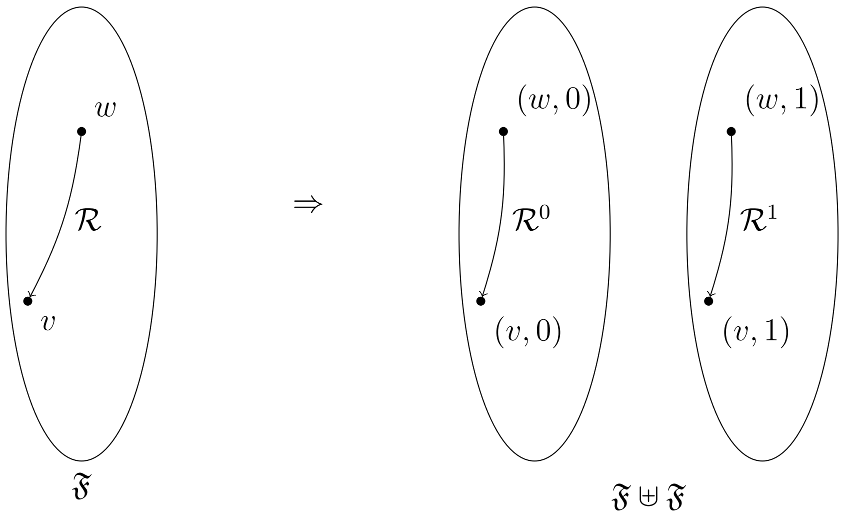


Figure 3.1: Construction of $\mathfrak{F} \uplus \mathfrak{F}$

The necessity for the construction of $\mathfrak{F} \uplus \mathfrak{F}$ is clear: we wish to construct a surjective bounded morphism onto $\mathfrak{F}'$, but the size of the domain of $\mathfrak{F}$ is insufficient (i.e. through the addition of a single world to the domain of $\mathfrak{F}$, namely w'). This choice for intermediate structure was made namely for its property of validity preservation. From Proposition 26 in *The Handbook of Modal Logic*[4, p. 262], part iv gives us:

$$\uplus_{i \in I} \mathfrak{F}^i \models \psi \Leftrightarrow \forall i \in I \;\; \mathfrak{F}^i \models \psi$$

The validity of axiom schema ψ is invariant under construction of the disjoint union, given that ψ is valid in all components; this means that since we are constructing a disjoint union of $\mathfrak{F}$ with itself, then $\mathfrak{F} \uplus \mathfrak{F} \models \psi \Leftrightarrow \mathfrak{F} \models \psi$.

3.3.3 Bounded morphism from $\mathfrak{F} \uplus \mathfrak{F}$ to $\mathfrak{F}'$

For the subsequent direction of the proof, we construct a bounded morphism onto $\mathfrak{F}'$, since part v of Corollary 16 in *The Handbook of Modal Logic*[4, p. 259] states that, for arbitrary modal formula ψ:

$$\text{If bounded morphism } g \text{ is onto, then } \mathfrak{F} \models \psi \Rightarrow \mathfrak{F}' \models \psi \tag{3.7}$$

Definition 3.3.3 (Bounded morphism g)**.** Define bounded morphism $g : \mathfrak{F} \uplus \mathfrak{F} \to \mathfrak{F}'$ as follows:

$$g\big((v,i)\big) = \left\{ \begin{array}{ll} \hat{x} & \text{if } (v,i) \in \big\{(x,0),(x,1)\big\} \text{ with } x \in \mathcal{W} \\ \hat{w} & \text{if } (v,i) = (w,0) \\ w' & \text{if } (v,i) = (w,1) \end{array} \right\}$$

♦

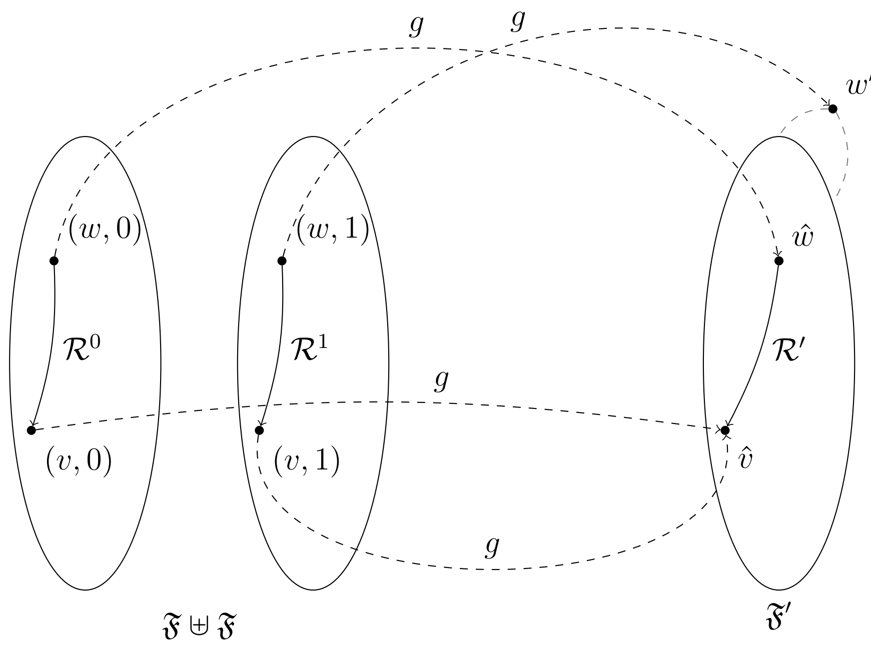


Figure 3.2: Construction of $g : \mathfrak{F} \uplus \mathfrak{F} \to \mathfrak{F}'$

We now show that g satisfies properties 1 to 3 of Definition 1.3.7.

1. We show that g is a surjective function. Suppose $b \in \mathfrak{F}'$ is arbitrary:

   (a) $b \neq \hat{w},\ b \neq w'$: then $b = \hat{x}$ for some $x \in \mathcal{W}$. Therefore, by Definition 3.3.3, $g\big((x,i)\big) = \hat{x}$ for $i \in \{0,1\}$.

   (b) $b = \hat{w}$: by Definition 3.3.3, $g\big((w,0)\big) = \hat{w}$

   (c) $b = w'$: by Definition 3.3.3, $g\big((w,1)\big) = w'$

   All elements of $\mathcal{W}'$ are therefore mapped to at least once, so g is a surjective function. ■

2. Suppose $a\mathcal{R}^{\uplus}b$ in $\mathfrak{F} \uplus \mathfrak{F}$. As in the prior proof, we must show that steps taken in the domain are mirrored by steps taken in the codomain: $a\mathcal{R}^{\uplus}b \Rightarrow g(a)\mathcal{R}'g(b)$.

   If $a\mathcal{R}^{\uplus}b = (x,i)\mathcal{R}^{i}(y,i)$ in $\mathfrak{F} \uplus \mathfrak{F}$, then by construction outlined in Definition line 3.6, it must have been the case that $x\mathcal{R}y$ in $\mathfrak{F}$. From the construction of $\mathfrak{F}'$, we derive the following cases:

   (a) If $x \neq w$ and $y \neq w$, then $\hat{x}, \hat{y}$ are their images in $\mathfrak{F}'$, neither of which is $\hat{w}$ nor w'. By Definition 3.3.3, $g\big((x,i)\big) = \hat{x}$ and $g\big((y,i)\big) = \hat{y}$ for $i \in \{0,1\}$, so since Definition 3.3.1 gives us $\hat{x}\mathcal{R}'\hat{y}$, we have $g\big((x,i)\big)\mathcal{R}'g\big((y,i)\big)$. ■

   (b) If $x = w$ and $y \neq w$, then $\hat{w}\mathcal{R}'\hat{y}$ and $w'\mathcal{R}'\hat{y}$ in $\mathfrak{F}'$ by Definition line 3.1 and 3.2.

      i. $i = 0$ Since $g\big((w,0)\big) = \hat{w}$ and $g\big((y,0)\big) = \hat{y}$ by Definition 3.3.3, $g\big((w,0)\big)\mathcal{R}'g\big((y,0)\big)$. ■

      ii. $i = 1$ Since $g\big((w,1)\big) = w'$ and $g\big((y,1)\big) = \hat{y}$ by Definition 3.3.3, $g\big((w,1)\big)\mathcal{R}'g\big((y,1)\big)$. ■

   (c) If $x \neq w$ and $y = w$,

      i. $i = 0$ Since $g\big((x,0)\big) = \hat{x}$ and $g\big((w,0)\big) = \hat{w}$ by Definition 3.3.3, $g\big((x,0)\big)\mathcal{R}'g\big((w,0)\big)$. ■

ii. $i = 1$ Since $g\big((x,1)\big) = \hat{x}$ and $g\big((w,1)\big) = w'$ by Definition 3.3.3, $g\big((x,1)\big)\mathcal{R}'g\big((w,1)\big)$. ■

(d) If $x = w$ and $y = w$, then by Definition line 3.3 and 3.4, we have $\hat{w}\mathcal{R}'\hat{w}$ and $w'\mathcal{R}'w'$. Definition 3.3.3 gives us $g\big((w,0)\big)\mathcal{R}'g\big((w,0)\big)$ and $g\big((w,1)\big)\mathcal{R}'g\big((w,1)\big)$ respectively. ■

3. Suppose $g(a)\mathcal{R}'z$ in $\mathfrak{F}'$. As above, we must show that z will be the image of some element of $\mathcal{W}^{\uplus}$ under g: $g(a)\mathcal{R}'z \Rightarrow \exists b \in \mathcal{W}^{\uplus}$ such that $a\mathcal{R}^{\uplus}b \wedge g(b) = z$.

(a) $g(a) = \hat{x} \neq \hat{w}$

i. If $z = \hat{y} \neq \hat{w}$, then $\hat{x}\mathcal{R}'\hat{y}$ in $\mathfrak{F}'$ means by Definition line 3.1 that $x\mathcal{R}y$ in $\mathfrak{F}$, so we have $(x,0)\mathcal{R}^0(y,0)$ and $(x,1)\mathcal{R}^1(y,1)$ by Definition line 3.6. Definition 3.3.3 gives us $g\big((x,i)\big) = \hat{x}$ and $g\big((y,i)\big) = \hat{y},\ i \in \{0,1\}$, which satisfies the condition. ■

ii. If $z = \hat{w}$, then $\hat{x}\mathcal{R}'\hat{w}$ in $\mathfrak{F}'$ means by Definition line 3.1 that $x\mathcal{R}w$ in $\mathfrak{F}$, so we have $(x,0)\mathcal{R}^0(w,0)$ by Definition line 3.6. Definition 3.3.3 gives us $g\big((x,0)\big) = \hat{x}$ and $g\big((w,0)\big) = \hat{w}$, which satisfies the condition. ■

iii. If $z = w'$, then $\hat{x}\mathcal{R}'w'$ in $\mathfrak{F}'$ means by Definition line 3.3 that $x\mathcal{R}w$ in $\mathfrak{F}$, so we have $(x,1)\mathcal{R}^1(w,1)$ by Definition line 3.6. Definition 3.3.3 gives us $g\big((x,1)\big) = \hat{x}$ and $g\big((w,1)\big) = w'$, which satisfies the condition. ■

(b) $g(a) = \hat{w}$

i. If $z = \hat{y} \neq \hat{w}$, then $\hat{w}\mathcal{R}'\hat{y}$ in $\mathfrak{F}'$ means by Definition line 3.1 that $w\mathcal{R}y$ in $\mathfrak{F}$, so we have $(w,0)\mathcal{R}^0(y,0)$ by Definition line 3.6. Definition 3.3.3 gives us $g\big((w,0))\big) = \hat{w}$ as well as $g\big((y,0)\big) = \hat{y}$, which satisfies the condition. ■

ii. If $z = \hat{w}$, then $\hat{w}\mathcal{R}'\hat{w}$ in $\mathfrak{F}'$ means by Definition line 3.1 that $w\mathcal{R}w$ in $\mathfrak{F}$, so we have $(w,0)\mathcal{R}^0(w,0)$ by Definition line 3.6. Definition 3.3.3

gives us $g\big((w,0))\big) = \hat{w}$, which satisfies the condition. ■

iii. If $z = w'$, then $\hat{w}\mathcal{R}'w'$ in $\mathfrak{F}'$ means by Definition line 3.3 that $w\mathcal{R}w$ in $\mathfrak{F}$, so we have $(w,0)\mathcal{R}^0(w,1)$ by Definition line 3.6. Definition 3.3.3 gives us $g\big((w,0))\big) = \hat{w}$ as well as $g\big((w,1))\big) = w'$, which satisfies the condition. ■

(c) $g(a) = w'$

i. If $z = \hat{y} \neq \hat{w}$, then $w'\mathcal{R}'\hat{y}$ in $\mathfrak{F}'$ means by Definition line 3.2 that $w\mathcal{R}y$ in $\mathfrak{F}$, so we have $(w,1)\mathcal{R}^1(y,1)$ by Definition line 3.6. Definition 3.3.3 gives us $g\big((w,1))\big) = w'$ as well as $g\big((y,0)\big) = \hat{y}$, which satisfies the condition. ■

ii. If $z = \hat{w}$, then $w'\mathcal{R}'\hat{w}$ in $\mathfrak{F}'$ means by Definition line 3.2 that $w\mathcal{R}w$ in $\mathfrak{F}$, so we have $(w,1)\mathcal{R}^1(w,0)$ by Definition line 3.6. Definition 3.3.3 gives us $g\big((w,1))\big) = w'$ as well as $g\big((w,0))\big) = \hat{w}$, which satisfies the condition. ■

iii. If $z = w'$, then $w'\mathcal{R}'w'$ in $\mathfrak{F}'$ means by Definition line 3.4 that $w\mathcal{R}w$ in $\mathfrak{F}$, so we have $(w,1)\mathcal{R}^1(w,1)$ by Definition line 3.6. Definition 3.3.3 gives us $g\big((w,1))\big) = w'$, which satisfies the condition. ■

Therefore, $g : \mathfrak{F} \uplus \mathfrak{F} \to \mathfrak{F}'$ is a surjective bounded morphism, and we can again conclude by condition iv of Corollary 16 in *The Handbook of Modal Logic*[4, p. 259] that:

$$\text{If } g \text{ is an onto bounded morphism, } \mathfrak{F} \uplus \mathfrak{F} \models \psi \Rightarrow \mathfrak{F}' \models \psi$$

And since part iv of Proposition 26 in *The Handbook of Modal Logic*[4, p. 262] gives us:

$$\mathfrak{F} \uplus \mathfrak{F} \models \psi \Leftrightarrow \mathfrak{F} \models \psi$$

The other direction is complete [4, p. 259]. □

3.4 Conclusion

The construction of surjective bounded morphism $f : \mathfrak{F}' \to \mathfrak{F}$ and surjective bounded morphism $g : \mathfrak{F} \uplus \mathfrak{F} \to \mathfrak{F}'$ from intermediate structure $\mathfrak{F} \uplus \mathfrak{F}$ onto $\mathfrak{F}$ gives us:

$$\mathfrak{F}' \models \psi \Rightarrow \mathfrak{F} \models \psi$$
$$\mathfrak{F} \uplus \mathfrak{F} \models \psi \Rightarrow \mathfrak{F}' \models \psi$$

From Proposition 26 part iv in *The Handbook of Modal Logic*[4, p. 262]:

$$\mathfrak{F} \uplus \mathfrak{F} \models \psi \Leftrightarrow \mathfrak{F} \models \psi$$

And so:

$$\mathfrak{F} \models \psi \Rightarrow \mathfrak{F}' \models \psi$$

Hence:

$$\mathfrak{F} \models \psi \Leftrightarrow \mathfrak{F}' \models \psi \tag{3.8}$$

Therefore, axiom characterizations are valid in $\mathfrak{F}$ if and only if they are valid in $\mathfrak{F}'$, as desired.

Furthermore, since we have constructed surjective bounded morphisms, part i of Corollary 16 gives us:

$$\forall u \in \mathcal{W}' \quad \mathfrak{M}, u \models \varphi \Leftrightarrow \mathfrak{M}', \hat{u} \models \varphi$$

And so for the pointed model $\mathfrak{M}, w$, using instantiation $u = w$ gives us:

$$\mathfrak{M}, w \models \varphi \Leftrightarrow \mathfrak{M}', \hat{w} \models \varphi$$

So the truth of formula φ is preserved, as desired[4, pp. 259, 262].

This completes the proof of the theorem $\square$

Theorem 3.1.1 guarantees that there will be no "gaps" in the existence of satisfying models for φ, that if a model with $|\mathcal{W}| = \ell$ exists, then for every $k > 0$, there will be a satisfying model with $|\mathcal{W}| = \ell + k$. Representing this graphically, if 0 represents the absence of a model of a certain size, and 1 represents the existence of a satisfying model for φ which validates the axiom schema ψ, we are guaranteed that

...	$\ell - 2$	$\ell - 1$	ℓ	$\ell + 1$	$\ell + 2$	...
...	0	0	1	1	1	...

And not

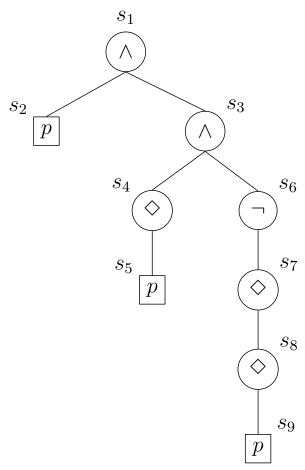
...	$\ell - 1$	ℓ	$\ell + 1$	$\ell + 2$	$\ell + 3$	...
...	0	1	0	1	1	...

Chapter 4

Results and Examples

In this chapter, we will go through the process of using the modal solver suite, starting with a specification of the formula to be satisfied and an example of a simple pointed model in the modal logic $\mathbb{K}$ that satisfies that formula, then producing an Enfragmo problem instance, and the steps necessary to obtain the raw XML, .dot format representation, and final .svg image. Then we will discuss the effects of different axiom characterizations on the model produced by Enfragmo.

Example 4.0.1. For the formula $\varphi = p \wedge \Diamond p \wedge \neg\Diamond\Diamond p$ - designed for educational purposes to be unsatisfiable on a symmetric frame - we have the following syntax tree:



This translates to the Enfragmo problem instance file:

```
TYPE Subformula [1.. 9]
TYPE World [1.. 1]
PREDICATE Atom
(2)
(5)
(9)
PREDICATE And
(1,2,3)
(3,4,6)
PREDICATE Diamond
(4,5)
(7,8)
(8,9)
PREDICATE Not
(6,7)
PREDICATE SameAtom
(2,5)
(2,9)
```

Running the decision procedure in the base modal logic $\mathbb{K}$ yields raw XML output:

```
<Main>
Parsing is finished!
c running SatElite... done.
c solved during preprocessing!
<Satisfiable/>
<TimingInfo>
<Grounding Time= "0.000s"/>
<MakeCNF Time= "0.000s"/>
<Solving Time= "0.000s"/>
<SolutionProcessing Time= "0.000s"/>
</TimingInfo>
<ToBePrintedPredicates>
<PredicateInfo>
<PredicateSymbol>
<BasicInfo Name='TrueAt' IsGiven= '1' ToBePrinted= '0'/>
<TypeInfoCollection><IntTypeInfo Name='Subformula'/>
<IntTypeInfo Name='World'/>
</TypeInfoCollection>
</PredicateSymbol>
<DataSet Name= 'TrueAt' TypeSize= '2' >
<ARow><IntValue Name= '1'/><IntValue Name= '1'/><True/></ARow>
<ARow><IntValue Name= '2'/><IntValue Name= '1'/><True/></ARow>
<ARow><IntValue Name= '2'/><IntValue Name= '2'/><True/></ARow>
<ARow><IntValue Name= '3'/><IntValue Name= '1'/><True/></ARow>
<ARow><IntValue Name= '4'/><IntValue Name= '1'/><True/></ARow>
<ARow><IntValue Name= '5'/><IntValue Name= '1'/><True/></ARow>
<ARow><IntValue Name= '5'/><IntValue Name= '2'/><True/></ARow>
<ARow><IntValue Name= '6'/><IntValue Name= '1'/><True/></ARow>
<ARow><IntValue Name= '6'/><IntValue Name= '2'/><True/></ARow>
<ARow><IntValue Name= '8'/><IntValue Name= '1'/><True/></ARow>
<ARow><IntValue Name= '9'/><IntValue Name= '1'/><True/></ARow>
<ARow><IntValue Name= '9'/><IntValue Name= '2'/><True/></ARow>
</DataSet>
</PredicateInfo>
<PredicateInfo>
<PredicateSymbol>
<BasicInfo Name='Accessible' IsGiven= '1' ToBePrinted= '0'/>
<TypeInfoCollection>
<IntTypeInfo Name='World'/>
<IntTypeInfo Name='World'/>
</TypeInfoCollection>
</PredicateSymbol>
<DataSet Name= 'Accessible' TypeSize= '2' >
<ARow><IntValue Name= '1'/><IntValue Name= '2'/><True/></ARow>
</DataSet>
</PredicateInfo>
</ToBePrintedPredicates>
</Main>
```

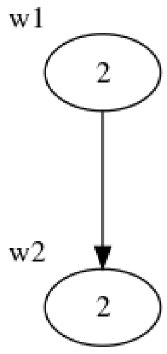
◀

Even with a small model, this raw XML output is hard for humans to parse and to gain useful information from. Therefore, the additional machinery of the modal solver suite was developed to parse this output and to create a Digraph object in the .dot graph representation language as an intermediate step to producing an .svg image.

Example 4.0.2. The output from Example 4.0.1 is used to produce the following .dot graph representation format:

```
digraph {
graph [nodesep=1.0]
edge [minlen=2.0]
1 -> 2
1 [label=2 xlabel=w1]
2 [label=2 xlabel=w2]
}
```

The Kripke model constructor is then invoked to produce a visual representation of the model produced by Enfragmo:



◀

With the added machinery for restricting the accessibility relation of satisfying models, users can dictate what features a model should have in the Enfragmo specification language. It is important to note that we do not check the properties of the user-specified characterizations, in particular for the finite model property. This means that if a user specifies conditions on the frame which do not allow for finite models, then the modal solver suite will erroneously report that no model exists, where an infinite satisfying model might exist.

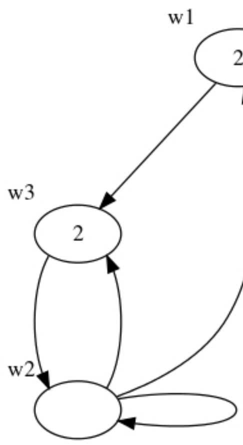
Additionally, as mentioned in § 1.3.4, the truth of first-order formulas is not guaranteed to be invariant under bounded morphisms; therefore, the proof of Chapter 3 does not necessarily guarantee that the doubling and halving approach will work for all first-order constraints, but rather only those with modal correspondents. As long as the user is aware of these limitations, then given appropriate conditions on the frame, we can obtain interesting new information in the form of the models produced, as well as insight from their absence.

Example 4.0.3. Suppose we require that the frame of a satisfying model be serial (i.e. no terminal worlds). Then we would tell the modal solver suite to add the following condition contained in an external file:

```
// Serial
! w1: World : ? w2: World : ( Accessible (w1,w2) );
```

This ultimately produces the following .dot format file and .svg image:

```
digraph {
graph [nodesep=1.0]
edge [minlen=2.0]
1 -> 3
2 -> 1
2 -> 2
2 -> 3
3 -> 2
1 [label=2 xlabel=w1]
2 [label="" xlabel=w2]
3 [label=2 xlabel=w3]
}
```



Note that although we do have some symmetric links in the model produced, not all possible symmetric relational links are present (e.g. since $w_2\mathcal{R}w_1$, we would have $w_1\mathcal{R}w_2$ if symmetry were imposed.)

◀

Example 4.0.4. Now suppose that we want to see if a satisfying model can be symmetric. We add the following condition:

```
// Symmetric
! w1:World w2:World : ( Accessible (w1,w2) => Accessible (w2,w1) );
```

No model can be produced for the given formula, as demonstrated by the modal solver suite:

```
/usr/bin/python3.4/home/wbkboyer/PycharmWorkspace/modalSolverSuite/driverObj.py


The formula failed to have a satisfying model with at most 512 worlds.
```

The underlying Enfragmo output gives a more comprehensive analysis of the final run for $|\mathcal{W}| = 2^9 = 512$:

```
<Main>
Parsing is finished!
c running SatElite... done.
c 818691 assignments made, starting to guess now.
c ===================================[MXC]=====================================
c | Conflicts |                Clause Statistics                   |   Vars   |
c |           |    CARD     BIN      O3+      L3+   L3+Lim  Lit/CL |          |
c ============================================================================
c |         0 |       0 4699148  1572351        0        0    -nan |  1865729 |
c |       100 |       0 4699125  1572346       97     1050   512.8 |  1047033 |
c |       210 |       0 4699231  1572346      101     1105   507.0 |  1047033 |
c |       441 |       0 4699459  1572346      104     1220   497.7 |  1047033 |
c |       805 |       0 4699821  1572346      106     1402   497.4 |  1047033 |
c |      1315 |       0 4695743  1571462      106     1657   497.4 |  1046149 |
c |      1975 |       0 4696069  1571462      108     1987   491.9 |  1044513 |
c restarts: 15
c decisions: 5167075
c propagations: 288744781
c conflicts: 1975
c 156.751s elapsed
<Unsatisfiable/>
<TimingInfo>
<Grounding Time= "0.010s"/>
<MakeCNF Time= "1.120s"/>
<Solving Time= "156.440s"/>
<SolutionProcessing Time= "0.000s"/>
</TimingInfo>
Unsatisfiable
</Main>
```

It is the `<Unsatisfible/>` tag that the Modal Solver Suite looks for when it determines that a model of a certain size does not exist, and when it finally determines that no finite model exists with the given axiom characterization.

◀

Chapter 5

Conclusions

5.1 Summary of Contributions

Presented in this thesis are the specifications for Enfragmo theory files capturing the semantics of modal logic, the means for expressing modal formulas as Enfragmo problem instances, and the machinery of the modal solver suite, which allows us to run the procedure on large numbers of modal formulas with flexible frame conditions and to produce meaningful output.

The proof of correctness presented demonstrates that the doubling and halving/binary search procedure is an appropriate technique for finding the minimum number of worlds required for a satisfying pointed model of a formula, regardless of the frame conditions enforced.

This procedure for extracting the relational information from Enfragmo's raw XML output is easily generalizable for application to other model expansion problems.

5.2 Future Work

The procedure outlined in the preceding chapters can be further developed in numerous important ways, which are not yet well-explored in the literature, so as to be fruitful in multiple domains.

5.2.1 Dealing with multiple modalities

The current implementation handles a single accessibility relation through the use of the predicate `Accessible(s, t)` where `s, t` are worlds. This can be extended to the case where we represent multiple accessibility relations using a three place accessibility relation: `Accessible(`i`, s, t)` where $i \in I$ is an index set of the desired relations.

Each of these accessibility relations would be able to obey their own axiom characterization, as the procedure to insert frame conditions can be easily modified to accommodate the new first argument. This can even be further automated, for example indicating that $\mathcal{R}_1, \mathcal{R}_4, \mathcal{R}_5$ should obey the conditions outlined in `FrameConditions1.txt`, and that $\mathcal{R}_2, \mathcal{R}_3$ must obey those in `FrameConditions2.txt`; all indices would be substituted with an appropriate script and inserted sequentially into the final theory file.

5.2.2 Dynamic model updates using Enfragmo

Modal logics with update operators, such as public announcement logic, propositional dynamic logic, and dynamic epistemic logic, all model different phenomena but require similar mechanisms vis-á-vis dynamic updates. By creating a new theory file to represent the semantics of a model update, and parsing the initial model generated by Enfragmo to create a new problem instance file, one could feed the result back into Enfragmo to obtain an updated model.

5.2.3 Different phrasings of frame correspondents

In using Enfragmo, one acknowledges that there might be some computational overhead from the use of a general purpose model expansion solver for this application, although Enfragmo performs extremely competitively with respect to other systems [2, p. 20]. An interesting avenue of exploration would be to see the effect of using logically equivalent formulations of the same first-order expressible conditions on the accessibility relation. After observing the change in how Enfragmo grounds the formulas, obtaining concrete timing results to demonstrate if there is an appreciable effect would be another interesting result.

Additionally, developing a characterization of which phrasings of relation conditions behaved poorly would be a useful guideline for future application of the system.

5.2.4 Improved Visualizations and User Interface

The use of this modal solver suite as a learning tool is an exciting area for expansion. Although the .svg files produced by the Kripke model constructor are informative, they are not always rendered nicely by the Python Graphviz library, in particular for larger models. However, taking the intermediate .dot files, which capture the structure of the models in this graph representation language, we can use any one of a number of rendering programs to create different styles of visualizations.

These visualizations would be easy to incorporate into a self-contained suite with a GUI, which would even further improve ease of use by students and professionals alike. A program such as PyInstaller could be used to create a standalone executable file for distribution to multiple platforms [19].

Bibliography

[1] Python 3 Documentation - 16.1. os — Miscellaneous operating system interfaces, 2016.

[2] Aavani, A., Wu, X., Tasharrofi, S., Ternovska, E., and Mitchell, D. Enfragmo: A system for modelling and solving search problems with logic. In *Logic for Programming, Artificial Intelligence, and Reasoning*, N. Bjørner and A. Voronkov, Eds., vol. 7180 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 15–22.

[3] Blackburn, P., de Rijke, M., and Venema, Y. *Modal logic*, vol. 53. Cambridge University Press, Cambridge, UK;New York;, 2001.

[4] Blackburn, P., van Benthem, J., and Wolter, F. *Handbook of Modal Logic*, 1st ed., vol. 3. Elsevier, Amsterdam;Boston;, 2007.

[5] Déharbe, D., and Ranise, S. Satisfiability solving for software verification. *International journal on software tools for technology transfer 11*, 3 (2009), 255–260.

[6] Dovier, A., Piazza, C., and Policriti, A. A fast bisimulation algorithm. In *Computer Aided Verification* (2001), Springer, pp. 79–90.

[7] Enjalbert, P., and del Cerro, L. F. Modal resolution in clausal form. *Theoretical Computer Science 65*, 1 (1989), 1 – 33.

[8] Fischer, M. J., and Ladner, R. E. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences 18*, 2 (1979), 194 – 211.

[9] Gasquet, O., Herzig, A., Said, B., and Schwarzentruber, F. *Kripke's Worlds: An Introduction to Modal Logics via Tableaux*. Springer, 2014.

[10] Giunchiglia, F., and Sebastiani, R. Building decision procedures for modal logics from propositional decision procedures: The case study of modal k(m). *Information and Computation 162*, 1–2 (2000), 158 – 178.

[11] Hustadt, U., and Schmidt, R. A. Using resolution for testing modal satisfiability and building models. *Journal of Automated Reasoning 28*, 2 (2002), 205–232.

[12] Olivetti, N. *Automated Reasoning with Analytic Tableaux and Related Methods: 16th International Conference, TABLEAUX 2007, Aix en Provence, France, July 3-6, 2007, Proceedings*, vol. 4548. Springer Science & Business Media, 2007.

[13] Pan, G., Sattler, U., and Vardi, M. Y. BDD-based decision procedures for the modal logic K. *Journal of Applied Non-Classical Logics 16*, 1-2 (2006), 169–207.

[14] Schmidt, R. A., and Hustadt, U. The axiomatic translation principle for modal logic. *ACM Transactions on Computational Logic (TOCL) 8*, 4 (2007), 19.

[15] Sebastiani, R., and Tacchella, A. SAT Techniques for Modal and Description Logics. In *Handbook of Satisfiability*, A. Biere, M. Heule, H. van Maaren,

and T. Walsh, Eds., Frontiers in Artificial Intelligence and Applications. IOS Press.

[16] van Benthem, J. *Modal Logic and Classical Logic.* Distributed in the U.S.A. By Humanities Press, 1983.

[17] van Benthem, J. *Modal Logic for Open Minds.* Center for the study of Language and Information, 2010.

[18] Vardi, M. Y. Why is modal logic so robustly decidable? In *Descriptive Complexity and Finite Models*, N. Immerman and P. G. Kolaitis, Eds., vol. 31 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science.* AMS and DIMACS, 1997, pp. 149–184.

[19] Zibricky, M. PyInstaller official website, 2015.

Appendix A

The Modal Solver Suite

Please view current code repositories hosted on GitHub:

- Python implementation of the Modal Solver Suite:

  `https://github.com/wandaboyer/modalSolverSuite.git`

- Problem instance files for simple modal formulas, as well as theory file for modal logic $\mathbb{K}$ with a sample file of optional conditions:

  `https://github.com/wandaboyer/MSS-SupplementaryFiles.git`

Index