IA008: Computational Logic

6. Modal Logic

Achim Blumensath blumens@fi.muni.cz

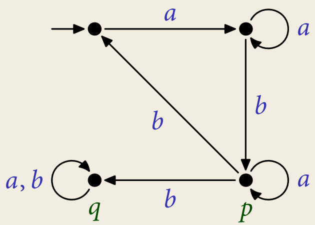
Faculty of Informatics, Masaryk University, Brno

Basic Concepts

Transition Systems

directed graph $\mathfrak{S} = \langle S, (E_a)_{a \in A}, (P_i)_{i \in I}, s_0 \rangle$ with

- **states** S
- **initial state** $s_0 \in S$
- **edge relations** E_a with **edge colours** $a \in A$ (‘**actions**’)
- **unary predicates** P_i with **vertex colours** $i \in I$ (‘**properties**’)



Modal logic

Propositional logic with **modal operators**

- $\langle a \rangle \varphi$ ‘there exists an a-successor where φ holds’
- $[a]\varphi$ ‘φ holds in every a-successor’

Notation: $\Diamond\varphi, \Box\varphi$ if there are no edge labels

Formal semantics

$$
\begin{aligned}
\mathfrak{S}, s \models P &\quad :\text{iff} \quad s \in P \\
\mathfrak{S}, s \models \varphi \wedge \psi &\quad :\text{iff} \quad \mathfrak{S}, s \models \varphi \text{ and } \mathfrak{S}, s \models \psi \\
\mathfrak{S}, s \models \varphi \vee \psi &\quad :\text{iff} \quad \mathfrak{S}, s \models \varphi \text{ or } \mathfrak{S}, s \models \psi \\
\mathfrak{S}, s \models \neg\varphi &\quad :\text{iff} \quad \mathfrak{S}, s \not\models \varphi \\
\mathfrak{S}, s \models \langle a \rangle \varphi &\quad :\text{iff} \quad \text{there is } s \to^a t \text{ such that } \mathfrak{S}, t \models \varphi \\
\mathfrak{S}, s \models [a]\varphi &\quad :\text{iff} \quad \text{for all } s \to^a t\,,\ \text{we have } \mathfrak{S}, t \models \varphi
\end{aligned}
$$

Examples

$P \wedge \Diamond Q$	'The state is in P and there exists a transition to Q.'
$[a]\bot$	'The state has no outgoing a-transition.'

Interpretations

- **Temporal Logic** talks about time:
  - states: points in time (discrete/continuous)
  - $\Diamond \varphi$ 'sometime in the future φ holds'
  - $\Box \varphi$ 'always in the future φ holds'
- **Epistemic Logic** talks about knowledge:
  - states: possible worlds
  - $\Diamond \varphi$ 'φ might be true'
  - $\Box \varphi$ 'φ is certainly true'

Examples: Temporal Logic

system $\mathfrak{S} = \langle S, \leq, \bar{P} \rangle$

- "P never holds."

Examples: Temporal Logic

system $\mathfrak{S} = \langle S, \leq, \bar{P} \rangle$

- "P never holds."

  $$\neg \Diamond P$$

- "After every P there is some Q."

Examples: Temporal Logic

system $\mathfrak{S} = \langle S, \leq, \bar{P} \rangle$

- "P never holds."

  $$\neg \Diamond P$$

- "After every P there is some Q."

  $$\Box(P \rightarrow \Diamond Q)$$

- "Once P holds, it holds forever."

Examples: Temporal Logic

system $\mathfrak{S} = \langle S, \leq, \bar{P} \rangle$

- "P never holds."

  $\neg \Diamond P$

- "After every P there is some Q."

  $\Box(P \to \Diamond Q)$

- "Once P holds, it holds forever."

  $\Box(P \to \Box P)$

- "There are infinitely many P."

Examples: Temporal Logic

system $\mathfrak{S} = \langle S, \leq, \bar{P} \rangle$

- "P never holds."

  $\neg \Diamond P$

- "After every P there is some Q."

  $\Box(P \to \Diamond Q)$

- "Once P holds, it holds forever."

  $\Box(P \to \Box P)$

- "There are infinitely many P."

  $\Box \Diamond P$

Translation to first-order logic

Proposition

For every formula φ of propositional modal logic, there exists a formula $\varphi^*(x)$ of first-order logic such that

$$\mathfrak{S}, s \vDash \varphi \quad \text{iff} \quad \mathfrak{S} \vDash \varphi^*(s)\,.$$

Proof

Translation to first-order logic

Proposition

For every formula φ of propositional modal logic, there exists a formula $\varphi^*(x)$ of first-order logic such that

$$\mathfrak{S}, s \vDash \varphi \quad \text{iff} \quad \mathfrak{S} \vDash \varphi^*(s)\,.$$

Proof

$$\begin{aligned}
P^* &:= P(x) \\
(\varphi \wedge \psi)^* &:= \varphi^*(x) \wedge \psi^*(x) \\
(\varphi \vee \psi)^* &:= \varphi^*(x) \vee \psi^*(x) \\
(\neg\varphi)^* &:= \neg\varphi^*(x) \\
(\langle a\rangle\varphi)^* &:= \exists y[E_a(x,y) \wedge \varphi^*(y)] \\
([a]\varphi)^* &:= \forall y[E_a(x,y) \rightarrow \varphi^*(y)]
\end{aligned}$$

Bisimulation

$\mathfrak{S}$ and $\mathfrak{T}$ transition systems

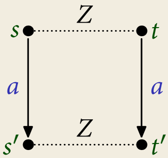
$Z \subseteq S \times T$ is a **bisimulation** if, for all $\langle s, t \rangle \in Z$,

(local) $s \in P \iff t \in P$

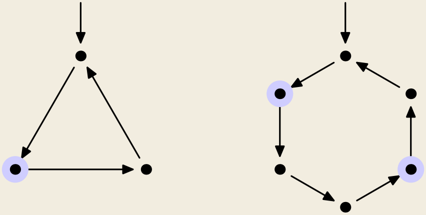
(forth) for every $s \to^a s'$, exists $t \to^a t'$ with $\langle s', t' \rangle \in Z$,

(back) for every $t \to^a t'$, exists $s \to^a s'$ with $\langle s', t' \rangle \in Z$.

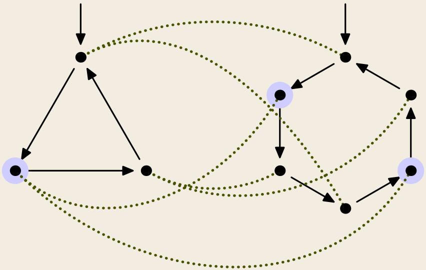
$\mathfrak{S}, s$ and $\mathfrak{T}, t$ are **bisimilar** if there is a bisimulation Z with $\langle s, t \rangle \in Z$.



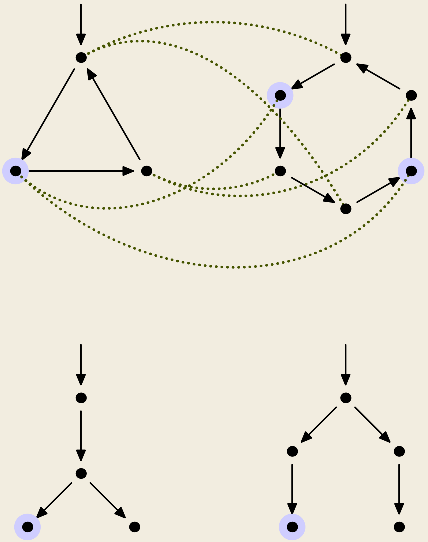
Examples



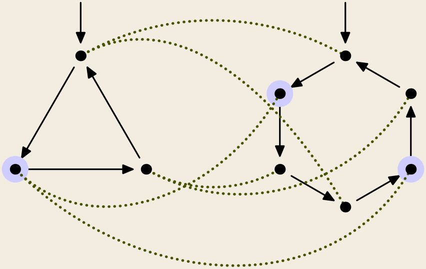
Examples

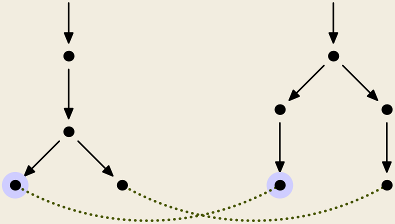


Examples

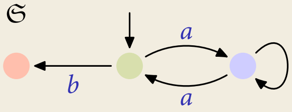


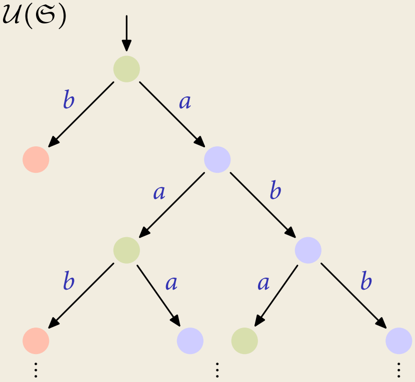
Examples





Unravelling





Lemma
$\mathfrak{S}$ and $\mathcal{U}(\mathfrak{S})$ are bisimilar.

Bisimulation invariance

Theorem

Two **finite** transition systems $\mathfrak{S}$ and $\mathfrak{T}$ are **bisimilar** if, and only if,

$$\mathfrak{S} \vDash \varphi \quad \Leftrightarrow \quad \mathfrak{T} \vDash \varphi, \qquad \text{for every modal formula } \varphi.$$

Definition

A formula $\varphi(x)$ is **bisimulation invariant** if

$$\mathfrak{S}, s \sim \mathfrak{T}, t \quad \text{implies} \quad \mathfrak{S} \vDash \varphi(s) \Leftrightarrow \mathfrak{T} \vDash \varphi(t).$$

Theorem

A first-order formula φ is equivalent to a **modal formula** if, and only if, it is **bisimulation invariant.**

First-Order Modal Logic

Syntax

first-order logic with modal operators $\langle a \rangle \varphi$ and $[a]\varphi$

First-Order Modal Logic

Syntax

first-order logic with modal operators $\langle a \rangle \varphi$ and $[a]\varphi$

Models

transistion systems where each state s is labelled with a Σ-structure $\mathfrak{A}_s$ such that

$$s \to^{a} t \quad \text{implies} \quad A_s \subseteq A_t$$

First-Order Modal Logic

Syntax

first-order logic with modal operators $\langle a \rangle \varphi$ and $[a]\varphi$

Models

transistion systems where each state s is labelled with a Σ-structure $\mathfrak{A}_s$ such that

$$s \to^a t \quad \text{implies} \quad A_s \subseteq A_t$$

Examples

- $\Box \forall x \varphi(x) \to \forall x \Box \varphi(x)$ is valid.
- $\forall x \Box \varphi(x) \to \Box \forall x \varphi(x)$ is not valid.

Tableaux

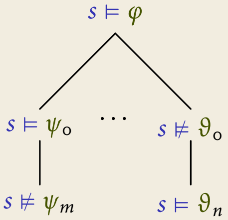
Tableau Proofs

Statements

$s \models \varphi \qquad s \not\models \varphi \qquad s \to^{a} t$

s, t **state labels,** φ a **modal formula**

Rules



Tableaux

Construction

A **tableau** for a formula φ is constructed as follows:

- start with $s_0 \nvDash \varphi$
- choose a branch of the tree
- choose a statement $s \vDash \psi / s \nvDash \psi$ on the branch
- choose a rule with head $s \vDash \psi / s \nvDash \psi$
- add it at the bottom of the branch
- repeat until every branch contains both statements $s \vDash \psi$ and $s \nvDash \psi$ for some formula ψ

Tableaux

Construction

A **tableau** for a formula φ is constructed as follows:

- start with $s_0 \nvDash \varphi$
- choose a branch of the tree
- choose a statement $s \vDash \psi / s \nvDash \psi$ on the branch
- choose a rule with head $s \vDash \psi / s \nvDash \psi$
- add it at the bottom of the branch
- repeat until every branch contains both statements $s \vDash \psi$ and $s \nvDash \psi$ for some formula ψ

Tableaux with premises Γ

- choose a branch, a state s on the branch, a premise $\psi \in \Gamma$, and add $s \vDash \psi$ to the branch

Rules

$s \vDash \neg\varphi$ followed by $s \nvDash \varphi$

$s \nvDash \neg\varphi$ followed by $s \vDash \varphi$

$s \vDash \varphi \wedge \psi$ followed by $s \vDash \varphi$, then $s \vDash \psi$

$s \nvDash \varphi \wedge \psi$ branches into $s \nvDash \varphi$ | $s \nvDash \psi$

$s \vDash \varphi \vee \psi$ branches into $s \vDash \varphi$ | $s \vDash \psi$

$s \nvDash \varphi \vee \psi$ followed by $s \nvDash \varphi$, then $s \nvDash \psi$

$s \vDash \varphi \rightarrow \psi$ branches into $s \nvDash \varphi$ | $s \vDash \psi$

$s \nvDash \varphi \rightarrow \psi$ followed by $s \vDash \varphi$, then $s \nvDash \psi$

$s \vDash \varphi \leftrightarrow \psi$ branches into ($s \vDash \varphi$, then $s \vDash \psi$) | ($s \nvDash \varphi$, then $s \nvDash \psi$)

$s \nvDash \varphi \leftrightarrow \psi$ branches into ($s \vDash \varphi$, then $s \nvDash \psi$) | ($s \nvDash \varphi$, then $s \vDash \psi$)

Rules

$$\begin{array}{c} s \models \langle a \rangle \varphi \\ | \\ s \to^a t \\ | \\ t \models \varphi \end{array} \qquad \begin{array}{c} s \not\models \langle a \rangle \varphi \\ | \\ t' \not\models \varphi \end{array} \qquad \begin{array}{c} s \models [a] \varphi \\ | \\ t' \models \varphi \end{array} \qquad \begin{array}{c} s \not\models [a] \varphi \\ | \\ s \to^a t \\ | \\ t \not\models \varphi \end{array}$$

$$\begin{array}{c} s \models \forall x \varphi \\ | \\ s \models \varphi[x \mapsto u] \end{array} \qquad \begin{array}{c} s \not\models \forall x \varphi \\ | \\ s \not\models \varphi[x \mapsto c] \end{array} \qquad \begin{array}{c} s \models \exists x \varphi \\ | \\ s \models \varphi[x \mapsto c] \end{array} \qquad \begin{array}{c} s \not\models \exists x \varphi \\ | \\ s \not\models \varphi[x \mapsto u] \end{array}$$

t a new state, t' every state with entry $s \to^a t'$ on the branch,
c a new constant symbol, u an arbitrary term

Example $\varphi \vDash \Box\varphi$

$$s \nvDash \Box\varphi$$

$$s \to t$$

$$t \nvDash \varphi$$

$$t \vDash \varphi$$

Example $\vDash \Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$

$$s \nvDash \Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$$

$$s \vDash \Box(\varphi \to \psi)$$

$$s \nvDash \Box\varphi \to \Box\psi$$

$$s \vDash \Box\varphi$$

$$s \nvDash \Box\psi$$

$$s \to t$$

$$t \nvDash \psi$$

$$t \vDash \varphi$$

$$t \vDash \varphi \to \psi$$

$$t \nvDash \varphi \qquad\qquad t \vDash \psi$$

Example $\vDash \Box \forall x \varphi \to \forall x \Box \varphi$

$$s \nvDash \Box \forall x \varphi \to \forall x \Box \varphi$$

$$s \vDash \Box \forall x \varphi$$

$$s \nvDash \forall x \Box \varphi$$

$$s \nvDash \Box \varphi[x \mapsto c]$$

$$s \to t$$

$$t \nvDash \varphi[x \mapsto c]$$

$$t \vDash \forall x \varphi$$

$$t \vDash \varphi[x \mapsto c]$$

Soundness and Completeness

Consequence

ψ is a **consequence** of Γ if, and only if, for all transition systems $\mathfrak{S}$,

$$\mathfrak{S}, s \vDash \varphi\,, \quad \text{for all } s \in S \text{ and } \varphi \in \Gamma\,,$$

implies that

$$\mathfrak{S}, s \vDash \psi\,, \quad \text{for all } s \in S\,.$$

Soundness and Completeness

Consequence

ψ is a **consequence** of Γ if, and only if, for all transition systems $\mathfrak{S}$,

$$\mathfrak{S}, s \vDash \varphi\,, \quad \text{for all } s \in S \text{ and } \varphi \in \Gamma\,,$$

implies that

$$\mathfrak{S}, s \vDash \psi\,, \quad \text{for all } s \in S\,.$$

Theorem

A modal formula φ is a consequence of Γ if, and only if, there exists a tableau T for φ with premises Γ where every branch is contradictory.

Complexity

Theorem

Satisfiability for propositional modal logic is in **deterministic linear space.**

Theorem

Satisfiability for first-order modal logic is **undecidable.**

Temporal Logics

Linear Temporal Logic (LTL)

Speaks about **paths.** $P \longrightarrow \bullet \longrightarrow \bullet \longrightarrow P, Q \longrightarrow Q \longrightarrow \bullet \longrightarrow \cdots$

Syntax

- atomic predicates $P, Q, \ldots$
- boolean operations $\wedge, \vee, \neg$
- next $X\varphi$
- until $\varphi U \psi$
- finally $F\varphi := \top U \varphi$
- generally $G\varphi := \neg F \neg \varphi$

Examples

FP	a state in P is reachable
GFP	we can reach infinitely many states in P
$(\neg P) U (P \wedge Q)$	the first reachable state in P is also in Q

Linear Temporal Logic (LTL)

Theorem

Let L be a set of paths. The following statements are equivalent:

- L can be defined in LTL.
- L can be defined in first-order logic.
- L can be defined by a star-free regular expression.

Linear Temporal Logic (LTL)

Theorem

Let L be a set of paths. The following statements are equivalent:

- L can be defined in LTL.
- L can be defined in first-order logic.
- L can be defined by a star-free regular expression.

Translation LTL to FO

$$\begin{aligned}
P^* &:= P(x) \\
(\varphi \wedge \psi)^* &:= \varphi^*(x) \wedge \psi^*(x) \\
(\varphi \vee \psi)^* &:= \varphi^*(x) \vee \psi^*(x) \\
(\neg\varphi)^* &:= \neg\varphi^*(x) \\
(X\varphi)^* &:= \exists y[x < y \wedge \neg\exists z(x < z \wedge z < y) \wedge \varphi^*(y)] \\
(\varphi U \psi)^* &:= \exists y[x \leq y \wedge \psi^*(y) \wedge \forall z[x \leq z \wedge z < y \rightarrow \varphi^*(z)]]
\end{aligned}$$

Linear Temporal Logic (LTL)

Theorem

Let L be a set of paths. The following statements are equivalent:

- L can be defined in LTL.
- L can be defined in first-order logic.
- L can be defined by a star-free regular expression.

Theorem

Satisfiablity of LTL formulae is **PSPACE-complete.**

Theorem

Model checking $\mathfrak{S}, s \vDash \varphi$ for LTL is **PSPACE-complete.** It can be done in

$$\textbf{time } \mathcal{O}\big(|S| \cdot 2^{\mathcal{O}(|\varphi|)}\big) \quad \text{or} \quad \textbf{space } \mathcal{O}\big((|\varphi| + \log|S|)^2\big).$$

(formula complexity: **PSPACE-complete;** data complexity: **NLOGSPACE-complete**)

Computation Tree Logic (CTL and CTL*)

Applies LTL-formulae to the branches of a tree.

Syntax (of CTL*)

- **state formulae** φ:

$$\varphi ::= P \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid A\psi \mid E\psi$$

- **path formulae** ψ:

$$\psi ::= \varphi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \neg\psi \mid X\psi \mid \psi U \psi \mid F\psi \mid G\psi$$

Examples

EFP	a state in P is reachable
AFP	every branch contains a state in P
$EGFP$	there is a branch with infinitely many P
$EGEFP$	there is a branch such that we can reach P from every of its states

Theorem

Satisfiability for CTL is **EXPTIME-complete.**

Model checking $\mathfrak{S}, s \vDash \varphi$ for CTL is **P-complete.** It can be done in

$$\textbf{time } \mathcal{O}\big(|\varphi| \cdot |S|\big) \quad \text{or} \quad \textbf{space } \mathcal{O}\big(|\varphi| \cdot \log^2 (|\varphi| \cdot |S|)\big).$$

(data complexity: **NLOGSPACE-complete**)

Theorem

Satisfiability for CTL is **EXPTIME-complete.**

Model checking $\mathfrak{S}, s \models \varphi$ for CTL is **P-complete.** It can be done in

$$\textbf{time } \mathcal{O}\big(|\varphi| \cdot |S|\big) \quad \text{or} \quad \textbf{space } \mathcal{O}\big(|\varphi| \cdot \log^2 (|\varphi| \cdot |S|)\big) .$$

(data complexity: **NLOGSPACE-complete**)

Theorem

Satisfiability for CTL* is **2EXPTIME-complete.**

Model checking $\mathfrak{S}, s \models \varphi$ for CTL* is **PSPACE-complete.** It can be done in

$$\textbf{time } \mathcal{O}\big(|S|^2 \cdot 2^{\mathcal{O}(|\varphi|)}\big) \quad \text{or} \quad \textbf{space } \mathcal{O}\big(|\varphi|(|\varphi| + \log|S|)^2\big) .$$

(formula complexity: **PSPACE-complete;** data complexity: **NLOGSPACE-complete**)

The modal μ-calculus (L_μ)

Adds recursion to modal logic.

Syntax

$$\varphi ::= P \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle a\rangle\varphi \mid [a]\varphi \mid \mu X.\varphi(X) \mid \nu X.\varphi(X)$$

(X positive in $\mu X.\varphi(X)$ and $\nu X.\varphi(X)$)

The modal μ-calculus (L_μ)

Adds recursion to modal logic.

Syntax

$$\varphi ::= P \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle a \rangle \varphi \mid [a]\varphi \mid \mu X.\varphi(X) \mid \nu X.\varphi(X)$$

(X positive in $\mu X.\varphi(X)$ and $\nu X.\varphi(X)$)

Semantics

$$F_\varphi(X) := \{\, s \in S \mid \mathfrak{S}, s \vDash \varphi(X) \,\}$$

$$\mu X.\varphi(X) : \quad X_0 := \varnothing, \quad X_{i+1} := F_\varphi(X_i)$$

$$\nu X.\varphi(X) : \quad X_0 := S, \quad X_{i+1} := F_\varphi(X_i)$$

The modal μ-calculus (L_μ)

Adds recursion to modal logic.

Syntax

$$\varphi ::= P \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle a \rangle \varphi \mid [a]\varphi \mid \mu X.\varphi(X) \mid \nu X.\varphi(X)$$

(X positive in $\mu X.\varphi(X)$ and $\nu X.\varphi(X)$)

Semantics

$$F_\varphi(X) := \{\, s \in S \mid \mathfrak{S}, s \vDash \varphi(X) \,\}$$

$$\mu X.\varphi(X): \quad X_0 := \emptyset, \quad X_{i+1} := F_\varphi(X_i)$$

$$\nu X.\varphi(X): \quad X_0 := S, \quad X_{i+1} := F_\varphi(X_i)$$

Examples

$\mu X(P \vee \Diamond X)$ a state in P is reachable

$\nu X(P \wedge \Diamond X)$ there is a branch with all states in P

Expressive power

Theorem

For every CTL*-formula φ there exists an equivalent formula φ^* of the modal μ-calculus.

Expressive power

Theorem

For every CTL*-formula φ there exists an equivalent formula φ^* of the modal μ-calculus.

Proof (for CTL)

$$
\begin{aligned}
P^* &:= P \\
(\varphi \wedge \psi)^* &:= \varphi^* \wedge \psi^* \\
(\varphi \vee \psi)^* &:= \varphi^* \vee \psi^* \\
(\neg \varphi)^* &:= \neg \varphi^* \\
(EX\varphi)^* &:= \Diamond \varphi^* \\
(AX\varphi)^* &:= \Box \varphi^* \\
(E\varphi U \psi)^* &:= \mu X[\psi^* \vee (\varphi^* \wedge \Diamond X)] \\
(A\varphi U \psi)^* &:= \mu X[\psi^* \vee (\varphi^* \wedge \Box X)]
\end{aligned}
$$

The modal μ-calculus (L_μ)

Theorem

A regular tree language can be defined in the **modal μ-calculus** if, and only if, it is **bisimulation invariant.**

Theorem

Satisfiability of μ-calculus formulae is **decidable** and complete for **exponential time.**

Model checking $\mathfrak{S}, s \vDash \varphi$ for the modal μ-calculus can be done in **time** $\mathcal{O}\big((|\varphi| \cdot |S|)^{|\varphi|}\big)$.

(The satisfiability algorithm uses tree automata and parity games.)

Fixed points

Theorem

Let $\langle A, \leq \rangle$ be a **complete** partial order and $f : A \to A$ **monotone**. Then f has a **least** and a **greatest fixed point** and

$$\mathrm{lfp}(f) = \lim_{n\to\infty} f^n(\bot) \quad \text{and} \quad \mathrm{gfp}(f) = \lim_{n\to\infty} f^n(\top)$$

Monotonicity

$\perp \leq f(\perp)$

Monotonicity

$\bot \leq f(\bot)$

$\Rightarrow f(\bot) \leq f(f(\bot))$

Monotonicity

$\bot \leq f(\bot)$
$\Rightarrow f(\bot) \leq f(f(\bot))$
$\Rightarrow f(f(\bot)) \leq f(f(f(\bot)))$

Monotonicity

$\bot \leq f(\bot)$

$\Rightarrow f(\bot) \leq f(f(\bot))$

$\Rightarrow f(f(\bot)) \leq f(f(f(\bot)))$

$\Rightarrow \ldots$

$\Rightarrow f^n(\bot)) \leq f^{n+1}(\bot)$

Monotonicity

$\bot \leq f(\bot)$

$\Rightarrow f(\bot) \leq f(f(\bot))$

$\Rightarrow f(f(\bot)) \leq f(f(f(\bot)))$

$\Rightarrow \ldots$

$\Rightarrow f^n(\bot)) \leq f^{n+1}(\bot)$

$\Rightarrow \bot \leq f(\bot) \leq f^2(\bot)) \leq \cdots \leq f^n(\bot) \leq \cdots$

Monotonicity

$\bot \leq f(\bot)$

$\Rightarrow f(\bot) \leq f(f(\bot))$

$\Rightarrow f(f(\bot)) \leq f(f(f(\bot)))$

$\Rightarrow \ldots$

$\Rightarrow f^n(\bot)) \leq f^{n+1}(\bot)$

$\Rightarrow \bot \leq f(\bot) \leq f^2(\bot)) \leq \cdots \leq f^n(\bot) \leq \cdots$

exists n with $f^n(\bot) = f^{n+1}(\bot)$

Monotonicity

$\bot \leq f(\bot)$
$\Rightarrow f(\bot) \leq f(f(\bot))$
$\Rightarrow f(f(\bot)) \leq f(f(f(\bot)))$
$\Rightarrow \ldots$
$\Rightarrow f^n(\bot)) \leq f^{n+1}(\bot)$

$\Rightarrow \bot \leq f(\bot) \leq f^2(\bot)) \leq \cdots \leq f^n(\bot) \leq \cdots$

exists n with $f^n(\bot) = f^{n+1}(\bot)$

Least fixed point

$P = f(P)$ fixed point, $f^n(\bot) = f^{n+1}(\bot)$

Monotonicity

$\bot \leq f(\bot)$

$\Rightarrow f(\bot) \leq f(f(\bot))$

$\Rightarrow f(f(\bot)) \leq f(f(f(\bot)))$

$\Rightarrow \ldots$

$\Rightarrow f^n(\bot)) \leq f^{n+1}(\bot)$

$\Rightarrow \bot \leq f(\bot) \leq f^2(\bot)) \leq \cdots \leq f^n(\bot) \leq \cdots$

exists n with $f^n(\bot) = f^{n+1}(\bot)$

Least fixed point

$P = f(P)$ fixed point, $f^n(\bot) = f^{n+1}(\bot)$

$\bot \leq P$

Monotonicity

$\bot \leq f(\bot)$
$\Rightarrow f(\bot) \leq f(f(\bot))$
$\Rightarrow f(f(\bot)) \leq f(f(f(\bot)))$
$\Rightarrow \dots$
$\Rightarrow f^n(\bot)) \leq f^{n+1}(\bot)$

$\Rightarrow \bot \leq f(\bot) \leq f^2(\bot)) \leq \cdots \leq f^n(\bot) \leq \cdots$

exists n with $f^n(\bot) = f^{n+1}(\bot)$

Least fixed point

$P = f(P)$ fixed point, $f^n(\bot) = f^{n+1}(\bot)$

$\bot \leq P$
$\Rightarrow f^n(\bot) \leq f^n(P) = P$

Monadic Second-Order Logic

Syntax

- element variables: $x, y, z, \ldots$
- set variables: $X, Y, Z, \ldots$
- atomic formulae: $R(\bar{x}), x = y, X(x)$
- boolean operations: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$
- quantifiers: $\exists x, \forall x, \exists X, \forall X$

Example

- "The set X is empty."

Monadic Second-Order Logic

Syntax

- element variables: $x, y, z, \dots$
- set variables: $X, Y, Z, \dots$
- atomic formulae: $R(\bar{x}), x = y, X(x)$
- boolean operations: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$
- quantifiers: $\exists x, \forall x, \exists X, \forall X$

Example

- "The set X is empty."

  $\neg \exists x X(x)$
- "$X \subseteq Y$"

Monadic Second-Order Logic

Syntax

- element variables: $x, y, z, \dots$
- set variables: $X, Y, Z, \dots$
- atomic formulae: $R(\bar{x}), x = y, X(x)$
- boolean operations: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$
- quantifiers: $\exists x, \forall x, \exists X, \forall X$

Example

- "The set X is empty."

  $\neg \exists x X(x)$
- "$X \subseteq Y$"

  $\forall z[X(z) \rightarrow Y(z)]$
- "There exists a path from x to y."

Monadic Second-Order Logic

Syntax

- element variables: $x, y, z, \dots$
- set variables: $X, Y, Z, \dots$
- atomic formulae: $R(\bar{x}), x = y, X(x)$
- boolean operations: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$
- quantifiers: $\exists x, \forall x, \exists X, \forall X$

Example

- "The set X is empty."

  $\neg \exists x X(x)$

- "$X \subseteq Y$"

  $\forall z [X(z) \rightarrow Y(z)]$

- "There exists a path from x to y."

  $\forall Z \big[Z(x) \wedge \forall u \forall v [Z(u) \wedge E(u, v) \rightarrow Z(v)] \rightarrow Z(y) \big]$

Equivalence

Quantifier rank

nesting depth of quantifiers in φ

Equivalence

Quantifier rank

nesting depth of quantifiers in φ

Equivalence

$\mathfrak{A}, \bar{P}, \bar{a} \equiv_m \mathfrak{B}, \bar{Q}, \bar{b}$:iff $\mathfrak{A} \vDash \varphi(\bar{P}, \bar{a}) \Leftrightarrow \mathfrak{B} \vDash \varphi(\bar{Q}, \bar{b})$
for all $\varphi(\bar{X}, \bar{x})$ of quantifier rank $\leq m$

Equivalence

Quantifier rank

nesting depth of quantifiers in φ

Equivalence

$$\mathfrak{A}, \bar{P}, \bar{a} \equiv_m \mathfrak{B}, \bar{Q}, \bar{b} \quad \text{:iff} \quad \mathfrak{A} \vDash \varphi(\bar{P}, \bar{a}) \Leftrightarrow \mathfrak{B} \vDash \varphi(\bar{Q}, \bar{b})$$

for all $\varphi(\bar{X}, \bar{x})$ of quantifier rank $\leq m$

word structures: $\mathfrak{W} = \langle [n], \leq, (P_a)_{a \in \Sigma} \rangle$

Equivalence

Quantifier rank

nesting depth of quantifiers in φ

Equivalence

$$\mathfrak{A}, \bar{P}, \bar{a} \equiv_m \mathfrak{B}, \bar{Q}, \bar{b} \quad :\text{iff} \quad \mathfrak{A} \vDash \varphi(\bar{P}, \bar{a}) \Leftrightarrow \mathfrak{B} \vDash \varphi(\bar{Q}, \bar{b})$$

for all $\varphi(\bar{X}, \bar{x})$ of quantifier rank $\leq m$

word structures: $\mathfrak{W} = \langle [n], \leq, (P_a)_{a \in \Sigma} \rangle$

Lemma

$u \equiv_m u'$ and $v \equiv_m v'$ implies $uv \equiv_m u'v'$

Proof induction on m

Automata

Given φ of quantifier rank m, construct $\mathcal{A}_\varphi = \langle Q, \Sigma, \delta, q_0, F \rangle$

Automata

Given φ of quantifier rank m, construct $\mathcal{A}_\varphi = \langle Q, \Sigma, \delta, q_0, F \rangle$

- $Q := \Sigma^* / \equiv_m$

Automata

Given φ of quantifier rank m, construct $\mathcal{A}_\varphi = \langle Q, \Sigma, \delta, q_0, F \rangle$

- $Q := \Sigma^* / \equiv_m$
- $q_0 := [\varepsilon]_m$

Automata

Given φ of quantifier rank m, construct $\mathcal{A}_\varphi = \langle Q, \Sigma, \delta, q_0, F \rangle$

- $Q := \Sigma^* / \equiv_m$
- $q_0 := [\varepsilon]_m$
- $\delta([w]_m, c) := [wc]_m$

Automata

Given φ of quantifier rank m, construct $\mathcal{A}_\varphi = \langle Q, \Sigma, \delta, q_0, F \rangle$

- $Q := \Sigma^* / \equiv_m$
- $q_0 := [\varepsilon]_m$
- $\delta([w]_m, c) := [wc]_m$
- $F := \{ [w]_m \mid w \vDash \varphi \}$

Automata

Given φ of quantifier rank m, construct $\mathcal{A}_\varphi = \langle Q, \Sigma, \delta, q_0, F \rangle$

- $Q := \Sigma^* / \equiv_m$
- $q_0 := [\varepsilon]_m$
- $\delta([w]_m, c) := [wc]_m$
- $F := \{ [w]_m \mid w \vDash \varphi \}$

Theorem

$\mathcal{A}_\varphi$ accepts a word $w \in \Sigma^*$ if, and only if, $w \vDash \varphi$.

Automata

Given φ of quantifier rank m, construct $\mathcal{A}_\varphi = \langle Q, \Sigma, \delta, q_0, F \rangle$

- $Q := \Sigma^* / {\equiv_m}$
- $q_0 := [\varepsilon]_m$
- $\delta([w]_m, c) := [wc]_m$
- $F := \{ [w]_m \mid w \vDash \varphi \}$

Theorem

$\mathcal{A}_\varphi$ accepts a word $w \in \Sigma^*$ if, and only if, $w \vDash \varphi$.

Corollary

φ is satisfiable if, and only if, $\mathcal{A}_\varphi$ accepts some word.

Description Logics

Description Logic

General Idea

Extend **modal logic** with operations that are **not bisimulation-invariant.**

Applications

Knowledge representation, deductive databases, system modelling, semantic web

Ingredients

- **individuals:** elements (Anna, John, Paul, Marry,…)
- **concepts:** unary predicates (person, male, female,…)
- **roles:** binary relations (has_child, is_married_to,…)
- **TBox:** terminology definitions
- **ABox:** assertions about the world

Example

TBox

$$\begin{aligned} \text{man} &:= \text{person} \wedge \text{male} \\ \text{woman} &:= \text{person} \wedge \text{female} \\ \text{father} &:= \text{man} \wedge \exists \text{has\_child.person} \\ \text{mother} &:= \text{woman} \wedge \exists \text{has\_child.person} \end{aligned}$$

ABox

man(John)

man(Paul)

woman(Anna)

woman(Marry)

has_child(Anna, Paul)

is_married_to(Anna, John)

Syntax

Concepts

$$\varphi ::= P \mid \top \mid \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \forall R\varphi \mid \exists R\varphi \mid (\geq nR) \mid (\leq nR)$$

Terminology axioms

$$\varphi \sqsubseteq \psi \qquad \varphi \equiv \psi$$

TBox Axioms of the form $P \equiv \varphi$.

Assertions

$$\varphi(a) \qquad R(a, b)$$

Extensions

- operations on roles: $R \cap S$, $R \cup S$, $R \circ S$, $\neg R$, R^+, R^*, R^-
- extended number restrictions: $(\geq nR)\varphi$, $(\leq nR)\varphi$

Algorithmic Problems

- **Satisfiability:** Is φ satisfiable?
- **Subsumption:** $\varphi \models \psi$?
- **Equivalence:** $\varphi \equiv \psi$?
- **Disjointness:** $\varphi \wedge \psi$ unsatisfiable?

All problems can be solved with standard methods like **tableaux** or **tree automata.**

Semantic Web: OWL (functional syntax)

```
Ontology(
  Class(pp:man    complete
        intersectionOf(pp:person pp:male))
  Class(pp:woman  complete
        intersectionOf(pp:person pp:female))
  Class(pp:father complete
        intersectionOf(pp:man
          restriction(pp:has_child pp:person)))
  Class(pp:mother complete
        intersectionOf(pp:woman
          restriction(pp:has_child pp:person)))
  Individual(pp:John  type(pp:man))
  Individual(pp:Paul  type(pp:man))
  Individual(pp:Anna  type(pp:woman)
             value(pp:has_child     pp:Paul)
             value(pp:is_married_to pp:John))
  Individual(pp:Marry type(pp:woman))
)
```