MORE ABOUT MODAL LOGIC

1. Finite model property
2. Decidability
3. Translation to predicate logic
4. Multi-modal logics
5. Computational complexity


Semi-Decidability

- If there is a proof system (for example Hilbert-style proof theory) for logic L, then L is semi-decidable: there is an algorithm that halts and says “yes” if the formula given as input is L-valid (but might not halt if the formula is not L-valid).

Proof sketch: the algorithm starts enumerating L-proofs in some systematic way and halts if a proof for the formula is found.

⇒ The set of L-valid formulas is recursively enumerable.

1. Finite Model Property

Decidability

- Next we consider computational properties of modal logics. As usual we use as the formal model of computation Turing machines. Hence, when saying that there is an algorithm computing/solving some problem we mean that there is a Turing machine capable of computing/solving the problem.

- We say that logic L is decidable, if there is an algorithm such that given a formula as input it decides whether the formula is L-valid or not.

- Such an algorithm is called a decision procedure for L.

Finiteness Property

Definition. Modal logic L has the finite model property if for every formula P which is not L-valid, there is a finite L-model that has a world where P is false.

- If a logic L has the finite model property, then the set of formulas that are not L-valid is semi-decidable.

Proof sketch. Enumerate in some systematic way finite models from small models to bigger ones and check for each of them whether the model has a world where P is false.

- Hence, if a modal logic L has a proof system and the finite model property, then L is decidable.

- How can we show that a logic has the finite model property?

⇒ Filtration.
**Example. Filtration for Modal Logic K**

**Theorem.** Modal logic K has the finite model property.

**Proof.** Assume that a formula $Q$ is not K-valid. Hence, there is a model $M = (S,R,v)$ and a world $s_0 \in S$ such that $M,s_0 \not\models Q$.

We construct a finite (quotient) model $|M|$ from $M$ by filtration.

Let $\text{Sub}(Q)$ be the set of all subformulas of $Q$.

- Define an equivalence relation $\sim$ for the set $S$: $s \sim t$ if for all $P \in \text{Sub}(Q)$, $M,s \models P$ iff $M,t \models P$.
- Let $|S| = \{ s : t \sim s \}$ and $|S| = \{ |s| : s \in S \}$. Now $|S|$ is finite because it has at most $2^n$ elements where $n$ is the number of subformulas of $Q$: every $|s| \in |S|$ is a set of worlds such that exactly the same subset of subformulas of $Q$ are true in each world in $|s|$.

**Filtration—cont’d**

Consider now a model $|M| = (|S|,|R|,|v|)$ where the relation $|R|$ is defined as follows:

$|v|(|S|,|P|) = v(s,P)$.

We show by structural induction that for every $P \in \text{Sub}(Q)$

$M,s \models P$ iff $|M|,|s| \models P$.

- $P$ is an atomic proposition: $|v|(|S|,P) = v(s,P)$.
- $\neg P$: $M,s \models \neg P$ iff $M,s \not\models P$ (IH)

$|M|,|s| \models P$ iff $|M|,|s| \not\models \neg P$.

$P \rightarrow Q$: can be proved in a similar way.

**2. Decidability**

- If we can give an upper bound on the size of the counter-model, the logic is decidable.
- This observation does not lead to a very efficient decision procedure.
- The tableau method provides a more efficient approach.

**Example.** We can show that logic K is decidable by showing that the tableau method for K provides a decision procedure that halts on every formula.

For this argument we need König’s lemma:

**Lemma.** If a tree has an infinite number of nodes but every node has a finite number of child nodes, then the tree has an infinite branch.
A decision procedure deciding whether a formula $P$ is $K$-valid:

1. Take as the root of the tableau $(1)\neg P$.
2. Until the tableau is closed or all formulas have been marked used:
   2.1 Choose the uppermost unused node $\sigma Q$ in the tableau.
   2.2 If $Q$ is not a literal, then for each open branch $\theta$ containing $\sigma Q$ do:
      - If $\sigma Q$ is of the form $\sigma \neg Q'$, extend $\theta$ by the node $\sigma Q'$.
      - If $\sigma Q$ is of the form $\sigma \alpha$, extend $\theta$ by nodes $\sigma \alpha_1$ and $\sigma \alpha_2$.
      - If $\sigma Q$ is of the form $\sigma \beta$, extend $\theta$ to two branches one containing $\sigma \beta_1$ and the other $\sigma \beta_2$.
      - If $\sigma Q$ is of the form $\sigma \Box P$, extend $\theta$ by $\sigma \Box P$ with some $\sigma \alpha$ which is unrestricted on $\theta$ and then with $\sigma \alpha X$ for every $\sigma \Box X$ that appears on the branch $\theta$.
      - If $\sigma Q$ is of the form $\sigma \Diamond P$, extend $\theta$ with $\sigma \alpha P$ for every $\sigma \alpha$ which is available on $\theta$ if $\sigma \alpha P$ is not already contained in $\theta$.
   2.3 Mark $\sigma Q$ used.

Proposition. The procedure above halts for every formula $P$ after a finite number of steps.

Proof. Assume that there is a formula $P$ for which the procedure does not halt after a finite number of steps.

- Then the procedure constructs an infinite $K$-tableau, i.e., a tree where each node has at most child nodes.
- By König’s lemma the tableau has an infinite branch $\theta$.
- In $\theta$ there is an infinite number of prefixed formulas, which are all different (we are assuming that the procedure extends the branch with a prefixed formula only if it is not contained in the branch).
- For every prefix $\sigma$, if $\sigma Q$ occurs on a branch $\theta$, then $Q$ is a subformula of $P$ or the negation of a subformula. As $P$ has only a finite number of subformulas, each prefix can occur only a finite number of times.
- Hence, the branch $\theta$ must contain an infinite number of different prefixes.

Proof—cont’d

There are two possibilities:

(i) The branch $\theta$ has an infinite number of prefixes of some length $n$.
   - Let $n$ be the smallest natural number such that $\theta$ has infinitely many prefixes of length $n$.
   - Now it must be the case that $n \neq 1$ because the only prefix of length one is $(1)$ and it occurs only finitely many times in $\theta$.
   - If $n > 1$, then the only way of producing a prefix of length $n$ is to use $\neg\Box$ or $\Box$ rule to a formula $\sigma Q$ where $\sigma$ is of length $n-1$.
   - Hence, if the branch contains an infinite number of prefixes of length $n$, then it must have an infinite number of prefixes of length $(n-1)$, which is in contradiction with the choice of $n$.

⇒ Hence, possibility (i) cannot be the case and, thus, (ii) $\theta$ has a finite number of prefixes of length $n$ for every natural $n$.

- As there is an infinite number of prefixes in $\theta$, it must contain prefixes of infinitely many different lengths.
- We show that this is impossible by establishing that for every formula $\sigma Q$ in $\theta$ the following property $L$ holds:
  the sum of the length of $\sigma$ with the number of modal operators in $\sigma Q$ is always at most $1 + m$, where $m$ is the number modal operators in $P$.
- Clearly ($L$) holds for the root of the tableau $(1)\neg P$.
- If a formula is obtained by $\alpha$ or $\beta$ rules, then ($L$) holds also for any new prefixed formula obtained by the rules.
Proof—cont’d

- If the formula has been obtained using the □ or ±□ rule, it is of the form σkQ and it is derived from a formula of the form σQ'. Assume that (L) holds for σQ'. Now the length of the prefix σk is the length of the prefix σ + 1 but the formula Q has one modal operator less than the formula Q'. Hence, (L) holds also for σkQ.
- Since (L) holds for every prefixed formula on the branch, the branch can contain only prefixes of the length at most m + 1.
  \[ \implies \text{Hence, the remaining possibility (ii) leads also to contradiction and, hence, the procedure halts on every formula P.} \]
- A corresponding decision procedure works for modal logics where the transitivity of the frames is not required.
- For logics which assume transitivity a stronger halting condition is needed, i.e., a condition for stopping of expanding branches.

3. Translation to Predicate Logic

Modal logic can be translated to a restricted subset of predicate logic where we use

- for every atomic proposition P, a one-argument predicate symbol P;
- two variables x_1 and x_2;
  (notation: if x one of the variables, then x' is the other.)
- two-place predicate symbol R.

Definition. Let \( \tau \) be a mapping from modal formulas and variables to formulas in predicate logic such that

1. \( \tau(\top, x) = \top; \tau(\bot, x) = \bot; \)
2. \( \tau(P, x) = P(x) \) for atomic propositions \( P; \)
3. \( \tau(\neg P, x) = \neg \tau(P, x); \)
4. \( \tau(P \rightarrow Q, x) = \tau(P, x) \rightarrow \tau(Q, x); \)
5. \( \tau(\Box P, x) = \forall x'(R(x, x') \rightarrow \tau(P, x')); \)

Example. \( \neg \neg P \rightarrow \neg \neg P, x_1) = \neg \neg \neg P, x_1) = \tau(\neg \neg P, x_1) \)
\[ = \tau(\neg P, x_1) \rightarrow \forall x_2(R(x_1, x_2) \rightarrow \tau(\neg P, x_2)) \]
\[ = \neg \forall x_2(R(x_1, x_2) \rightarrow \tau(P, x_2)) \rightarrow \forall x_2(R(x_1, x_2) \rightarrow \neg \tau(P, x_2)) \]
\[ = \neg \forall x_2(R(x_1, x_2) \rightarrow \tau(P, x_2)) \rightarrow \forall x_2(R(x_1, x_2) \rightarrow \tau(P, x_2)) \]
\[ = \neg \forall x_2(R(x_1, x_2) \rightarrow \tau(P, x_2)) \rightarrow \forall x_2(R(x_1, x_2) \rightarrow \tau(P, x_2)) \]
\[ = \neg \forall x_2(R(x_1, x_2) \rightarrow \tau(P, x_2)) \rightarrow \forall x_2(R(x_1, x_2) \rightarrow \tau(P, x_2)) \]

Mapping \( \tau \) Preserves Validity

Theorem. Let \( P \) be a modal formula.

1. \( P \) is K-valid \( \forall x_1 \tau(P, x_1) \) is valid in predicate logic.
2. \( \Sigma \models_K \emptyset \implies \Sigma \models \tau(\Sigma) \models_{cl} \forall x_1 \tau(P, x_1), \)
   where \( \tau(\Sigma) = \forall x_1 \tau(Q, x_1) \mid Q \in \Sigma \) and \( \models_{cl} \) is the logical consequence relation in predicate logic.
   - Modal logic \( T: \)
     \( P \) is T-valid \( \models_{cl} \forall x_1 \tau(P, x_1). \)
   - Modal logic \( S5: \)
     \( P \) is S5-valid if \( \Sigma_{S5} \models_{cl} \forall x_1 \tau(P, x_1) \)
     where \( \Sigma_{S5} = \forall x_1 R(x_1, x_1), \forall x_1 \forall x_2 (R(x_1, x_2) \rightarrow R(x_2, x_3)), \)
     \( \forall x_1 \forall x_2 \forall x_3 (R(x_1, x_2) \land R(x_2, x_3) \rightarrow R(x_1, x_3)). \)
4. Multi-modal Logic

**Example.** Multi-agent logic of knowledge $S5_n$:

- $n$ agents and corresponding knowledge operators $K_i, i = 1, \ldots, n$. For example, formulas of the form $K_1K_2\neg K_1P$ allowed.
- Models are tuples of the form $(S, R_1, \ldots, R_n, v)$ and $M, s \models K_iP$ iff $M, t \models P$ for every $t \in S$ such that $sR_it$.
- A (Hilbert-style) proof system consists of $S5$ axioms for every $K_i$.
- Everybody knows ($EP$):
  $M, s \models EP$ iff $M, s \models K_iP$ for all $i = 1, \ldots, n$.
- Axiom: $EP \iff K_1P \land \cdots \land K_nP$.

---

5. Computational Complexity

**Different computational problems**

- Model checking (Is a formula true in a model?)
- Satisfiability (Does a formula have a model where it is true?)
- Validity
- Logical consequence

We consider here model checking and satisfiability because

1. a formula $P$ is valid iff $\neg P$ is not satisfiable;
2. $\{\} \models Q_1, \ldots, Q_n \implies P$ iff $Q_1 \land \cdots \land Q_n \rightarrow P$ is valid iff $Q_1 \land \cdots \land Q_n \land \neg P$ is not satisfiable.

---

**Common Knowledge: $CP$**

- $M, s \models CP$ iff $M, s \models E^kP$ for every $k = 1, 2, \ldots$
  $\implies M, s \models CP$ iff $M, t \models P$ for each $t \in S$ such that the world $i$ is $C$-accessible from the world $s$ (i.e., there is $k \geq 1$ and a sequence $(s = )t_0, t_1, \ldots, t_k(= t)$ where for every $j = 0, \ldots, k - 1, t_jR_it_{j+1}$ holds for some $i, 1 \leq i \leq n$.)
- Axiom: $CP \rightarrow E(P \land CP)$.
- Inference rule:
  \[
  \frac{P \rightarrow E(Q \land P)}{P \rightarrow CQ}
  \]
- **Example.** $CP \rightarrow K_1K_2\cdots K_nP$ is $S5_n$-valid

---

**Computational Complexity of Model Checking**

- Whether a formula is true in a given world of a model can be checked in polynomial time (w.r.t. the size of the model and length of the formula) for all modal logics we have considered so far.
- However, not every modal logic has this property (for example, many temporal logics do not).
- All problems solvable in polynomial time can be reduced to evaluating a propositional formula:
  Evaluating the truth value of a Boolean circuit is a $P$-complete problem!
Computational Complexity of Satisfiability

- Deciding satisfiability is NP-complete for propositional logic.
- All normal modal logics contain propositional logic as a special, so deciding satisfiability for them is NP-hard.
- For modal logics S5, KD45 deciding satisfiability is NP-complete. For these logics, non-valid formulas have small counter-models (the number of worlds is at most the number of subformulas which implies that deciding satisfiability is in NP).
- For modal logics K, T, S4 the problem is PSPACE-complete: in these logics counter-models can be of exponential size.
- For logics S5n, KD45n the problem is PSPACE-complete.
- For modal logics S5C, KD45C the problem is EXPTIME-complete.

Summary

- Modal logics have varying computational properties.
- Decidability can often be established by providing a proof system and showing that the logic has the finite model property.
- More efficient decision procedures can be obtained using the tableau method.
- Many modal logics can be translated in a systematic way to a restricted subset of predicate logic.
- The one modal operator case can be generalized to the multi-modal case in the possible world semantics by introducing an accessibility relation to each of the modal operators.
- Modal logics differ in their inherent computational complexity.