UNIVERSITÉ DE GENÈVE

OUTILS FORMELS AVANCÉS
14X014

Exploring other Modal logics

Authors: Dany A. DARGHOUTH
Ethan ARM

E-mails: dany.al-moghrabi@etu.unige.ch
ethan.arm@etu.unige.ch

June 2024

UNIVERSITÉ
DE GENÈVE
FACULTÉ DES SCIENCES
Département d'informatique

Summary

1 Introduction **2**

2 Basic notions **3**
2.1 Modal logic . . . 3
2.2 Possible words . . . 3
2.3 Kripke semantics . . . 3
2.4 Metalogic . . . 3
2.5 Orders of logic . . . 4

3 Exploration and deduction **5**
3.1 Modal logic overview . . . 5
3.1.1 Deontic logic . . . 5
3.1.2 Computational Tree logic . . . 5
3.1.3 Past + CTL . . . 5
3.1.4 PCTL . . . 6
3.2 Metalogic influences on Modal logic . . . 6

4 Developed Methodology **7**
4.1 Intuition . . . 7
4.2 Formalization . . . 7
4.2.1 From natural language to formal logic . . . 7
4.2.2 From a Logic to another . . . 7
4.2.3 Expressivity limitations . . . 7
4.3 Example . . . 8

5 Project Evolution **9**
5.1 Initial Expectations . . . 9
5.2 Expectations Evolution . . . 9
5.3 Retrospective . . . 9

6 Conclusion **10**

The grammar, spelling and formatting of this report have been reviewed using AI tools such as Quillbot and Grammarly, explaining a possible high detection score when using AI detection tools.

1. Introduction

Modal logic is a branch of logic that expands the traditional framework by introducing modalities, which qualify the truth of propositions in terms of necessity and possibility. Unlike classical logic, which confines itself to definitive truth values—true or false—modal logic enables the examination of statements within the context of different scenarios or "possible worlds." This approach provides a more sophisticated means of evaluating propositions that consider various conditions and perspectives.

The origins of modal logic can be traced back to ancient philosophical inquiries by figures such as Aristotle, who delved into the notions of necessity and possibility. However, it was not until the 20th century that modal logic was rigorously formalized by pioneers like C.I. Lewis, who introduced systems of strict implication, and Saul Kripke, whose development of "possible worlds" semantics revolutionized the field. Kripke's framework allowed for a systematic interpretation of modal propositions, laying the groundwork for the modern understanding of modal logic.

Today, modal logic plays a pivotal role in a multitude of disciplines. In philosophy, it aids in addressing metaphysical and epistemological questions, such as the nature of necessity, possibility, and counterfactuals. In computer science, modal logic is instrumental in the specification and verification of software and hardware systems, particularly through temporal logic, which handles the dynamic behavior of systems over time. Linguistics benefits from modal logic in modeling the semantics of natural language, especially in understanding modal verbs and conditional statements. Moreover, in artificial intelligence, modal logic is crucial for knowledge representation and reasoning, enabling more advanced AI systems that can handle complex scenarios involving beliefs, intentions, and obligations.

This report explores the foundational concepts, various types, and significant applications of modal logic. By examining its theoretical underpinnings and practical uses, we aim to highlight the importance of modal logic in both theoretical research and practical applications. Despite facing challenges such as computational complexity and expressiveness limitations, modal logic continues to be a dynamic and evolving field, offering valuable insights and solutions across different domains. As research progresses, modal logic is expected to contribute to new areas and technologies, further expanding its impact and utility.

2. Basic notions

In this section, we will be presenting a few basic notions and notation necessary to understand our work, such as *Modal logic*, *Meta-logic*, *Kripke structures*, etc. The initiate reader could skip this part and go to section 3.

2.1 Modal logic

Modal logic is a type of logic characterized by a pair of dual operators expressing both the possibility and necessity, respectively. This type of logic can be defined over any preexisting logic in which case we refer to both operators as *modal operators*. The first operator, representing possibility, is generally represented using the $\Diamond$ symbol. The other, representing necessity, is represented using the $\Box$ symbol. Both of these operators are unary, meaning that they are applied over a single modal proposition. For example on a proposition P, giving us $\Diamond P$ "possibly P", and $\Box P$ "necessary P". It is of note that depending on the context, and thus employed logic, the notion of "necessity and possibility" can be interpreted differently e.g. as "obligation and permission" for deontic logic, or even as "knowledge" in epistemic logic.

Syntactically *modal logic* accepts modal proposition, like $\Box(P \wedge Q)$, as well as non-modal proposition, such as $P \wedge Q$. For instance we can defined the language of *modal proposition logic* $\mathscr{M}$ over the language of *propositional logic* $\mathscr{L}$:

- If ϕ is a formula of $\mathscr{L}$, then ϕ is a formula of $\mathscr{M}$;
- If ϕ is a formula of $\mathscr{M}$, then $\Box\phi$ is too;
- if ϕ is a formula of $\mathscr{M}$, then $\Diamond\phi$ is too;

2.2 Possible words

The concept of possible worlds is a central idea in modal logic, which deals with necessity and possibility. Possible worlds are hypothetical scenarios or states of affairs that help in understanding and evaluating modal statements. They provide a framework for analyzing the truth of propositions relative to different circumstances, allowing us to reason about what could be true, what must be true, and what might not be true.

A possible world is a complete and consistent way the world could be. Each possible world represents a unique way things could have been, encompassing all aspects of reality within that scenario. The actual world, the real world as we experience it, is just one of these possible worlds.

2.3 Kripke semantics

Kripke semantics, named after the logician Saul Kripke, is a formal semantics for modal logic that uses possible worlds to interpret modal operators. It provides a rigorous framework for understanding how propositions can be true or false in different contexts or "possible worlds."

A *Kripke structure*, or *relational model* consists of three key components: a set of possible worlds, an accessibility relation, and a valuation function. The set of possible worlds, denoted as W. The accessibility relation, denoted as R, is a binary relation on W that specifies which worlds are accessible from which others; essentially, $R(w, w')$ indicates that world w' is considered a possible alternative to world w. This relation is crucial for defining the truth conditions of modal operators. Lastly, the valuation function, denoted as V, assigns truth values to atomic propositions in each possible world. Specifically, $V(P)$ is the set of worlds where proposition P holds true. Together, these components create a robust framework for evaluating the truth of modal statements, enabling a systematic approach to reasoning about necessity and possibility across different scenarios.

2.4 Metalogic

Metalogic is the study of the properties and foundations of logical systems. While traditional logic focuses on the structure of arguments and the validity of inference within a given logical system, metalogic examines the systems themselves from an external perspective. It deals with questions about the nature, consistency, completeness, soundness, and other meta-properties of these logical systems.

Key properties of logical systems include consistency, soundness, completeness, and decidability. Consistency refers to a logical system being free from contradictions, meaning it is impossible to derive both a statement and its negation within the system. Metalogic investigates methods to prove or disprove the consistency of various logical systems. Soundness ensures that all theorems derivable within the system are true in every model of the system, thus making the system's derivations reliable. Completeness is the property of a logical system wherein every statement that is true in all models of the system can be derived using the system's axioms and inference rules, guaranteeing that the system can express all truths within its scope. Decidability concerns whether there exists an effective method (algorithm) to determine if any given statement within the system is provable, with metalogic exploring the decidability of various logical systems and its implications for computational logic.

2.5 Orders of logic

Logic can be categorized into different levels or "orders" based on the complexity and types of variables and quantifiers they employ. The orders of logic provide a framework for understanding and analyzing different logical systems, each with varying degrees of expressiveness and complexity. Here is an overview of the primary orders of logic:

1. **Propositional Logic (Zero-order Logic):**
   - **Variables:** Propositional logic uses variables that represent entire propositions, which can be either true or false.
   - **Operators:** It employs logical connectives like AND ($\wedge$), OR ($\vee$), NOT ($\neg$, IMPLIES ($\rightarrow$), and IFF ($\Longleftrightarrow$).
   - **Expressiveness:** Propositional logic is the simplest form of logic, dealing only with the relationships between whole propositions without internal structure.
   - **Example:** $P \wedge (Q \rightarrow R) P \wedge (Q \rightarrow R)$, where P, Q, and R are propositions.
2. **First-order Logic (Predicate Logic):**
   - **Variables:** First-order logic introduces variables that can represent objects within a domain of discourse.
   - **Quantifiers:** It uses quantifiers such as universal quantifier ($\forall$) and existential quantifier ($\exists$) to express statements about all objects or some objects in the domain.
   - **Predicates:** It uses predicates to express properties of objects or relationships between objects.
   - **Expressiveness:** First-order logic is more expressive than propositional logic as it can represent statements about individual objects and their relationships.
   - **Example:** $\forall x(Human(x) \rightarrow Mortal(x))$, meaning "All humans are mortal."
3. **Second-order Logic:**
   - **Variables:** Second-order logic extends first-order logic by allowing quantification over predicates or sets.
   - **Quantifiers:** It can quantify over variables that represent sets or relations, not just individual objects.
   - **Expressiveness:** Second-order logic is significantly more expressive than first-order logic as it can express properties about properties or sets.
   - **Example:** $\forall P(P(a) \wedge \forall x(P(x) \rightarrow P(f(x))) \rightarrow \forall x P(x))$, meaning "If a property P holds for a and is preserved by function f, then P holds for all x."
4. **Higher-order Logic:**
   - **Variables:** Higher-order logics generalize second-order logic to allow quantification over functions, predicates, and relations of any order.
   - **Quantifiers:** These logics include quantifiers that range over higher-order entities like functions of functions, etc.
   - **Expressiveness:** Higher-order logics are even more expressive, capable of representing very complex statements and properties, but they are also more complex and harder to reason about computationally.

3. Exploration and deduction

3.1 Modal logic overview

Modal logics are commonly found in two main domain, those being Philosophy and Computer Sciences. In the first one they are used to describe knowledge inferences, or in the case of *Deontic Logic* to describe ethics and legal obligations. At the opposite side in Computer Sciences they are mostly used to infer over properties of discrete systems.

3.1.1 Deontic logic

Deontic logic is mostly based is generally conceived over natural languages such as English, thus giving us logical formulae with a sentences looking propositions. For example, we could have the proposition O(a person is moral), meaning that "A person is obligated to be moral".

In *Deontic logic* the modal operators are represented as O and P, for the *obligation* and *permission* respectively. Moreover, this logic use three axioms over those of standard propositional logic:

- $(\models A) \rightarrow (OA)$
- $O(AB) \rightarrow (OA \rightarrow OB)$
- $OA \rightarrow PA$

In a general manner we also add the *prohibition* defined as $\neg\Diamond$ or $\Box\neg$.

3.1.2 Computational Tree logic

Computational Tree Logic (CTL) is a form of temporal logic used to reason about the states and paths of computation in concurrent systems. It extends classical logic by incorporating temporal operators that allow statements to reference the ordering of events over time. This makes CTL particularly useful in the specification and verification of systems where the sequence and timing of actions are critical, such as in software and hardware design, formal verification, and model checking.

CTL combines the concepts of branching time with modal operators to express how properties of systems evolve over time. The key operators in CTL includes, *Path qualifiers*:

- A (for all paths): A property holds on all paths emanating from a given state.
- E (there exists a path): A property holds on at least one path emanating from a given state.

And *Temporal Operators*:

- X (next): A property holds in the next state along a path.
- F (eventually): A property holds at some future state along a path.
- G (globally): A property holds at all future states along a path.
- U (until): A property P holds until another property Q holds along a path.

The syntax of CTL formulas is built from atomic propositions, logical connectives, path quantifiers, and temporal operators. A typical CTL formula might look like $A[G(request \rightarrow F(grant))]$, which means "on all paths, globally, if a request occurs, then eventually it will be granted."

3.1.3 Past + CTL

Past Computational Tree Logic (Past + CTL) extends the standard Computational Tree Logic (CTL) by incorporating past-time operators. While CTL allows reasoning about future states of a system, Past + CTL enables reasoning about both future and past states. This dual temporal perspective enhances the expressiveness and utility of the logic, particularly in systems where the history of states is as important as their future behavior.

In addition to the standard CTL operators, Past + CTL includes past-time operators that allow assertions about the history of states. The key operators in PCTL include:

- Y (yesterday): A property held in the previous state along a path.
- H (historically): A property has held at all past states along a path.
- O (once): A property held at some past state along a path.
- S (since): A property P has held since another property Q was true along a path.

It is important to note that those operators can be defined using CTL operators meaning that this logic simplifies the writing of logical properties about the past, but does not enhance the total expresivity of the logic. Moreover a Past+CTL formula might look like $A[G(request \rightarrow F(grant) \wedge H(log))]$, meaning "on all paths, globally, if a request occurs, then eventually it will be granted, and it has always been logged."

3.1.4 PCTL

Probabilistic Computational Tree Logic (PCTL) extends the standard Computational Tree Logic (CTL) by incorporating probabilistic operators, enabling the formal specification and verification of properties in stochastic systems. PCTL is particularly useful for reasoning about systems where uncertainty and probabilistic behaviors are inherent, such as in randomized algorithms, communication networks, and biological systems.

In addition to the standard CTL operators, PCTL introduces probabilistic operators that allow for quantitative assertions about the likelihood of certain events. The key elements in PCTL includes *Probabilistic operators*:

- $P_{\leq p}[\phi]$: The probability of formula ϕ being true is less than or equal to p.
- $P_{\geq p}[\phi]$: The probability of formula ϕ being true is greater than or equal to p.

These operators allow statements like $P_{\geq 0.95}[F(\phi)]$, meaning "the probability that ϕ will eventually hold is at least 0.95."

3.2 Metalogic influences on Modal logic

We know from the work of Jaakko Hintikka that it is possible to find equisatisfiable formulae in second-order logic to higher-order logic, meaning that the two are equivalent under certain conditions. Also using Henkin semantics we can reduce second-order logic to many-sorted first-order logic, pushing us to think there is a way to express complex problems using more simple logics.

In our previous exploration, every cited logics is a first-order modal logic, implying that, they have the same expressivity. Basing ourselves on this, and the aforementioned property, we proposed the aftermentioned methodology.

4. Developed Methodology

4.1 Intuition

Intuitively, as an idea becomes more concrete, the type of modal logic used to describe it shifts to suit its complexity and application. For general policies or guidelines, Deontic Logic is most appropriate because it deals with what is permissible or obligatory, fitting well with broad, theoretical concepts. As these ideas accumulate more specific information and become more applied, it becomes necessary to transition to more dynamic logics. Here, Computation Tree Logic (CTL) is useful for mapping out different scenarios and potential outcomes, making it ideal for intermediate stages of idea development. Finally, as the idea fully matures into a concrete application, Probabilistic Computation Tree Logic (PCTL) comes into play. PCTL extends CTL by incorporating probabilities, which is essential for expressing and handling the uncertainties inherent in practical applications. This progression from Deontic Logic through CTL to PCTL illustrates a natural flow from the abstract to the applied, reflecting the increasing specificity and complexity of ideas.

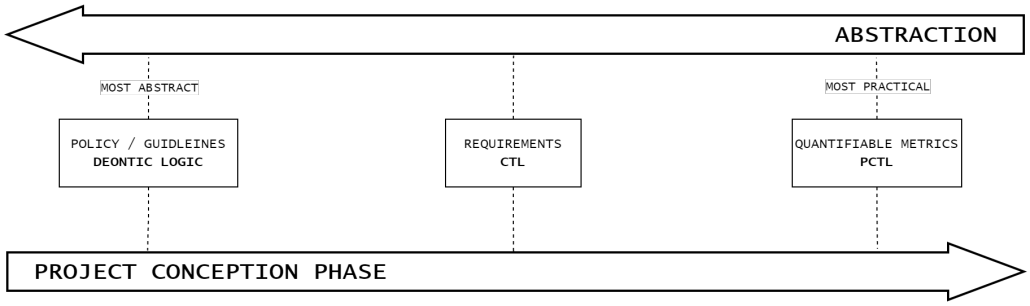

Figure 1: Intuitive use of the modal logics over the execution course of a project conception.

4.2 Formalization

4.2.1 From natural language to formal logic

When transitioning from natural language to formal logic, specifically Deontic Logic, its operators formalization offer a straightforward way to formalize abstract ideas. Deontic Logic, with its focus on obligations (O), permissions (P), and prohibitions (F), provides a clear framework for articulating broad, theoretical concepts. By using these operators, we can convert general statements and guidelines expressed in natural language into a more structured and precise logical format. This process is relatively direct as Deontic Logic's constructs naturally align with the ways we typically discuss rules and norms in everyday language. Therefore, it acts as an effective bridge between informal, conceptual discussions and their formal logical representation, ensuring that the essence of the original idea is preserved while gaining the clarity and rigor required for theoretical analysis.

4.2.2 From a Logic to another

As an idea goes from a general and abstract guideline into its implementation to a dynamic process, a transition from deontic logic to a dynamic one occurs. Moving from Deontic Logic to Computation Tree Logic (CTL) involves translating the concepts of obligations, permissions, and prohibitions into CTL's syntax and operators.

Considering a property p :

- $O(p)$ can be translated into CTL as $AG(p)$ i.e. in all possible executions of the process p is verified.
- $P(p)$ can be translated as EX meaning that there is an eventuality of p being true in an execution.
- $F(p)$ becomes $AG(\neg p)$ meaning that p can not be true at any point in any execution path of the the process / system.

4.2.3 Expressivity limitations

As more information is added going form a less expressive logic to a more expressive one can be done without any loss of information, naturally, the question rose as to weather it was possible to go the other way around, and the only way to do so would be to act on the statement p, having it include the information that would otherwise be lost.

4.3 Example

To illustrate this methodology let us consider a very simplistic system, let's consider designing a simple calculator, some of the general ideas one would want to follow when designing it in the early stages would be the following :

- The calculator must always provide correct arithmetic results.
- The calculator may have an advanced mode that includes functions like roots and powers.
- The calculator must never allow division by zero.

Formally, in deontic logic these statements, would translate to the following :

- O(provide correct arithmetic results) (called correct results thereafter)
- P(advanced mode)
- F(division by zero)

Then when planning the implementation and architecture of the system, and execution paths, these ideas have to be included in the dynamic flow of an execution, leading to the following statements :

- AG(correct results) i.e. no matter the execution path, the output should be arithmetically correct
- EX(advanced mode) meaning that when following some execution path the advanced mode can be entered
- AG(¬division by zero)

Finally when the system, and in order to adapt to real world constrains preventing from dealing with absolutes some quantifiable thresholds may be wished for, in order to evaluate it; hence the inclusion of quantities such as probabilities with PCTL leading to the following :

- $P >= 0.9999[AG(\text{correct results})]$ i.e. 99.99% of the answers should be arithmetically correct.
- $P = 0.5[EX(\text{advanced mode})]$ The chance of entering advanced mode or not are considered to be equal
- $P = 0.001[AG(\neg\text{division by zero})]$ Divisions by 0 should only happen less than 0.1% of the time.

5. Project Evolution

5.1 Initial Expectations

At the start of our semester project, we were stepping into new territory, as we were not familiar with modal logics. Our focus was on exploring Probabilistic Computation Tree Logic (PCTL), Deontic Logic, and combining Past operators with CTL. Early on, we felt there might be a way to transition from one type of logic to another, even though we were not sure exactly how.[1] It was also clear from the beginning that our work would lead to a final report. This report was intended to clearly summarize our discoveries and thoughts throughout the project.

At first, our understanding of modal logics was quite elementary, confined to the theoretical knowledge gained throughout the previous semester. The task required us to not only understand but also apply concepts from PCTL, Deontic Logic, and Past + CTL in novel ways. This required going through existing literature and precedents in the field, allowing us to gradually build a foundational understanding suitable for further exploration. Our initial explorations were guided by a series of hypotheses about how these logics could potentially interrelate, providing a structured approach to our investigation.

5.2 Expectations Evolution

As our project progressed and we dug deeper into the subject matter, our initial expectations had to evolve. Our first intuition about the potential to seamlessly transition between different modal logics proved more complex and challenging to formalize than we had anticipated. This realization emerged as we encountered the intricate nuances and theoretical depth of PCTL, Deontic Logic, and the integration of Past operators with CTL. The task of developing a formal method to translate statements from one logic to another initially seemed feasible, but it quickly became apparent that this would require a much more extensive effort than we could accommodate within the scope of our semester project.

Consequently, we shifted our focus towards gaining a broader and more profound understanding of these modal logics. We concentrated on studying their properties, hierarchical structure, and levels of expressivity. This approach allowed us to appreciate the unique characteristics and applications of each logic more fully. As we acquired this enriched perspective, our goal evolved from attempting direct translations to exploring the possibility of creating a logical progression or 'pipeline' from the more abstract logics, like Deontic Logic, to those with practical applications, such as PCTL. This adjusted objective aimed not only to enhance our comprehension but also to lay down a foundation for future explorations that could potentially establish more concrete connections between these modal logics.

5.3 Retrospective

The entire process of our semester project offers a valuable perspective on what we gained from organizing and conducting this in-depth exploration into modal logics. Throughout the project, one of the most significant learnings was how to manage a complex research task effectively. We developed a systematic approach to tackling theoretical content, which involved setting clear milestones, frequent team discussions, and iterative refinement of our research methods. This structured methodology not only kept the project on track but also ensured that each phase was executed with thoroughness and precision.

Additionally, we learned the importance of flexibility in research. As our understanding evolved and our initial expectations were challenged, we adapted our objectives and methods accordingly. This adaptability was crucial in maintaining the relevance and effectiveness of our study, allowing us to explore more fruitful avenues of inquiry.

The collaboration aspect of the project also provided us with insights into the dynamics of working as a team on a complex academic task. We honed our communication skills, learned to leverage each other's strengths, and supported one another through challenges, which proved invaluable for personal and professional growth.

[1] This initial idea lead to a brainstorming session on the classromm's whiteboard rather confusing for those watching.

6. Conclusion

Our project has provided us with valuable insights into the process of transforming broad, every day ideas into precise, formal statements through the application of various types of logic. By employing Deontic logic, we have been able to articulate general rules and obligations. Computational Tree Logic (CTL) has enabled us to map these rules over time, capturing the dynamic nature of systems. Probabilistic Computational Tree Logic (PCTL) has further allowed us to incorporate probabilities and uncertainties, providing a more nuanced and realistic modeling framework.

Through this exploration, we have found that our method is not only effective for understanding how to apply these logics but also offers promising directions for future research and development. One such direction is the refinement and integration of Deontic logic with probabilistic elements, enhancing our ability to model complex systems where rules and uncertainties intersect. This could lead to more sophisticated and accurate representations of real-world scenarios.

Another exciting avenue is the application of our logical framework to practical systems beyond theoretical models. Testing and implementing our logic in software and hardware systems could reveal its robustness and effectiveness in real-world applications. This would provide practical validation and potentially lead to improvements in system design and verification processes.