Modal Logic in Computer Science

Leigh Lambert
Edith Hemaspaandra, chairperson

September 11, 2003

1 Introduction

Modal logic is similar to traditional propositional logic with the additions of extra operators, such as "it is possible that...," "it is necessary that...," "it will always be the case that...," "x believes that...," etc. For example, the phrase "always not deadlock" can be translated into a modal formula where "always" is the modal operator, "not" is the negation operator, and "deadlock" is a fact about a state of a program. Modal logic is useful in verifying the correctness of programs. The $\Box$ is introduced as a basic modal connective. $\Box\varphi$ intuitively asserts that φ is necessarily true. Also, $\neg\Box\neg\varphi$, abbreviated as $\Diamond\varphi$, asserts that φ is possibly true. Again, the semantic definitions of modal connectives can change depending on the situation they are used in. These modal operators need to follow certain rules. For example, if "always p," then, also, "always always p." Also, when we table processors, we want to make sure that if a processor "knows that p," then p is actually true. These rules will lead to different modal logics.

Modal logic is important to many aspects of Computer Science, such as Artificial Intelligence, Distributed Systems, linguistics, etc., because terms such as necessary and possibly can be given concrete applications such as "according to an agent's knowledge" or "after the program terminates" [3]. simulate applications using logic in order to verify the correctness of these applications. Modal logic is associated with other logics such as deontic logic and temporal logic. Deontic logics introduce operators for "it is obligatory that ..." and "it is permitted that..." Temporal logic is used to introduce logic about future and past facts. Temporal logic is extremely useful in program verification.

This proposal introduces more specifics about the following: (1) modal logic, (2) its uses in Computer Science, and (3) the computational complexity

of satisfiability and other problems in modal logics. These topics will be the basis for an in-depth survey of previous research in the area of modal logic. I will write an overview paper to discuss different results and methods that I will obtain. This paper will also give the necessary background in modal logic, its applications within Computer Science, and complexity theory. It will put the complexity of different modal logics into a general framework.

The rest of the proposal is organized as follows. Section 2 discusses the basics of uni-modal logic, the simplest form of modal logic. Section 3 introduces the uses of modal logic in areas within Computer Science. In this section, multi-modal logic, which is useful in Artificial Intelligence and Distributed Systems theory, will be examined. Common and distributed knowledge are introduced as well. Section 4 will then discuss the problem of satisfiability of formulas in propositional logic. We will also examine some of the basics of computational complexity. Our last section states some complexity results for modal logics.

2 Modal Logic

Let Φ be a nonempty set of primitive propositions, labelled $p, q, p', q', \ldots$. These propositions describe facts about a particular world or state, such as "The sky is cloudy today" or "Max's chair is broken." Let $L(\Phi)$ be the least set of formulas containing Φ that are closed under conjunction, negation, and the modal operator $\Box$. Thus, if φ and ψ are in $L(\Phi)$, then $\varphi \wedge \psi$, $\neg\varphi$, and $\Box\phi$ are also formulas in $L(\Phi)$. Some operators can be defined in terms of other operators. $\varphi \vee \psi$ can be written as $\neg(\neg\varphi \wedge \neg\psi)$ due to de Morgan's Law and $\varphi \rightarrow \psi$ as $\neg(\varphi \wedge \neg\psi)$. Also true is used as an abbreviation for some propositional tautology, like $p \vee \neg p$, and false is defined as $\neg$ true. The $\Diamond$ operator is defined as $\Diamond\psi \leftrightarrow \neg\Box\neg\psi$. The operators $\Diamond$, $\vee$, $\rightarrow$, and $\leftrightarrow$ do not appear in modal formulas for simplification, but for convenience they do appear in modal text.

Now that we have described the syntax of modal languages, we can introduce the semantics to help determine whether a given formula is true or false. Modal semantics is formally defined using Kripke structures. [4] defines a Kripke structure to be a tuple (W, R, V) where W is a set, R is a binary relation on W, and V maps $\Phi \times W$ to {true, false}. The set W is a set of 'possible worlds,' or states, and R determines which states are accessible[1] from any given world in W. V determines which facts, or propositions, are true in each of the worlds.

[1] We will say that state $b \in W$ is accessible from state $a \in W$ if and only if $(a, b) \in R$.

We can now describe the process of how to determine whether a formula is true at a given state in a structure. Truth depends on both the state and structure. Formulas are similar to propositions in that a formula can be true in one state and false in another. We will define the notation $(M, w) \models \varphi$ as meaning "φ is true at state w in structure M," or "(M, w) satisfies φ." Induction is used on φ to determine the $\models$ relation. V gives the information needed for the base case, in which φ is a proposition:

$$(M, w) \models p \text{ (for a proposition } p \in \Phi) \text{ if } V(p)(w) = \text{true}.$$

Induction on conjunction and negation are similar to the rules used in propositional logic. That is $\varphi \wedge \psi$ is true exactly if φ is true and ψ is true, and $\neg\varphi$ is true if and only if φ is false:

$$(M, w) \models \varphi \wedge \psi \text{ if } (M, w) \models \varphi \text{ and } (M, w) \models \psi.$$

$$(M, w) \models \neg\varphi \text{ if } (M, w) \not\models \varphi.$$

ψ is necessarily true in state w of structure M if and only if ψ is true at all states accessible from w. Written formally, we have

$$(M, w) \models \Box\psi \text{ if } (M, w') \models \psi \text{ for all } w' \text{ such that } (w, w') \in R.$$

Modal structures can be depicted using diagrams similar to directed graphs. Each node is a state in W. At each node, there are labels of which propositions are true or false at the particular state. The edge represents a possibility relation within R. If $(w, w') \in R$, then there exists an edge from w to w'.

For example, suppose $\Phi = \{p, q\}$ and $M = (W, R, V)$ where $W = \{u, v, w\}$. Let $V(u)(q) = V(v)(p) = V(w)(p) = \text{true}$. Also, v and w are accessible exactly at u and v. The following graph captures this situation [2].

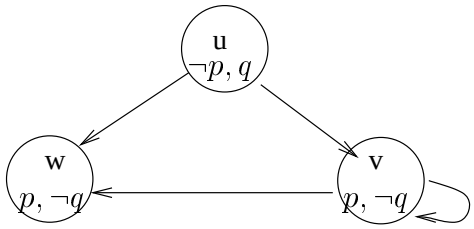

Figure 1

Note that we have $(M, w) \models p$ and $(M, v) \models p$. Thus, $(M, u) \models \Box p$, since p holds in all states accessible from u. Also, $(M, w) \models \neg q$ and $(M, v) \models \neg q$. It follows that $(M, v) \models \Box \neg q$. $(M, w) \models \Box \neg q$ because there is no state accessible from w. Therefore, $(M, u) \models \Box\Box \neg q$ since $\Box \neg q$ holds in all states accessible from u.

Now that we have introduced some basic modal logic, we will observe properties of different modal structures based on the modal operators. A modal system is a set of modal formulas. Modal systems are based on combinations of axioms that are developed from conditions placed on the possibility, or accessibility, relation R. The following axioms and rules of inference hold true for the basic modal system K. All other modal systems we will look at are extensions of K.

A1 All instances of tautologies of the propositional calculus are allowed.

A2 $(\Box\varphi \wedge \Box(\varphi \rightarrow \psi)) \rightarrow \Box\psi$.

R1 If $\vdash \varphi$ and $\vdash \varphi \rightarrow \psi$, then $\vdash \psi$ (Modus ponens).

R2 If $\vdash \varphi$, then $\vdash \Box\varphi$ (Generalization).

φ is said to be K *provable*, denoted $K \vdash \varphi$, if φ can be inferred from instances of A1 and A2, using R1 and R2 [2]. There are many different axioms. The axioms used within a modal logic depends on the modality used within an application. Some well-known axioms are the following:

A3 $\Box\varphi \rightarrow \varphi$;

A4 $\Box\varphi \rightarrow \Box\Box\varphi$;

A5 $\neg\Box\varphi \rightarrow \Box\neg\Box\varphi$;

A6 $\neg\Box$(false).

Listed below are some common modal systems along with axioms that are always true within each:

- $T \equiv K+$ A3,
- $S4 \equiv T+$ A4,
- $S5 \equiv S4+$ A5,
- $KD45 \equiv S5+$ A6.

If M is a modal structure, then it is a K-model. If M is a modal structure and R is (a)reflexive, (b)reflexive and transitive, (c) reflexive, transitive, and symmetric, (d)euclidean[2] and transitive, then M is respectively a (a)T-model, (b)$S4$-model, (c)$S5$-model, (d)$KD45$-model. For $X \in \{K, T, S4, S5, KD45\}$, a modal formula φ is satisfiable in an X-model if and only if $\neg\varphi$ is not X-provable[3]. Furthermore, φ is valid in an X-model exactly if φ is X-provable. Similar equivalences hold for most modal logics. This will be important in determining the complexity of satisfiability in some modal logics.

3 Applications of Modal Logic

In the section above, we introduced uni-modal logic. This means that there is only one modal operator allowed in our modal language. However, sometimes we may want more than one modality. For example, in Artificial Intelligence, a modal operator is used to describe what each agent within a group knows. Let the agents of a group be named $1, \ldots, n$. Then, $K_i\varphi$, where $1 \leq i \leq n$ would translate to "agent i knows φ." This information would change some of the details introduced in the previous section. For instance, a modal structure for multi-modal logic has $(W, R_1, R_2, \ldots, R_n, V)$, where each of $R_i, 1 \leq i \leq n$, is a binary relation that describes the possible worlds in W agent i can reach. Also, modal systems $K_n, T_n, S4_n, S5_n,$ and$KD45_n$ describe modal systems with n modal operators. Among the axioms and rules of inference, $\Box$ can be replaced with $K_i, 1 \leq i \leq n$, to allow them to apply for multi-modal logics.

Another application of modal logics is program verification. For example, let $[p]$ be a modal operator where p is a program. Then, let $[p]A$ be an assertion, where A is any arbitrary assertion, that means "if p terminates, then A holds." [4]. In this case, replacing the word 'assertion' with 'formula' will give the reader a similar interpretation of the definition of $L(\Phi)$ introduced in the last section. A temporal logic, where states within a structure are instances of time, is a modal style used typically for program verification since programs cannot return to instances of time that have already passed. This logic is especially popular among parallel programming. The relative timing of concurrent processes must be carefully coordinated to ensure that

[2]Let A be a binary relation on a set S. We say that A is euclidean if, for all $s, t, u \in S$, if $(s, t) \in A$ and $(s, u) \in A$, then $(t, u) \in A$. [2]

[3]To determine if φ is X-provable, φ can be inferred from instances of the axioms contained in X, using R1 and R2 [2].

integrity of shared information is contained.

Modal logic can also be used to describe common and distributed knowledge, which are useful in both Artificial Intelligence and Distributed Systems theory. Common knowledge within a group is described as facts that everyone knows that everyone knows that everyone knows that ... to be true. Distributed knowledge within a group are facts that are inferred from the combined knowledge of the individuals [2]. It is easy to see that the results of a program that is run in a distributed system can be classified as distributed knowledge. None of the individual systems could solve the program on its own. One example of common knowledge is the language used within a group to communicate.

4 Review of Satisfiability and Complexity Classes

The goal of satisfiability is to find if there exists some assignment (true or false) for each of the literals within some formula φ such that φ is true. Satisfiability of propositional logic is, in some formulas, fairly simple to solve. However, satisfiability among formulas that are written in conjunctive normal form is hard to solve, in that there is no algorithm today that runs in polynomial time in the length of the input and determines whether all formulas written in conjunctive normal form are satisfiable. A formula φ is in conjunctive normal form if it is written in the following way

$$(l_{11} \vee l_{12} \vee \ldots \vee l_{1k_1}) \wedge (l_{21} \vee l_{22} \vee \ldots \vee l_{2k_2}) \wedge \ldots \wedge (l_{m1} \vee l_{m2} \vee \ldots \vee l_{mk_m})$$

If the number of literals within each clause φ is less than three, then satisfiability of φ can be solved in polynomial time.

Languages that are decidable can be solved by algorithms. Algorithms can be defined in terms of how fast they can be run or how much space is needed to run them as a function of the input size. Algorithms that run in polynomial time can be run in $O(n^k)$, where n is the size of the input and $k \in N$. The algorithm for satisfiability runs in nondeterministic polynomial time, meaning that it can be run in polynomial time if there is guessing allowed in choosing assignments for the literals. Satisfiability is, in fact, NP-complete, the hardest of the NP problems.

In addition to P and NP, PSPACE is the class of languages that have algorithms that can be run such that the amount of memory used is no more than a polynomial of the input size. Like NP-complete, there is a subclass of PSPACE that contains the hardest languages of PSPACE, called PSPACE-complete. Note that EXPTIME contains algorithms that can be run in at

most an exponential of the input size. We know that P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXPTIME and that P $\neq$ EXPTIME. We do not know if P = NP.

5 Satisfiability for Modal Logics

The following is a list of some modal logics and their complexity classes, as seen in [2]. All modal logics are NP-hard, meaning that each modal logic will not be less hard than NP-complete. This is by no means an exhaustive list of modal logics.

$K_n, n \geq 1$ PSPACE-complete

$K_n, n \geq 1$, **with common knowledge operator** EXPTIME-complete

$T_n, n \geq 1$ PSPACE-complete

$T_n, n \geq 1$, **with common knowledge operator** EXPTIME-complete

$S5$, **with or without common knowledge operator** NP-complete

$S5_n, n \geq 2$ PSPACE-complete

$S5_n, n \geq 2$, **with common knowledge operator** EXPTIME-complete

$KD45$, **with or without common knowledge operator** NP-complete

$KD45_n, n \geq 2$ PSPACE-complete

$KD45_n, n \geq 2$, **with common knowledge operator** EXPTIME-complete

6 Deliverables

The end products of this project will contain the following deliverables:

- An overview paper on modal logic and its applications in Computer Science. A tentative outline of the paper is as follows.
  - Introduction
  - Basic Uni-modal Logic and Multi-modal Logic
  - Use of Modal Logics in Computer Science
  - Introduction to Complexity Theory
  - Classification of Complexity Results for Modal Logics

The overview paper is targeted towards any person that has a general background in Computer Science.

- A web page containing links to online sources that are pertinent to this project along with annotations for each source

7 Proposed Schedule

The following is a proposed schedule of the dates when parts of the overview paper will be completed, along with a possible date for the project defense.

Dates	Goals Accomplished
9/8 - 9/22	Gather sources on basic Modal logic systems and their complexities. Begin web page of online sources.
9/22 - 10/6	Write first version of introduction to complexity theory and basic modal logic.
10/6 - 10/20	Research the uses of modal logic in Computer Science and the complexity of more modal logics. Update web page of online sources.
10/20 - 11/3	Write first version of use of modal logics in Computer Science and classification of complexity results for modal logics
11/3 - 11/17	Write introduction and conclusion
11/17 - 12/1	Edit and review paper. Review and update web page.
12/8	Tentative date for project defense

References

[1] Erich Grädel, Why are modal logics so robustly decidable?, *Current Trends in Theoretical Computer Science. Entering the 21st Century,*: 393-408, World Scientific,(2001).

[2] Joseph Y. Halpern, Yoram Moses, A guide to completeness and complexity for modal logics of knowledge and belief, *Artificial Intelligence* **54**: 311-379(1992).

[3] J. Hintikka. *Knowledge and Belief,* Cornell University Press: Ithaca, NY. 1962.

[4] Richard E. Ladner, The Computational Complexity of Provability in Systems of Modal Propositional Logic, *SIAM Journal on Computing* **14(1)**: 113-118,(1981).

[5] Moshe Y. Vardi, Why is Modal Logic So Robustly Decidable?, Rice University.

[6] Edith Spaan, *Complexity of Modal Logics,* Haveka, B.V., Alblasserdam, 1993.