The μ-calculus and its proof theory

Bahareh Afshari

University of Gothenburg, Sweden

École de Printemps d'Informatique Théorique 2025, May 2025,
Aussois (France)

Outline

Outline

Section 1

Introduction

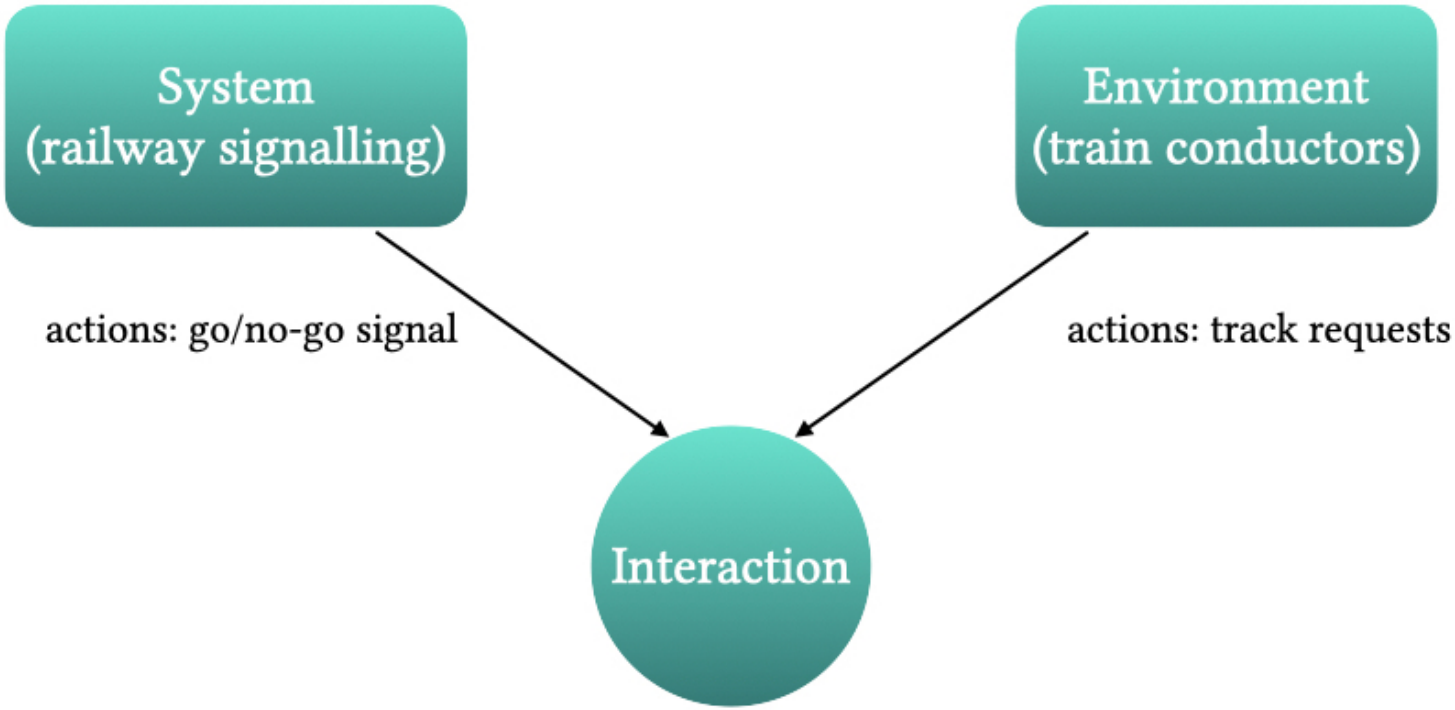
Modal μ-calculus

- *Blue Book (Chapter 8)* Temporal Logics in Computer Science: Finite-State Systems. Demri, Goranko & Lange, CUP (2016)
- *The μ-calculus and Model Checking.* Bradfield & Walukiewicz. Springer International Publishing, Cham (2018)
- *Green Book (Part V)* Automata, Logics, and Infinite Games. Grädel, Thomas & Wilke (editors). LNCS (2002)
- *Yellow Book (Chapter 6)* Rudiments of μ-calculus. Arnold & Niwinski. Vol. 146. Elsevier (2001)

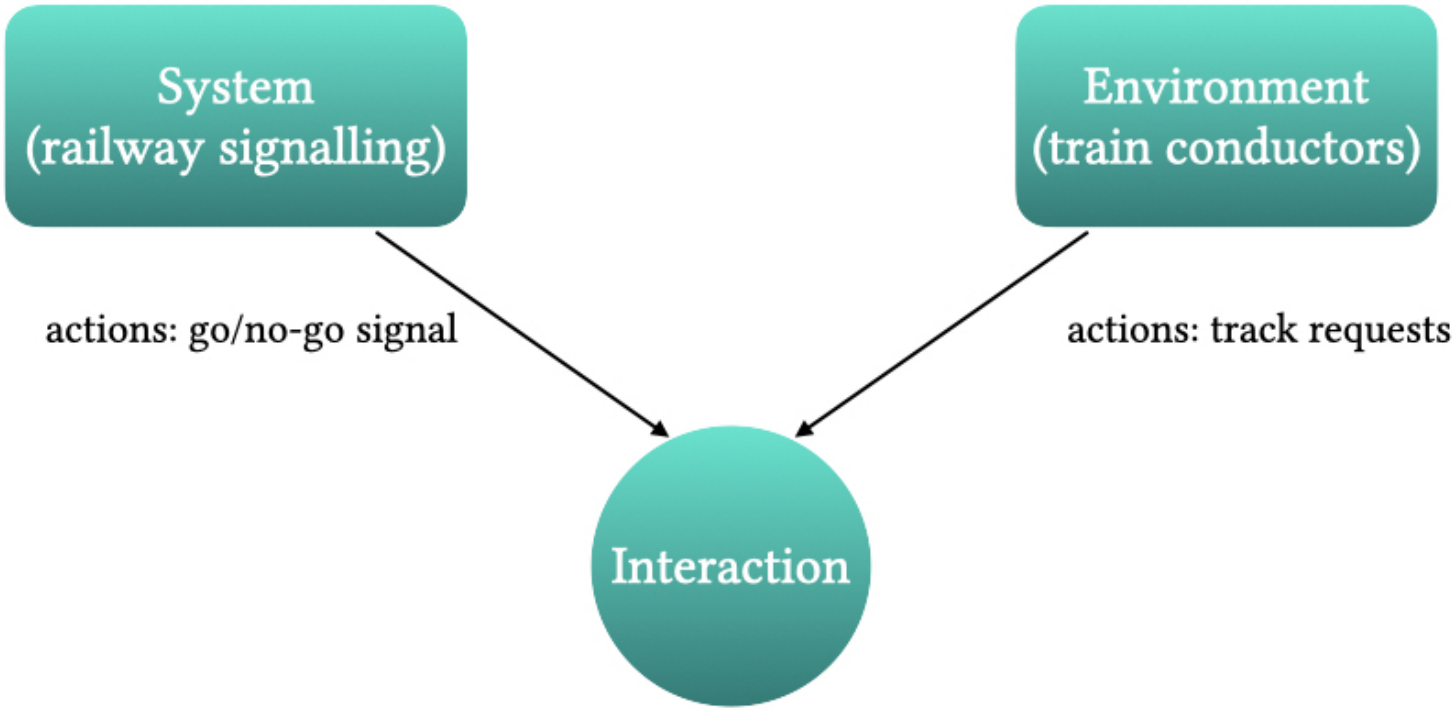
Proof Theory

- **Games for the μ-calculus.** Niwinski & Walukiewicz. TCS 163(1&2):99-116 (1996)
- **On the Proof Theory of the Modal mu-Calculus**. Studer. Studia Logica 89, 343-363 (2008)
- **Finitary Proof Systems for Kozen's μ.** Afshari and Leigh Oberwolfach Preprints. DOI 10.14760/OWP-2016-26 (2016)
- **Demystifying μ.** Afshari et al. CoRR abs/2401.01096 (2024)

Railway signalling



Railway signalling



Questions Is the system operating correctly? Are there any faults?

Reactive systems

Main characteristics:

- interact with the environment
- safety-critical
- run indefinitely

Other examples: *air traffic control systems* or *automated stock trading*

Reactive systems

Main characteristics:

- interact with the environment
- safety-critical
- run indefinitely

Other examples: *air traffic control systems* or *automated stock trading*

Want to verify

– **Liveness Good things eventually happen**

– **Safety Bad things never happen**

Railway signalling

- Let p_i denote "Train #i is granted permission to enter the signal block."

Railway signalling

- Let p_i denote "Train #i is granted permission to enter the signal block."
- Requirements:
  1. Eventually p_1 (liveness)
  2. Eventually p_2 (liveness)
  3. Never(p_1 and p_2) (safety)

Railway signalling

- Let p_i denote "Train #i is granted permission to enter the signal block."
- Requirements:
  1. Eventually p_1 (liveness)
  2. Eventually p_2 (liveness)
  3. Never(p_1 and p_2) (safety)

Liveness "Something good will eventually happen."

Safety "Something bad will never happen."

Railway signalling

- Let p_i denote "Train #i is granted permission to enter the signal block."
- Requirements:
  1. Eventually p_1 (liveness)
  2. Eventually p_2 (liveness)
  3. Never(p_1 and p_2) (safety)

Liveness "Something good will eventually happen."

Safety "Something bad will never happen."

Question How can we express these properties in a logical language?

- $1 \equiv p_1$ or **Next**(1)

Railway signalling

- Let p_i denote "Train #i is granted permission to enter the signal block."
- Requirements:
  1. Eventually p_1 (liveness)
  2. Eventually p_2 (liveness)
  3. Never(p_1 and p_2) (safety)

Liveness "Something good will eventually happen."

Safety "Something bad will never happen."

Question How can we express these properties in a logical language?

- $\mathbf{1} \equiv p_1$ or **Next**(**1**)
- **3** $\equiv$ not(p_1 and p_2) and **Next**(**3**)

Railway signalling

- Let p_i denote "Train #i is granted permission to enter the signal block."
- Requirements:
  1. Eventually p_1 (liveness)
  2. Eventually p_2 (liveness)
  3. Never(p_1 and p_2) (safety)

Liveness "Something good will eventually happen."

Safety "Something bad will never happen."

Question How can we express these properties in a logical language?

- $1 \equiv p_1$ or **Next**(1)
- $3 \equiv \text{not}(p_1 \text{ and } p_2)$ and **Next**(3)
- $\mathsf{x} \equiv \varphi(\mathsf{x})$ where φ is a statement using finite vocabulary such as **Next** and logical connectives.

Questions of interest

1. Are there logics suitable to express such temporal properties?
2. What is the cost of checking liveness, safety and other fairness constraints?
3. Can practical algorithms be developed for real-world examples?

Outline

Section 2

Modal Logic

Propositional logic:

- propositional variables p, q, ...
  connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$

Propositional logic:

- propositional variables p, q, ...
  connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$
- Syntax $\quad p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi$
- Semantics: formulas are either *true* or *false*

Propositional logic:

- propositional variables p, q, ...
  connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$
- Syntax $\quad p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi$
- Semantics: formulas are either *true* or *false*

One can extend propositional logic by

- adding two unary predicates:

$\Box$: box; necessarily

$\Diamond$: diamond; possibly

Propositional logic:

- propositional variables p, q, ...
  connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$
- Syntax $\quad p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi$
- Semantics: formulas are either *true* or *false*

One can extend propositional logic by

- adding two unary predicates:

  $\Box$: box; necessarily

  $\Diamond$: diamond; possibly

- If φ is a formula so are $\Box\varphi$ and $\Diamond\varphi$.

Propositional logic:

- propositional variables p, q, ...
  connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$
- Syntax $\quad p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi$
- Semantics: formulas are either *true* or *false*

One can extend propositional logic by

- adding two unary predicates:

  $\Box$: box; necessarily

  $\Diamond$: diamond; possibly

- If φ is a formula so are $\Box\varphi$ and $\Diamond\varphi$.
- Examples: $\Box(p \vee \Diamond q)$, $\Box\Diamond\neg p$

A model of modal logic, called a **transition system** (or Kripke structure), is a triple $T = (S, \to, \lambda)$ with a

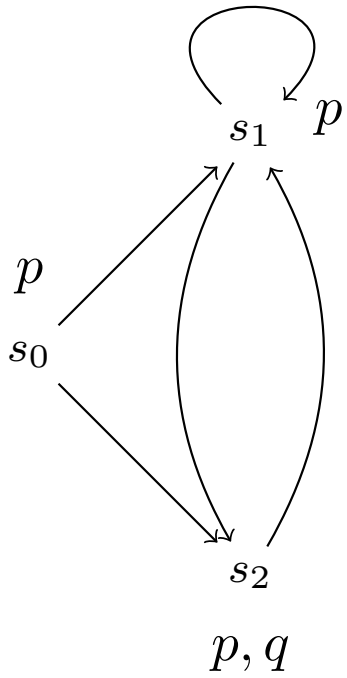
- set of states S;
- transition relation $\to \subseteq S \times S$, writing also $s \to t$ for $(s, t) \in \to$
- function $\lambda : \textsc{Prop} \to \mathcal{P}(S)$ interpreting atomic propositions

A model of modal logic, called a **transition system** (or Kripke structure), is a triple $T = (S, \to, \lambda)$ with a

- set of states S;
- transition relation $\to \subseteq S \times S$, writing also $s \to t$ for $(s, t) \in \to$
- function $\lambda : \textsc{Prop} \to \mathcal{P}(S)$ interpreting atomic propositions

Note that a transition system is simply a **directed graph** whose nodes are labelled by **finite sets of propositions**.

Example



Transition system with states $\{s_0, s_1, s_2\}$

We define what it means for a transition system T to model/satisfy a formula φ at a world s, written $T, s \models \varphi$, by structural induction:

- $T, s \models p$ iff $p \in \lambda(s)$
- $T, s \models \neg\varphi$ iff $T, s \not\models \varphi$
- $T, s \models \varphi \wedge \psi$ iff $T, s \models \varphi$ and $T, s \models \psi$
- $T, s \models \varphi \vee \psi$ iff $T, s \models \varphi$ or $T, s \models \psi$
- $T, s \models \Box\varphi$ iff for every $s \to t$ we have $T, t \models \varphi$
- $T, s \models \Diamond\varphi$ iff for some $s \to t$ we have $T, t \models \varphi$

We define what it means for a transition system T to model/satisfy a formula φ at a world s, written $T, s \models \varphi$, by structural induction:

- $T, s \models p$ iff $p \in \lambda(s)$
- $T, s \models \neg\varphi$ iff $T, s \not\models \varphi$
- $T, s \models \varphi \wedge \psi$ iff $T, s \models \varphi$ and $T, s \models \psi$
- $T, s \models \varphi \vee \psi$ iff $T, s \models \varphi$ or $T, s \models \psi$
- $T, s \models \Box\varphi$ iff for every $s \to t$ we have $T, t \models \varphi$
- $T, s \models \Diamond\varphi$ iff for some $s \to t$ we have $T, t \models \varphi$

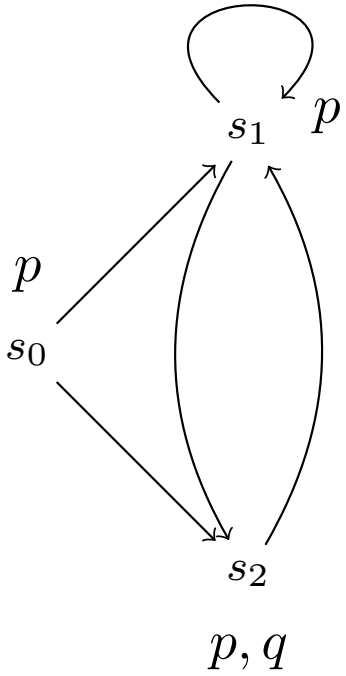
We define what it means for a transition system T to model/satisfy a formula φ at a world s, written $T, s \models \varphi$, by structural induction:

- $T, s \models p$ iff $p \in \lambda(s)$
- $T, s \models \neg\varphi$ iff $T, s \not\models \varphi$
- $T, s \models \varphi \wedge \psi$ iff $T, s \models \varphi$ and $T, s \models \psi$
- $T, s \models \varphi \vee \psi$ iff $T, s \models \varphi$ or $T, s \models \psi$
- $T, s \models \Box\varphi$ iff for every $s \to t$ we have $T, t \models \varphi$
- $T, s \models \Diamond\varphi$ iff for some $s \to t$ we have $T, t \models \varphi$

We write $\|\varphi\|^T$ to denote the set of all states s of transition system T that satisfy the formula φ. In other words,

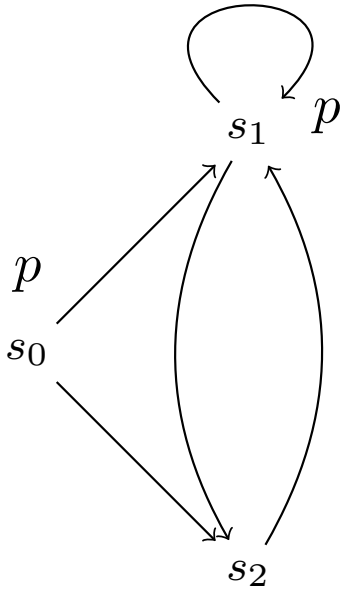
$$s \in \|\varphi\|^T \text{ if and only if } T, s \models \varphi.$$

Examples



Transition system T with states $\{s_0, s_1, s_2\}$

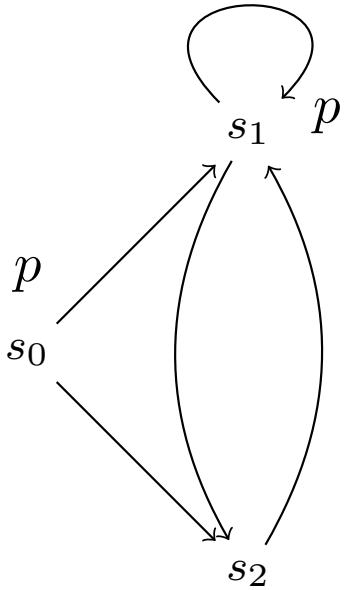
Examples



- $\|p\|^T =$

Transition system T with states $\{s_0, s_1, s_2\}$

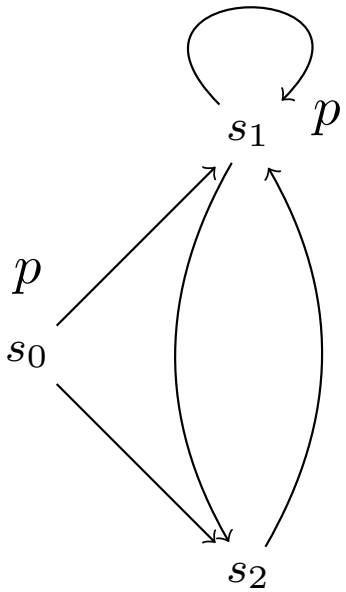
Examples



- $\|p\|^T = \{s_0, s_1, s_2\}$

Transition system T with states $\{s_0, s_1, s_2\}$

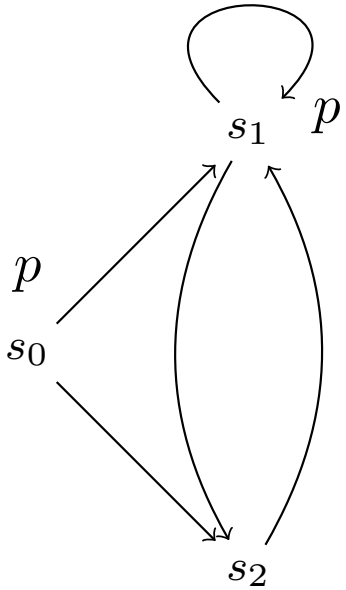
Examples



- $\|p\|^T = \{s_0, s_1, s_2\}$
- $s_0 \models^T \Box p$

Transition system T with states $\{s_0, s_1, s_2\}$

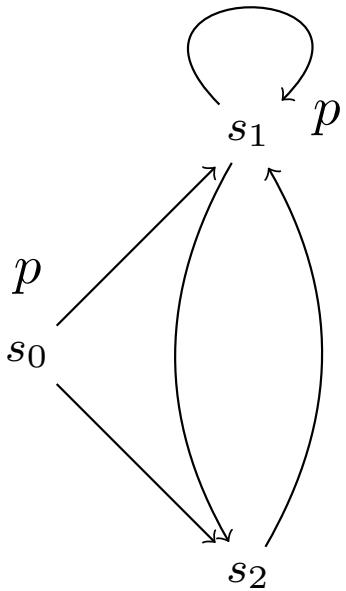
Examples



- $\|p\|^T = \{s_0, s_1, s_2\}$
- $s_0 \models^T \Box p$
- $\|q\|^T =$

Transition system T with states $\{s_0, s_1, s_2\}$

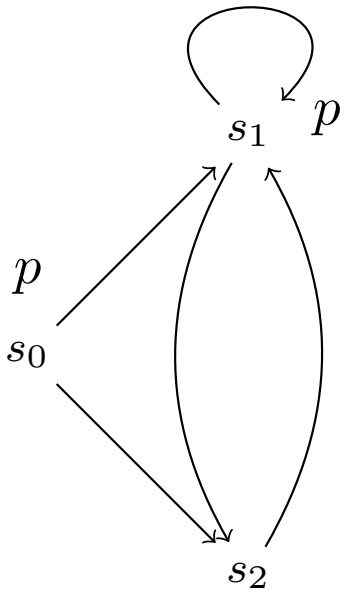
Examples



- $\|p\|^T = \{s_0, s_1, s_2\}$
- $s_0 \models^T \Box p$
- $\|q\|^T = \{s_2\}$

Transition system T with states $\{s_0, s_1, s_2\}$

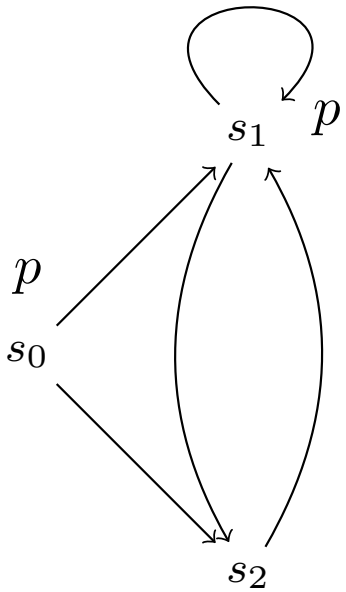
Examples



- $\|p\|^T = \{s_0, s_1, s_2\}$
- $s_0 \models^T \Box p$
- $\|q\|^T = \{s_2\}$
- $\|\Diamond q\|^T =$

Transition system T with states $\{s_0, s_1, s_2\}$

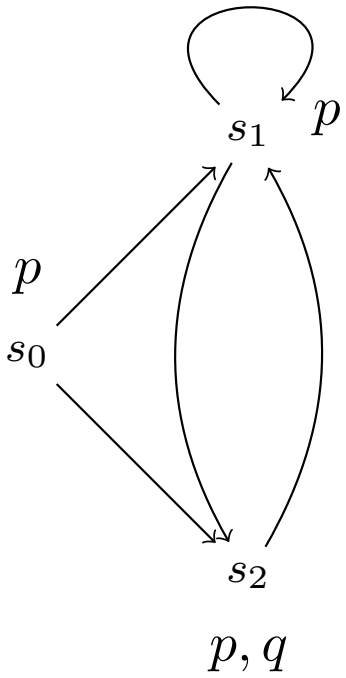
Examples



- $\|p\|^T = \{s_0, s_1, s_2\}$
- $s_0 \models^T \Box p$
- $\|q\|^T = \{s_2\}$
- $\|\Diamond q\|^T = \{s_0, s_1\}$

Transition system T with states $\{s_0, s_1, s_2\}$

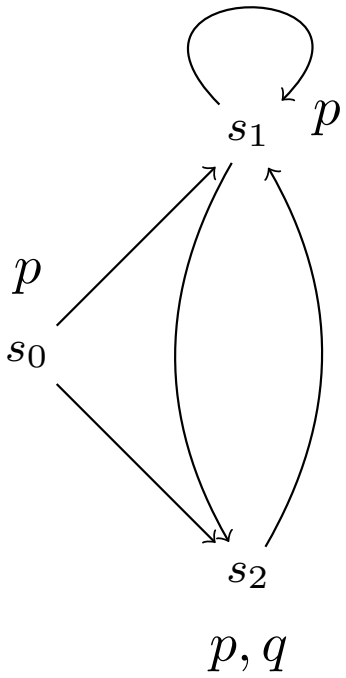
Examples



- $\|p\|^T = \{s_0, s_1, s_2\}$
- $s_0 \models^T \Box p$
- $\|q\|^T = \{s_2\}$
- $\|\Diamond q\|^T = \{s_0, s_1\}$
- $\|\Diamond\Diamond q\|^T =$

Transition system T with states $\{s_0, s_1, s_2\}$

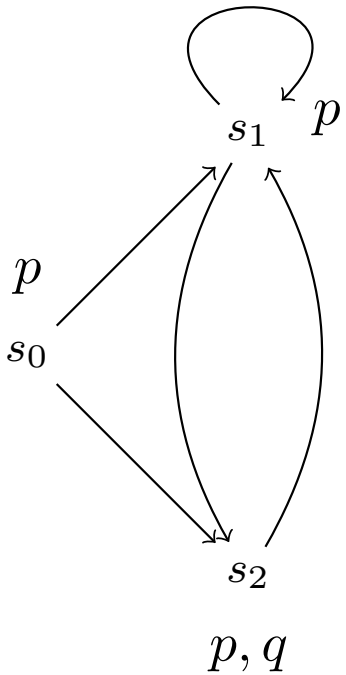
Examples



- $\|p\|^T = \{s_0, s_1, s_2\}$
- $s_0 \models^T \Box p$
- $\|q\|^T = \{s_2\}$
- $\|\Diamond q\|^T = \{s_0, s_1\}$
- $\|\Diamond\Diamond q\|^T = \{s_0, s_1, s_2\}$

Transition system T with states $\{s_0, s_1, s_2\}$

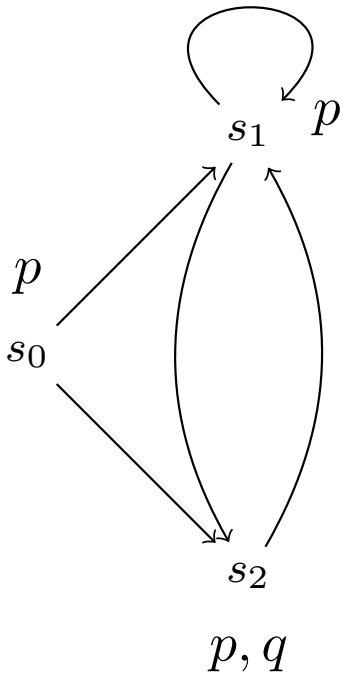
Examples



- $\|p\|^T = \{s_0, s_1, s_2\}$
- $s_0 \models^T \Box p$
- $\|q\|^T = \{s_2\}$
- $\|\Diamond q\|^T = \{s_0, s_1\}$
- $\|\Diamond\Diamond q\|^T = \{s_0, s_1, s_2\}$
- $\|\Box\Diamond\neg q\|^T = ?$

Transition system T with states $\{s_0, s_1, s_2\}$

Examples



- $\|p\|^T = \{s_0, s_1, s_2\}$
- $s_0 \models^T \Box p$
- $\|q\|^T = \{s_2\}$
- $\|\Diamond q\|^T = \{s_0, s_1\}$
- $\|\Diamond\Diamond q\|^T = \{s_0, s_1, s_2\}$
- $\|\Box\Diamond\neg q\|^T = ?$

Transition system T with states $\{s_0, s_1, s_2\}$

Satisfiability via tableaux

We will look at a **syntactic** way of constructing models for modal logics.

Satisfiability via tableaux

We will look at a **syntactic** way of constructing models for modal logics.

Semantic methods:

- selection
- canonical model construction
- filteration

Syntactic methods:

- tableaux
- automata
- games

Satisfiability via tableaux

We will look at a **syntactic** way of constructing models for modal logics.

Semantic methods:

- selection
- canonical model construction
- filteration

Syntactic methods:

- tableaux
- automata
- games

Remark Tableaux are often designed for satisfiability but one can also talk about tableaux for validity. As the two notions are dual if you have a tableau system for one you can define an analogous one for the other.

Yet another model construction technique

Yet another model construction technique

Questions

1. Can we get finite models? Often, but not always.

Yet another model construction technique

Questions

1. Can we get finite models? Often, but not always.
2. Is it a robust technique like canonical model construction? No!

Yet another model construction technique

Questions

1. Can we get finite models? Often, but not always.
2. Is it a robust technique like canonical model construction? No!
3. Can one use tableaux for decidability? Yes, sometimes.

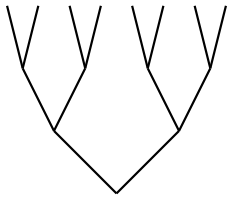
Yet another model construction technique

Questions

1. Can we get finite models? Often, but not always.
2. Is it a robust technique like canonical model construction? No!
3. Can one use tableaux for decidability? Yes, sometimes.
4. When should we use tableaux? Seeking computational properties.

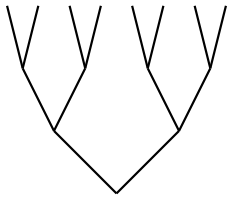
How does a tableau look like?

Our tableaux are going to look like (upwards growing) trees:



How does a tableau look like?

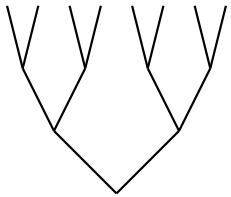
Our tableaux are going to look like (upwards growing) trees:



Given a formula φ we want to check if it satisfiable. The idea is to systematically build a model from **using the logical structure** of φ:

How does a tableau look like?

Our tableaux are going to look like (upwards growing) trees:

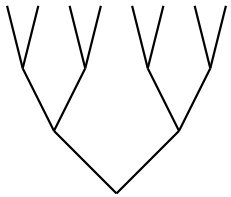


Given a formula φ we want to check if it satisfiable. The idea is to systematically build a model from **using the logical structure** of φ:

- Start the tree by putting φ at the **root**;

How does a tableau look like?

Our tableaux are going to look like (upwards growing) trees:

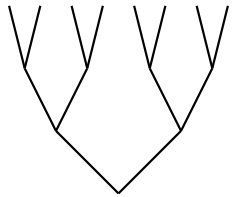


Given a formula φ we want to check if it satisfiable. The idea is to systematically build a model from **using the logical structure** of φ:

- Start the tree by putting φ at the **root**;
- From the root make new **branches** (unary, binary or more);

How does a tableau look like?

Our tableaux are going to look like (upwards growing) trees:

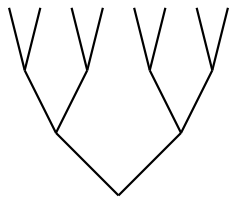


Given a formula φ we want to check if it satisfiable. The idea is to systematically build a model from **using the logical structure** of φ:

- Start the tree by putting φ at the **root**;
- From the root make new **branches** (unary, binary or more);
- The idea of these new nodes is to reduce checking satisfiability of φ to checking **satisfiability of its constituents**;

How does a tableau look like?

Our tableaux are going to look like (upwards growing) trees:

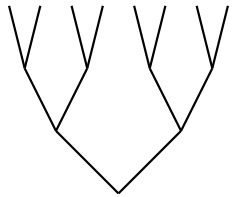


Given a formula φ we want to check if it satisfiable. The idea is to systematically build a model from **using the logical structure** of φ:

- Start the tree by putting φ at the **root**;
- From the root make new **branches** (unary, binary or more);
- The idea of these new nodes is to reduce checking satisfiability of φ to checking **satisfiability of its constituents**;
- To check satisfiability of $\varphi_1 \wedge \varphi_2$ we check both φ_1 and φ_2;

How does a tableau look like?

Our tableaux are going to look like (upwards growing) trees:



Given a formula φ we want to check if it satisfiable. The idea is to systematically build a model from **using the logical structure** of φ:

- Start the tree by putting φ at the **root**;
- From the root make new **branches** (unary, binary or more);
- The idea of these new nodes is to reduce checking satisfiability of φ to checking **satisfiability of its constituents**;
- To check satisfiability of $\varphi_1 \wedge \varphi_2$ we check both φ_1 and φ_2;
- Continue breaking down φ until only **literals** remain.

To make our tableaux construction as simple as possible we are going to work with modal formulas in negation normal form.

To make our tableaux construction as simple as possible we are going to work with modal formulas in negation normal form.

Definition

A formula of modal logic is in Negation Normal Form (NNF) iff it can be constructed by $\Box$, $\Diamond$, $\wedge$, $\vee$ from propositions and negated propositions).

To make our tableaux construction as simple as possible we are going to work with modal formulas in negation normal form.

Definition

A formula of modal logic is in Negation Normal Form (NNF) iff it can be constructed by $\Box$, $\Diamond$, $\wedge$, $\vee$ from propositions and negated propositions).

Examples:

- $\neg\Diamond(\neg p \wedge q)$ is not in NNF but $\Box(p \vee \neg q)$ is.
- What about $\Box\neg(p \wedge q)$?

To make our tableaux construction as simple as possible we are going to work with modal formulas in negation normal form.

Definition

A formula of modal logic is in Negation Normal Form (NNF) iff it can be constructed by $\Box$, $\Diamond$, $\wedge$, $\vee$ from propositions and negated propositions).

Examples:

- $\neg\Diamond(\neg p \wedge q)$ is not in NNF but $\Box(p \vee \neg q)$ is.
- What about $\Box\neg(p \wedge q)$?

NNF is not a restriction on expressibility:

Lemma (Exercise)

Every formula of modal logic is equivalent to one in NNF. The equivalence is provable in **K**.

Motivating example

Consider the formula $\varphi = (\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)$. We aim to find a model of φ, namely some $M = (W, \to, V)$ and $w_0 \in W$ such that $M, w_0 \models \varphi$.

Motivating example

Consider the formula $\varphi = (\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)$. We aim to find a model of φ, namely some $M = (W, \to, V)$ and $w_0 \in W$ such that $M, w_0 \models \varphi$.

$$w_0 \models (\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)$$
$$w_0 \models (\Box\Box p \wedge \Diamond q)$$
$$w_0 \models \{\Box\Box p, \Diamond q\}$$
$$\exists w_1 : w_0 \to w_1 \qquad w_1 \models \{\Box p, q\}$$

Motivating example

Consider the formula $\varphi = (\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)$. We aim to find a model of φ, namely some $M = (W, \to, V)$ and $w_0 \in W$ such that $M, w_0 \models \varphi$.

$$w_0 \models (\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)$$
$$w_0 \models (\Box\Box p \wedge \Diamond q)$$
$$w_0 \models \{\Box\Box p, \Diamond q\}$$
$$\exists w_1 : w_0 \to w_1 \qquad w_1 \models \{\Box p, q\}$$

This "model search" naturally gives rise to the structure M given by:

$$\begin{cases} W = \{w_0, w_1\} \\ w_0 \to w_1 \\ V(p) = \varnothing, V(q) = \{w_1\} \end{cases}$$

Motivating example

Consider the formula $\varphi = (\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)$. We aim to find a model of φ, namely some $M = (W, \to, V)$ and $w_0 \in W$ such that $M, w_0 \models \varphi$.

$$w_0 \models (\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)$$
$$w_0 \models (\Box\Box p \wedge \Diamond q)$$
$$w_0 \models \{\Box\Box p, \Diamond q\}$$
$$\exists w_1 : w_0 \to w_1 \qquad w_1 \models \{\Box p, q\}$$

This "model search" naturally gives rise to the structure M given by:

$$\begin{cases} W = \{w_0, w_1\} \\ w_0 \to w_1 \\ V(p) = \varnothing, V(q) = \{w_1\} \end{cases}$$

This search for satisfiability (or soundness) can be formalised using tableaux, a **tree** where each node is labelled by a **subset of** $Sub(\varphi)$ (the set of all subformulas of φ).

Trees

A **tree** (over Σ) is a triple $t = \langle S, \to, \lambda \rangle$ with a distinguished node $\rho \in S$ which satisfies the following conditions.

Trees

A **tree** (over Σ) is a triple $t = \langle S, \to, \lambda \rangle$ with a distinguished node $\rho \in S$ which satisfies the following conditions.

- $(S, \to)$ is a connected directed graph.

Trees

A **tree** (over Σ) is a triple $t = \langle S, \to, \lambda \rangle$ with a distinguished node $\rho \in S$ which satisfies the following conditions.

- $(S, \to)$ is a connected directed graph.
- There are no transitions into ρ.

Trees

A **tree** (over Σ) is a triple $t = \langle S, \to, \lambda \rangle$ with a distinguished node $\rho \in S$ which satisfies the following conditions.

- $(S, \to)$ is a connected directed graph.
- There are no transitions into ρ.
- For every $s \in S \setminus \{\rho\}$ there is exactly one $s_0 \in S$ such that $s_0 \to s$.

Trees

A **tree** (over Σ) is a triple $t = \langle S, \to, \lambda \rangle$ with a distinguished node $\rho \in S$ which satisfies the following conditions.

- $(S, \to)$ is a connected directed graph.
- There are no transitions into ρ.
- For every $s \in S \setminus \{\rho\}$ there is exactly one $s_0 \in S$ such that $s_0 \to s$.
- $\lambda \colon S \to \mathcal{P}(\Sigma)$ is called the **labelling function** of t.

Trees

A **tree** (over Σ) is a triple $t = \langle S, \to, \lambda \rangle$ with a distinguished node $\rho \in S$ which satisfies the following conditions.

- $(S, \to)$ is a connected directed graph.
- There are no transitions into ρ.
- For every $s \in S \setminus \{\rho\}$ there is exactly one $s_0 \in S$ such that $s_0 \to s$.
- $\lambda \colon S \to \mathcal{P}(\Sigma)$ is called the **labelling function** of t.

The node ρ is referred to as the **root** of the tree and any node without outgoing edges is a **leaf**.

Remark. A tree is a special case of a Kripke structure.

Trees

A **tree** (over Σ) is a triple $t = \langle S, \to, \lambda \rangle$ with a distinguished node $\rho \in S$ which satisfies the following conditions.

- $(S, \to)$ is a connected directed graph.
- There are no transitions into ρ.
- For every $s \in S \setminus \{\rho\}$ there is exactly one $s_0 \in S$ such that $s_0 \to s$.
- $\lambda \colon S \to \mathcal{P}(\Sigma)$ is called the **labelling function** of t.

The node ρ is referred to as the **root** of the tree and any node without outgoing edges is a **leaf**.

Remark. A tree is a special case of a Kripke structure.

A **path** through tree t is a function $\mathbb{P} \colon \mathbb{N} \to S$ such that

- $\mathbb{P}(0) = \rho$,
- For every n, if $\mathbb{P}(n)$ is not a leaf, then $\mathbb{P}(n) \to \mathbb{P}(n+1)$.

Notation

- From now on, all formulas considered are in **NNF**.
- We use $\varphi, \psi, \delta, \gamma, \ldots$ (also with indices) for denoting formulas.
- Finite sets of formulas are denoted by $\Gamma, \Delta, \Theta, \Lambda, \ldots$

Notation

- From now on, all formulas considered are in **NNF**.
- We use $\varphi, \psi, \delta, \gamma, \ldots$ (also with indices) for denoting formulas.
- Finite sets of formulas are denoted by $\Gamma, \Delta, \Theta, \Lambda, \ldots$

- $\textsc{Prop}$ a (possibly infinite) set of propositions.
- A **literal** is an element of $\textsc{Lit} = \textsc{Prop} \cup \{\neg p \mid p \in \textsc{Prop}\}$.
- A set $X \subseteq \textsc{Lit}$ is **inconsistent** if $\{p, \neg p\} \subseteq X$ for some $p \in \textsc{Prop}$.

Notation

- From now on, all formulas considered are in **NNF**.
- We use $\varphi, \psi, \delta, \gamma, \ldots$ (also with indices) for denoting formulas.
- Finite sets of formulas are denoted by $\Gamma, \Delta, \Theta, \Lambda, \ldots$

- $\textsc{Prop}$ a (possibly infinite) set of propositions.
- A **literal** is an element of $\textsc{Lit} = \textsc{Prop} \cup \{\neg p \mid p \in \textsc{Prop}\}$.
- A set $X \subseteq \textsc{Lit}$ is **inconsistent** if $\{p, \neg p\} \subseteq X$ for some $p \in \textsc{Prop}$.

Notation

- From now on, all formulas considered are in **NNF**.
- We use $\varphi, \psi, \delta, \gamma, \ldots$ (also with indices) for denoting formulas.
- Finite sets of formulas are denoted by $\Gamma, \Delta, \Theta, \Lambda, \ldots$
- $\textsc{Prop}$ a (possibly infinite) set of propositions.
- A **literal** is an element of $\textsc{Lit} = \textsc{Prop} \cup \{\neg p \mid p \in \textsc{Prop}\}$.
- A set $X \subseteq \textsc{Lit}$ is **inconsistent** if $\{p, \neg p\} \subseteq X$ for some $p \in \textsc{Prop}$.
- $\Box\Gamma = \{\, \Box\varphi \mid \varphi \in \Gamma \,\}$ and $\Diamond\Gamma = \{\, \Diamond\varphi \mid \varphi \in \Gamma \,\}$.

Notation

- From now on, all formulas considered are in **NNF**.
- We use $\varphi, \psi, \delta, \gamma, \ldots$ (also with indices) for denoting formulas.
- Finite sets of formulas are denoted by $\Gamma, \Delta, \Theta, \Lambda, \ldots$

- $\textsc{Prop}$ a (possibly infinite) set of propositions.
- A **literal** is an element of $\textsc{Lit} = \textsc{Prop} \cup \{\neg p \mid p \in \textsc{Prop}\}$.
- A set $X \subseteq \textsc{Lit}$ is **inconsistent** if $\{p, \neg p\} \subseteq X$ for some $p \in \textsc{Prop}$.

- $\Box\Gamma = \{\, \Box\varphi \mid \varphi \in \Gamma \,\}$ and $\Diamond\Gamma = \{\, \Diamond\varphi \mid \varphi \in \Gamma \,\}$.
- Γ, φ means $\Gamma \cup \{\varphi\}$, and Γ, Δ denotes $\Gamma \cup \Delta$.

A **pre-tableau** for φ is a tree $t = \langle S, \to, \lambda \rangle$ over $Sub(\varphi)$ such that

A **pre-tableau** for φ is a tree $t = \langle S, \to, \lambda \rangle$ over $Sub(\varphi)$ such that

1. $\lambda(\rho) = \{\varphi\}$,

A **pre-tableau** for φ is a tree $t = \langle S, \to, \lambda \rangle$ over $Sub(\varphi)$ such that

1. $\lambda(\rho) = \{\varphi\}$,
2. every leaf $l \in S$ is labelled by a sequent of the form $\Box\Delta, \Diamond\Lambda, \Theta$ where $\Theta \subseteq \textsc{Lit}$ and either
   - $\Lambda = \varnothing$, or
   - Θ is inconsistent.

A **pre-tableau** for φ is a tree $t = \langle S, \to, \lambda \rangle$ over $Sub(\varphi)$ such that

1. $\lambda(\rho) = \{\varphi\}$,
2. every leaf $l \in S$ is labelled by a sequent of the form $\Box\Delta, \Diamond\Lambda, \Theta$ where $\Theta \subseteq \textsc{Lit}$ and either
   - $\Lambda = \varnothing$, or
   - Θ is inconsistent.
3. every non-leaf of t is related to its immediate successors by one of the rules below.

A **pre-tableau** for φ is a tree $t = \langle S, \to, \lambda \rangle$ over $Sub(\varphi)$ such that

1. $\lambda(\rho) = \{\varphi\}$,
2. every leaf $l \in S$ is labelled by a sequent of the form $\Box\Delta, \Diamond\Lambda, \Theta$ where $\Theta \subseteq \textsc{Lit}$ and either
   - $\Lambda = \varnothing$, or
   - Θ is inconsistent.
3. every non-leaf of t is related to its immediate successors by one of the rules below.

$$\wedge\ \frac{\Delta, \varphi_0, \varphi_1}{\Delta, \varphi_0 \wedge \varphi_1} \qquad \vee_0\ \frac{\Delta, \varphi_0}{\Delta, \varphi_0 \vee \varphi_1} \qquad \vee_1\ \frac{\Delta, \varphi_1}{\Delta, \varphi_0 \vee \varphi_1}$$

$$\text{mod}\ \frac{\Delta, \delta_1 \qquad \Delta, \delta_2 \qquad \cdots \qquad \Delta, \delta_n}{\Box\Delta, \Diamond\{\delta_1, \cdots, \delta_n\}, \Theta}\ \Theta \subseteq \textsc{Lit} \text{ is consistent}$$

Rules for **K** pre-tableaux

Remarks

- A pre-tableau is a finitely branching tree.
- Branching occurs only mod-rules.

Remarks

- A pre-tableau is a finitely branching tree.
- Branching occurs only mod-rules.
- The rules are read bottom-up (from root towards leaves).
- For each rule the distinguished formulas in the lower and upper sequents are called respectively the **principal** and **residual** formulas of the rule.

Remarks

- A pre-tableau is a finitely branching tree.
- Branching occurs only mod-rules.
- The rules are read bottom-up (from root towards leaves).
- For each rule the distinguished formulas in the lower and upper sequents are called respectively the **principal** and **residual** formulas of the rule.
- In the mod-rule all formulas are considered distinguished, including those in Θ.
- The mod-rule can be applied only if no other rule is applicable.

Remarks

- A pre-tableau is a finitely branching tree.
- Branching occurs only mod-rules.
- The rules are read bottom-up (from root towards leaves).
- For each rule the distinguished formulas in the lower and upper sequents are called respectively the **principal** and **residual** formulas of the rule.
- In the mod-rule all formulas are considered distinguished, including those in Θ.
- The mod-rule can be applied only if no other rule is applicable.
- For modal logic **K** tableaux are well-founded trees. In other words, every path is finite. **Why?**

Examples of pre-tableaux

The formula $(\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)$ has two pre-tableaux:

Examples of pre-tableaux

The formula $(\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)$ has two pre-tableaux:

$$\vee_0 \dfrac{\mathrm{mod}\dfrac{\wedge\dfrac{\Box p, q}{\Box\Box p, \Diamond q}}{\Box\Box p \wedge \Diamond q}}{(\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)} \qquad \vee_1 \dfrac{\wedge\dfrac{\neg p, \Box\neg q}{\neg p \wedge \Box\neg q}}{(\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)}$$

Examples of pre-tableaux

The formula $(\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)$ has two pre-tableaux:

$$\vee_0 \dfrac{\mathrm{mod}\dfrac{\wedge\dfrac{\Box p, q}{\Box\Box p, \Diamond q}}{\Box\Box p \wedge \Diamond q}}{(\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)} \qquad \vee_1 \dfrac{\wedge\dfrac{\neg p, \Box\neg q}{\neg p \wedge \Box\neg q}}{(\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)}$$

Question How many pre-tableaux does $(\neg p \vee \Diamond\Diamond q) \wedge (p \vee \Box\Box\neg q)$ have?

Examples of pre-tableaux

$$\wedge\ \frac{\dfrac{\dfrac{\dfrac{\neg p, p}{\neg p, p \vee \Box\Box\neg q}\vee_0}{\neg p \vee \Diamond\Diamond q, p \vee \Box\Box\neg q}\vee_0}{(\neg p \vee \Diamond\Diamond q) \wedge (p \vee \Box\Box\neg q)}}{}$$

$$\wedge\ \frac{\dfrac{\dfrac{\dfrac{\dfrac{q, \neg q}{\Diamond q, \Box\neg q}\mathrm{mod}}{\Diamond\Diamond q, \Box\Box\neg q}\mathrm{mod}}{\Diamond\Diamond q, p \vee \Box\Box\neg q}\vee_1}{\neg p \vee \Diamond\Diamond q, p \vee \Box\Box\neg q}\vee_1}{(\neg p \vee \Diamond\Diamond q) \wedge (p \vee \Box\Box\neg q)}$$

$$\wedge\ \frac{\dfrac{\dfrac{\dfrac{\neg p, \Box\Box\neg q}{\neg p, p \vee \Box\Box\neg q}\vee_1}{\neg p \vee \Diamond\Diamond q, p \vee \Box\Box\neg q}\vee_0}{(\neg p \vee \Diamond\Diamond q) \wedge (p \vee \Box\Box\neg q)}}{}$$

$$\wedge\ \frac{\dfrac{\dfrac{\dfrac{\dfrac{q}{\Diamond q}\mathrm{mod}}{\Diamond\Diamond q, p}\mathrm{mod}}{\Diamond\Diamond q, p \vee \Box\Box\neg q}\vee_0}{\neg p \vee \Diamond\Diamond q, p \vee \Box\Box\neg q}\vee_1}{(\neg p \vee \Diamond\Diamond q) \wedge (p \vee \Box\Box\neg q)}$$

Tableaux definition

A **tableau** is a pre-tableau where the sequent (finite set of formulas) at each leaf has the form $\Box\Gamma, \Theta$ where $\Theta \subseteq \textsc{Lit}$ is consistent.

Tableaux definition

A **tableau** is a pre-tableau where the sequent (finite set of formulas) at each leaf has the form $\Box\Gamma, \Theta$ where $\Theta \subseteq \textsc{Lit}$ is consistent.

Question Which of these are tableaux?

$$\wedge\,\dfrac{\vee_0\,\dfrac{\vee_0\,\dfrac{\neg p, p}{\neg p, p \vee \Box\Box\neg q}}{\neg p \vee \Diamond\Diamond q, p \vee \Box\Box\neg q}}{(\neg p \vee \Diamond\Diamond q) \wedge (p \vee \Box\Box\neg q)}$$

$$\wedge\,\dfrac{\vee_1\,\dfrac{\vee_0\,\dfrac{\neg p, \Box\Box\neg q}{\neg p, p \vee \Box\Box\neg q}}{\neg p \vee \Diamond\Diamond q, p \vee \Box\Box\neg q}}{(\neg p \vee \Diamond\Diamond q) \wedge (p \vee \Box\Box\neg q)}$$

$$\wedge\,\dfrac{\vee_1\,\dfrac{\mathrm{mod}\,\dfrac{q, \neg q}{\Diamond q, \Box\neg q}}{\Diamond\Diamond q, p \vee \Box\Box\neg q}}{(\neg p \vee \Diamond\Diamond q) \wedge (p \vee \Box\Box\neg q)}$$

$$\wedge\,\dfrac{\vee_1\,\dfrac{\mathrm{mod}\,\dfrac{\mathrm{mod}\,\dfrac{q}{\Diamond q}}{\Diamond\Diamond q, p}}{\neg p \vee \Diamond\Diamond q, p \vee \Box\Box\neg q}}{(\neg p \vee \Diamond\Diamond q) \wedge (p \vee \Box\Box\neg q)}$$

Properties

Observation

Every formula of modal logic has at least one (and finitely many) pre-tableaux.

Properties

Observation

Every formula of modal logic has at least one (and finitely many) pre-tableaux.

Question How many pre-tableaux does a formula have?

Properties

Observation

Every formula of modal logic has at least one (and finitely many) pre-tableaux.

Question How many pre-tableaux does a formula have?

Lemma (Exercise)

Let $t = \langle S, \to, \lambda \rangle$ be a tableau and suppose $s \in S$. Then for all $p \in \textsc{Prop}$ we have $\{p, \neg p\} \not\subseteq \lambda(s)$.

Lemma (Soundness)

If φ has a tableau then φ is satisfiable.

Lemma (Soundness)

If φ has a tableau then φ is satisfiable.

Let $t = \langle S, \to, \lambda \rangle$ be a tableau for φ.

Lemma (Soundness)

If φ has a tableau then φ is satisfiable.

Let $t = \langle S, \to, \lambda \rangle$ be a tableau for φ.
Define a structure $M = (W, R, V)$ and a map $\tau \colon S \to W$ such that

Lemma (Soundness)

If φ has a tableau then φ is satisfiable.

Let $t = \langle S, \to, \lambda \rangle$ be a tableau for φ.
Define a structure $M = (W, R, V)$ and a map $\tau \colon S \to W$ such that

1. $\tau(\rho) = w_0 \in W$.

Lemma (Soundness)

If φ has a tableau then φ is satisfiable.

Let $t = \langle S, \to, \lambda \rangle$ be a tableau for φ.
Define a structure $M = (W, R, V)$ and a map $\tau \colon S \to W$ such that

1. $\tau(\rho) = w_0 \in W$.
2. If $s_0 \to s_1$ and the tableau rule applied at s is mod-rule then $(\tau(s_0), \tau(s_1)) \in R$, otherwise $\tau(s_0) = \tau(s_1)$.

Lemma (Soundness)

If φ has a tableau then φ is satisfiable.

Let $t = \langle S, \to, \lambda \rangle$ be a tableau for φ.
Define a structure $M = (W, R, V)$ and a map $\tau \colon S \to W$ such that

1. $\tau(\rho) = w_0 \in W$.
2. If $s_0 \to s_1$ and the tableau rule applied at s is mod-rule then $(\tau(s_0), \tau(s_1)) \in R$, otherwise $\tau(s_0) = \tau(s_1)$.
3. $w \in V(p)$ iff there exists $s \in S$ such that $\tau(s) = w$ and $p \in \lambda(s)$.

Exercise Complete the proof by first proving

- Show that for every $s \in S$ and $\psi \in \lambda(s)$ we have

$$\psi \in \lambda(s) \qquad \text{iff} \qquad M, \tau(s) \models \psi$$

Hint: Double induction i) complexity of ψ & ii) distance of s from leaves.

How to extract the tree-model?

Both of the following are tableaux.

$$\vee_0 \dfrac{\wedge \dfrac{\mathrm{mod} \dfrac{\Box p, q}{\Box\Box p, \Diamond q}}{\Box\Box p \wedge \Diamond q}}{(\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)}$$

$$\vee_0 \dfrac{\wedge \dfrac{\neg p, \Box\neg q}{\neg p \wedge \Box\neg q}}{(\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)}$$

How to extract the tree-model?

Both of the following are tableaux.

$$\vee_0 \dfrac{\wedge \dfrac{\mathrm{mod} \dfrac{\Box p, q}{\Box\Box p, \Diamond q}}{\Box\Box p \wedge \Diamond q}}{(\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)} \qquad \vee_0 \dfrac{\wedge \dfrac{\neg p, \Box\neg q}{\neg p \wedge \Box\neg q}}{(\Box\Box p \wedge \Diamond q) \vee (\neg p \wedge \Box\neg q)}$$

The models they give us are respectively:

$$\begin{cases} W = \{w_0, w_1\} \\ R = \{(w_0, w_1))\} \\ V(p) = \varnothing, V(q) = \{w_1\} \end{cases} \qquad \begin{cases} W = \{w_0\} \\ R = \varnothing \\ V(p) = \varnothing \end{cases}$$

Lemma (Completeness)

If φ is satisfiable then φ has a tableau.

Lemma (Completeness)

If φ is satisfiable then φ has a tableau.

Proof. Let $T = (S, \to, \lambda)$ be a transition system with distinguished node s_0 such that $T, s_0 \models \varphi$.

Lemma (Completeness)

If φ is satisfiable then φ has a tableau.

Proof. Let $T = (S, \to, \lambda)$ be a transition system with distinguished node s_0 such that $T, s_0 \models \varphi$.

- Use the model T as a guiding tool to construct a tableau for φ.

Lemma (Completeness)

If φ is satisfiable then φ has a tableau.

Proof. Let $T = (S, \to, \lambda)$ be a transition system with distinguished node s_0 such that $T, s_0 \models \varphi$.

- Use the model T as a guiding tool to construct a tableau for φ.
- More specifically, build a pre-tableau such that every sequent is satisfied by some $s \in S$.

Lemma (Completeness)

If φ is satisfiable then φ has a tableau.

Proof. Let $T = (S, \to, \lambda)$ be a transition system with distinguished node s_0 such that $T, s_0 \models \varphi$.

- Use the model T as a guiding tool to construct a tableau for φ.
- More specifically, build a pre-tableau such that every sequent is satisfied by some $s \in S$.
- Every path in the pre-tableau correspond to a path in T.

Lemma (Completeness)

If φ is satisfiable then φ has a tableau.

Proof. Let $T = (S, \to, \lambda)$ be a transition system with distinguished node s_0 such that $T, s_0 \models \varphi$.

- Use the model T as a guiding tool to construct a tableau for φ.
- More specifically, build a pre-tableau such that every sequent is satisfied by some $s \in S$.
- Every path in the pre-tableau correspond to a path in T.
- Every leaf is of the form $\Box\Gamma, \Theta$ where Θ is consistent.

Lemma (Completeness)

If φ is satisfiable then φ has a tableau.

Proof. Let $T = (S, \to, \lambda)$ be a transition system with distinguished node s_0 such that $T, s_0 \models \varphi$.

- Use the model T as a guiding tool to construct a tableau for φ.
- More specifically, build a pre-tableau such that every sequent is satisfied by some $s \in S$.
- Every path in the pre-tableau correspond to a path in T.
- Every leaf is of the form $\Box\Gamma, \Theta$ where Θ is consistent.

Exercise. Fill in the details of the proof.

Lemma (Completeness)

If φ is satisfiable then φ has a tableau.

Proof. Let $T = (S, \to, \lambda)$ be a transition system with distinguished node s_0 such that $T, s_0 \models \varphi$.

- Use the model T as a guiding tool to construct a tableau for φ.
- More specifically, build a pre-tableau such that every sequent is satisfied by some $s \in S$.
- Every path in the pre-tableau correspond to a path in T.
- Every leaf is of the form $\Box\Gamma, \Theta$ where Θ is consistent.

Exercise. Fill in the details of the proof.

Corollary

Modal logic has the finite model property.

Exercises

Consider the formulas φ and ψ given by

$$\varphi := \Box p \to \Box\Box p$$
$$\psi := \Diamond(p \to q) \to (\Box p \to \Diamond q)$$

1. Give an equivalent NNF formula for each
2. Using tableaux check if they are valid/satisfiable

Notes

- If you are claiming it is satisfiable then you should provide a tableau and describe the model it gives rise to.
- If you are claiming it is not satisfiable then you must argue that it has no tableaux; you can do this by listing all pre-tableaux and showing none satisfy the tableaux condition.
- Validity of φ can be deduced from satisfiability of $\neg\varphi$.

Expressive limitation of modal logic

Suppose we are interested to express the following property

there is a finite path that reaches a node labelled by p

Expressive limitation of modal logic

Suppose we are interested to express the following property

there is a finite path that reaches a node labelled by p

We can try (and fail) using the modal syntax: $p \vee \Diamond p \vee \Diamond\Diamond p \vee \cdots$

Expressive limitation of modal logic

Suppose we are interested to express the following property

there is a finite path that reaches a node labelled by p

We can try (and fail) using the modal syntax: $p \vee \Diamond p \vee \Diamond\Diamond p \vee \cdots$

Modal language does not allow us to express properties along arbitrary (finite or infinite) paths in a structure. Can we do better?

Expressive limitation of modal logic

Suppose we are interested to express the following property

there is a finite path that reaches a node labelled by p

We can try (and fail) using the modal syntax: $p \vee \Diamond p \vee \Diamond\Diamond p \vee \cdots$

Modal language does not allow us to express properties along arbitrary (finite or infinite) paths in a structure. Can we do better? Yes!

Temporal Logics

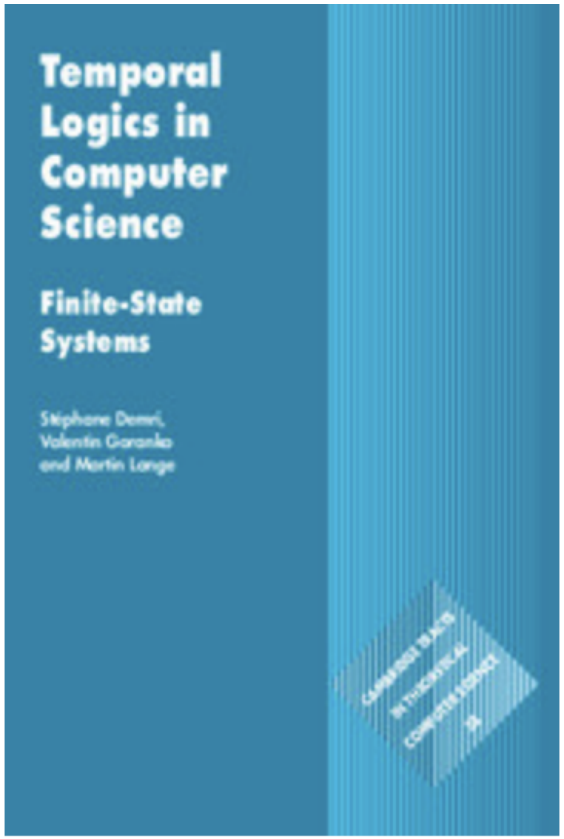
Temporal logics provide us with a language that can express path quantification:

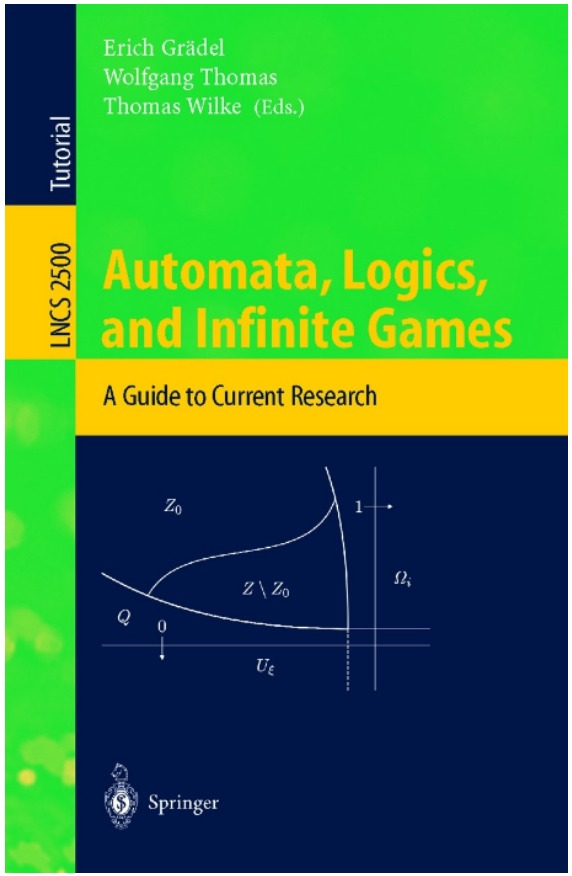
- "along every path ..."
- "there exists a path along which ..."

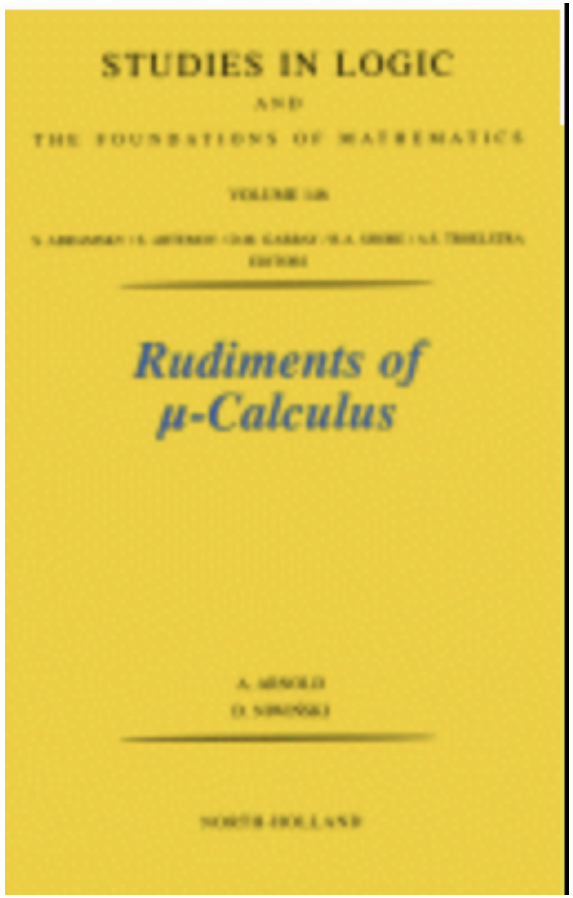
and much more complex properties.

Recap

Recap


Temporal Logics in Computer Science
Finite-State Systems
Stéphane Demri, Valentin Goranko and Martin Lange


Tutorial
LNCS 2500
Erich Grädel
Wolfgang Thomas
Thomas Wilke (Eds.)
Automata, Logics, and Infinite Games
A Guide to Current Research
Springer


STUDIES IN LOGIC
Rudiments of μ-Calculus

LTL-formulas, over atomic propositions $p_1, \cdots, p_n$, are defined

$$\begin{array}{rcll} \varphi & ::= & p_i & \text{atomic proposition} \\ & | & \neg\varphi & \text{negation} \\ & | & \varphi \wedge \psi & \text{conjunction} \\ & | & \varphi \vee \psi & \text{disjunction} \\ & | & \mathbf{X}\,\varphi & \text{next} \\ & | & \varphi\,\mathbf{U}\,\psi & \text{until} \end{array}$$

LTL-formulas, over atomic propositions $p_1, \cdots, p_n$, are defined

$$\begin{array}{rcll} \varphi & ::= & p_i & \text{atomic proposition} \\ & | & \neg\varphi & \text{negation} \\ & | & \varphi \wedge \psi & \text{conjunction} \\ & | & \varphi \vee \psi & \text{disjunction} \\ & | & \mathbf{X}\,\varphi & \text{next} \\ & | & \varphi\,\mathbf{U}\,\psi & \text{until} \end{array}$$

Intended meaning

$\mathbf{X}\,\varphi$ "φ is true at the **next** time-step"

$\varphi\,\mathbf{U}\,\psi$ "φ is true **until** ψ is true (and ψ holds eventually)"

Two additional constructs

$\mathbf{F}\,\varphi$ "φ is **eventually** true"
φ is true at **some** point in the future

$\mathbf{G}\,\varphi$ "φ is **always** true"
φ is true at **every** point in the future (including the present)

Two additional constructs

$\mathbf{F}\,\varphi$ "φ is **eventually** true"
φ is true at **some** point in the future

$\mathbf{G}\,\varphi$ "φ is **always** true"
φ is true at **every** point in the future (including the present)

They are expressible in LTL by

$$\begin{aligned} \mathbf{F}\,\varphi &:= \mathsf{true}\,\mathbf{U}\,\varphi \\ \mathbf{G}\,\varphi &:= \neg(\mathbf{F}\,\neg\varphi) \end{aligned}$$

Examples of properties expressible in LTL

- **Recurrence**: "p_1 holds infinitely often"

$$\mathbf{G}\,(\mathbf{F}\,p_1)$$

Examples of properties expressible in LTL

- **Recurrence**: "p_1 holds infinitely often"

$$\mathbf{G}\,(\mathbf{F}\,p_1)$$

- **Request-response**: "It is *always* the case that whenever p_1 holds, p_2 will hold sometime later"

$$\mathbf{G}\,(p_1 \to \mathbf{X}\,\mathbf{F}\,p_2)$$

Examples of properties expressible in LTL

- **Recurrence**: "p_1 holds infinitely often"

$$\mathbf{G}\,(\mathbf{F}\,p_1)$$

- **Request-response**: "It is *always* the case that whenever p_1 holds, p_2 will hold sometime later"

$$\mathbf{G}\,(p_1 \to \mathbf{X}\,\mathbf{F}\,p_2)$$

- **Fairness**: "If p_1 is true infinitely often, then so is p_2"

$$\mathbf{G}\,\mathbf{F}\,p_1 \to \mathbf{G}\,\mathbf{F}\,p_2$$

Kripke structures revisited

A **(pointed) Kripke structure** over a set of atomic propositions $\{ p_1, \ldots, p_n \}$ is a quadruple (S, R, λ, s_0) with

- a finite state-set S
- an initial/distinguished state $s_0 \in S$
- a transition relation $R \subseteq S \times S$, and
- a labelling function $\lambda \colon S \longrightarrow \mathcal{P}(\{ p_1, \cdots, p_n \})$, associating to each $s \in S$ the set $\lambda(s)$ of those p_i that are satisfied at s.

Kripke structures revisited

A **(pointed) Kripke structure** over a set of atomic propositions $\{ p_1, \ldots, p_n \}$ is a quadruple (S, R, λ, s_0) with

- a finite state-set S
- an initial/distinguished state $s_0 \in S$
- a transition relation $R \subseteq S \times S$, and
- a labelling function $\lambda \colon S \longrightarrow \mathcal{P}(\{ p_1, \cdots, p_n \})$, associating to each $s \in S$ the set $\lambda(s)$ of those p_i that are satisfied at s.

Kripke structures revisited

A **(pointed) Kripke structure** over a set of atomic propositions $\{ p_1, \ldots, p_n \}$ is a quadruple (S, R, λ, s_0) with

- a finite state-set S
- an initial/distinguished state $s_0 \in S$
- a transition relation $R \subseteq S \times S$, and
- a labelling function $\lambda \colon S \longrightarrow \mathcal{P}(\{ p_1, \cdots, p_n \})$, associating to each $s \in S$ the set $\lambda(s)$ of those p_i that are satisfied at s.

Notation

We write $\lambda(s)$ as a bit vector $\begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} \in \mathbb{B}^n$ such that $b_i = 1$ iff $p_i \in \lambda(s)$.

Paths and label sequences

An infinite **path** $s_0 s_1 s_2 \cdots$ through a Kripke structure (S, R, λ, s_0) induces an infinite sequencee of "labels" over the alphabet $\mathbb{B}^n$

$$\lambda(s_0)\,\lambda(s_1)\,\lambda(s_2)\cdots$$

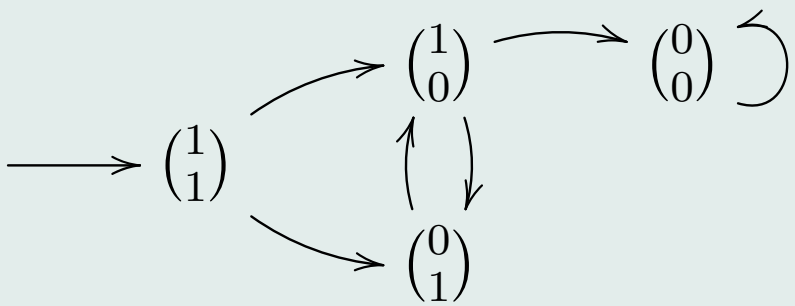
Paths and label sequences

An infinite **path** $s_0 s_1 s_2 \cdots$ through a Kripke structure (S, R, λ, s_0) induces an infinite sequencee of "labels" over the alphabet $\mathbb{B}^n$

$$\lambda(s_0)\,\lambda(s_1)\,\lambda(s_2)\cdots$$

Example

A Kripke structure over $\{p_1, p_2\}$.



$\binom{1}{1}\binom{1}{0}\binom{0}{1}\binom{1}{0}\binom{0}{0}\binom{0}{0}\cdots$
$\binom{1}{1}\binom{0}{1}\binom{1}{0}\binom{0}{1}\binom{1}{0}\binom{0}{0}\binom{0}{0}\cdots$

LTL-formulas over atomic propositions $p_1, \cdots, p_n$ are interpreted as sets of ω-words α over the alphabet $\mathbb{B}^n$.

Notation. Let $\alpha = \alpha(0)\ \alpha(1)\ \cdots \in (\mathbb{B}^n)^\omega$:

1. α^i stands for $\alpha(i)\alpha(i+1)\cdots$, the ***i*-suffix** of α; so $\alpha = \alpha^0$.
2. $(\alpha(i))_j$ is the j-th component of the vector $\alpha(i)$.

LTL-formulas over atomic propositions $p_1, \cdots, p_n$ are interpreted as sets of ω-words α over the alphabet $\mathbb{B}^n$.

Notation. Let $\alpha = \alpha(0)\ \alpha(1)\ \cdots \in (\mathbb{B}^n)^\omega$:

1. α^i stands for $\alpha(i)\alpha(i+1)\cdots$, the ***i*-suffix** of α; so $\alpha = \alpha^0$.
2. $(\alpha(i))_j$ is the j-th component of the vector $\alpha(i)$.

Satisfaction relation. Let $i \geq 0$. Define $\alpha^i \vDash \varphi$ by recursion over φ:

$$
\begin{array}{lll}
\alpha^i \vDash p_j & \text{iff} & (\alpha(i))_j = 1 \\
\alpha^i \vDash \neg\varphi & \text{iff} & (\alpha^i \nvDash \varphi) \\
\alpha^i \vDash \varphi \vee \psi & \text{iff} & \alpha^i \vDash \varphi \ \text{ or } \ \alpha^i \vDash \psi \\
\alpha^i \vDash \varphi \wedge \psi & \text{iff} & \alpha^i \vDash \varphi \ \text{ and } \ \alpha^i \vDash \psi \\
\alpha^i \vDash \mathbf{X}\,\varphi & \text{iff} & \alpha^{i+1} \vDash \varphi \\
\alpha^i \vDash \varphi\,\mathbf{U}\,\psi & \text{iff} & \exists j \geq i : \Big(\alpha^j \vDash \psi \ \wedge\ \forall i \leq k < j : \alpha^k \vDash \varphi\Big)
\end{array}
$$

LTL-formulas over atomic propositions $p_1, \cdots, p_n$ are interpreted as sets of ω-words α over the alphabet $\mathbb{B}^n$.

Notation. Let $\alpha = \alpha(0)\ \alpha(1)\ \cdots \in (\mathbb{B}^n)^\omega$:

1. α^i stands for $\alpha(i)\alpha(i+1)\cdots$, the ***i*-suffix** of α; so $\alpha = \alpha^0$.
2. $(\alpha(i))_j$ is the j-th component of the vector $\alpha(i)$.

Satisfaction relation. Let $i \geq 0$. Define $\alpha^i \vDash \varphi$ by recursion over φ:

$$
\begin{array}{lll}
\alpha^i \vDash p_j & \text{iff} & (\alpha(i))_j = 1 \\
\alpha^i \vDash \neg\varphi & \text{iff} & (\alpha^i \nvDash \varphi) \\
\alpha^i \vDash \varphi \vee \psi & \text{iff} & \alpha^i \vDash \varphi \ \text{ or } \ \alpha^i \vDash \psi \\
\alpha^i \vDash \varphi \wedge \psi & \text{iff} & \alpha^i \vDash \varphi \ \text{ and } \ \alpha^i \vDash \psi \\
\alpha^i \vDash \mathbf{X}\,\varphi & \text{iff} & \alpha^{i+1} \vDash \varphi \\
\alpha^i \vDash \varphi\,\mathbf{U}\,\psi & \text{iff} & \exists j \geq i : \Big(\alpha^j \vDash \psi \ \wedge\ \forall i \leq k < j : \alpha^k \vDash \varphi\Big)
\end{array}
$$

We say α **satisfies** φ, denoted $\alpha \vDash \varphi$, if $\alpha^0 \vDash \varphi$.

Linear vs Branching

- **Linear-time properties** set the same conditions on every infinite path through a system modelled by a Kripke structure.
- **Branching-time properties** are conditions on the structure of the tree formed by all paths through a Kripke structure

See e.g.
Branching vs. linear time: Final showdown, M.Y. Vardi, TACAS 2001.

- We work in **two-sided** sequent calculus so we can handle negation.
- Rules for the propositional part is as usual with left and right **intro/outro rules** for each connective.
- The main task is to deal with $\varphi \,\mathbf{U}\, \psi$.
- We treat this new connective by exploiting the following equivalence

$$\varphi \,\mathbf{U}\, \psi \equiv \psi \vee (\varphi \wedge \mathbf{X}\,(\varphi \,\mathbf{U}\, \psi))$$

- We work in **two-sided** sequent calculus so we can handle negation.
- Rules for the propositional part is as usual with left and right **intro/outro rules** for each connective.
- The main task is to deal with $\varphi \,\mathbf{U}\, \psi$.
- We treat this new connective by exploiting the following equivalence

$$\varphi \,\mathbf{U}\, \psi \equiv \psi \vee (\varphi \wedge \mathbf{X}\,(\varphi \,\mathbf{U}\, \psi))$$

$$X\ \frac{\Gamma \Rightarrow \Delta}{\Sigma, \mathbf{X}\,\Gamma \Rightarrow \mathbf{X}\,\Delta, \Lambda}\ \text{with } \Sigma, \Lambda \text{ arbitrary sequents}$$

$$U_R\ \frac{\Gamma \Rightarrow \Delta, \varphi, \psi \qquad \Gamma \Rightarrow \psi, \mathbf{X}\,(\varphi \,\mathbf{U}\, \psi)}{\Gamma \Rightarrow \Delta, \varphi \,\mathbf{U}\, \psi}$$

$$U_L\ \frac{\Gamma, \psi \Rightarrow \Delta, \qquad \Gamma, \varphi, \mathbf{X}\,(\varphi \,\mathbf{U}\, \psi) \Rightarrow \Delta}{\Gamma, \varphi \,\mathbf{U}\, \psi \Rightarrow \Delta}$$

Non-wellfounded proofs

Pre-proofs are finite branching but can have infinitely long branches

Non-wellfounded proofs

Pre-proofs are finite branching but can have infinitely long branches

We need to know the following notions for a pre-proof

- path
- trace: sequence of formulas along a path tracing the evolution of formulas

Non-wellfounded proofs

Pre-proofs are finite branching but can have infinitely long branches

We need to know the following notions for a pre-proof

- path
- trace: sequence of formulas along a path tracing the evolution of formulas

Notice

- every infinite trace stabilises on the left or right of $\Rightarrow$
- every infinite path take the right premise of U_L/U_R infinitely often

Non-wellfounded proofs

Pre-proofs are finite branching but can have infinitely long branches

We need to know the following notions for a pre-proof

- path
- trace: sequence of formulas along a path tracing the evolution of formulas

Notice

- every infinite trace stabilises on the left or right of $\Rightarrow$
- every infinite path take the right premise of U_L/U_R infinitely often

Proofhood condition every infinite path carries an infinite trace that passes through U_L with the until formula principal.

Example

$$R\,\frac{\varphi\,\mathbf{U}\,\psi \Rightarrow \top, \psi \qquad L\,\dfrac{\psi \Rightarrow \psi, \mathbf{X}\,(\top\,\mathbf{U}\,\psi) \qquad X\,\dfrac{\varphi\,\mathbf{U}\,\psi \Rightarrow \top\,\mathbf{U}\,\psi}{\varphi, \mathbf{X}\,(\varphi\,\mathbf{U}\,\psi) \Rightarrow \psi, \mathbf{X}\,(\top\,\mathbf{U}\,\psi)}}{\varphi\,\mathbf{U}\,\psi \Rightarrow \psi, \mathbf{X}\,(\top\,\mathbf{U}\,\psi)}}{\varphi\,\mathbf{U}\,\psi \Rightarrow \top\,\mathbf{U}\,\psi}$$

LTL finitary axiomatisation

We need to add an **induction** axiom to handle the until operator:

$$\text{ind}\ \frac{\psi \vee (\varphi \wedge \mathbf{X}\,\delta) \to \delta}{\varphi\,\mathbf{U}\,\psi \to \delta}$$

LTL finitary axiomatisation

We need to add an **induction** axiom to handle the until operator:

$$\text{ind}\ \frac{\psi \vee (\varphi \wedge \mathbf{X}\,\delta) \to \delta}{\varphi\,\mathbf{U}\,\psi \to \delta}$$

Or equivalently,

$$\text{ind}\ \frac{\psi \to \delta \qquad \varphi \wedge \mathbf{X}\,\delta \to \delta}{\varphi\,\mathbf{U}\,\psi \to \delta}$$

Outline

Section 3

Cyclic and Ill-founded Proofs

Proof Systems à la Hilbert

A proof of a sequent Γ in a given system $\mathscr{S}$ is a **finite tree** in which leaves are labelled by axioms, the labelling of inner nodes respect the inference rules of $\mathscr{S}$, and the root is labelled by Γ.

Proof Systems à la Hilbert

A proof of a sequent Γ in a given system $\mathscr{S}$ is a **finite tree** in which leaves are labelled by axioms, the labelling of inner nodes respect the inference rules of $\mathscr{S}$, and the root is labelled by Γ.

But these 'concrete' proofs are hard to produce because of

- non-determinism
- induction 'axioms'

Cyclic Proofs: the core idea

Geared towards *proof search*, a cyclic proof formalises **an infinite decent argument**.

Cyclic Proofs: the core idea

Geared towards *proof search*, a cyclic proof formalises **an infinite decent argument**.

$$\sqrt{2} = p/q \quad \rightarrow \quad p^2 = 2q^2 \quad \rightarrow \quad p(p-q) = q(2q-p)$$

Cyclic Proofs: the core idea

Geared towards *proof search*, a cyclic proof formalises **an infinite decent argument**.

$$\sqrt{2} = p/q \quad \rightarrow \quad p^2 = 2q^2 \quad \rightarrow \quad p(p-q) = q(2q-p)$$

$$\sqrt{2} = p/q = (2q-p)/(p-q)$$

Cyclic Proofs: the core idea

Geared towards *proof search*, a cyclic proof formalises **an infinite decent argument**.

$$\sqrt{2} = p/q \quad \rightarrow \quad p^2 = 2q^2 \quad \rightarrow \quad p(p-q) = q(2q-p)$$

$$\sqrt{2} = p/q = (2q-p)/(p-q)$$

$$\sqrt{2} = p/q = p'/q' = p''/q'' = \cdots \qquad \text{with } q > q' > q'' > \cdots$$

Cyclic proofs for modal logic K4

Formulas $p \mid \overline{p} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond\varphi \mid \Box\varphi$

Sequents $\Gamma, \Delta, \ldots$ finite sets of formulas treated as disjunction

Axioms $\Gamma, \varphi, \bar{\varphi}$

Inference rules

$$\frac{\Gamma, \bot}{\Gamma}\,\bot \qquad \frac{\Gamma, \varphi \qquad \Gamma, \psi}{\Gamma, \varphi \wedge \psi}\,\wedge \qquad \frac{\Gamma, \varphi, \psi}{\Gamma, \varphi \vee \psi}\,\vee \qquad \frac{\Gamma, \Diamond\Gamma, \varphi}{\Diamond\Gamma, \Box\varphi, \Delta}\,4$$

where $\Diamond\Gamma$ abbreviates $\{\Diamond\varphi_1, \Diamond\varphi_2, \ldots, \Diamond\varphi_n\}$ for $\Gamma = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$.

Cyclic proofs for modal logic K4

Formulas $p \mid \overline{p} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond\varphi \mid \Box\varphi$

Sequents $\Gamma, \Delta, \ldots$ finite sets of formulas treated as disjunction

Axioms $\Gamma, \varphi, \bar{\varphi}$

Inference rules

$$\frac{\Gamma, \bot}{\Gamma}\,\bot \qquad \frac{\Gamma, \varphi \qquad \Gamma, \psi}{\Gamma, \varphi \wedge \psi}\,\wedge \qquad \frac{\Gamma, \varphi, \psi}{\Gamma, \varphi \vee \psi}\,\vee \qquad \frac{\Gamma, \Diamond\Gamma, \varphi}{\Diamond\Gamma, \Box\varphi, \Delta}\,4$$

where $\Diamond\Gamma$ abbreviates $\{\Diamond\varphi_1, \Diamond\varphi_2, \ldots, \Diamond\varphi_n\}$ for $\Gamma = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$.

- **axiom 4:** $\Box\varphi \to \Box\Box\varphi$
- **Löb's axiom:** $\Box(\Box p \to p) \to \Box p$.

A cyclic proof of Löb's axiom

$$\dfrac{\dfrac{\dfrac{\dfrac{\Box p \wedge \bar{p}, \Diamond(\Box p \wedge \bar{p}), p}{\Box p, \Diamond(\Box p \wedge \bar{p}), p}\,4 \qquad \bar{p}, \Diamond(\Box p \wedge \bar{p}), p}{\Box p \wedge \bar{p}, \Diamond(\Box p \wedge \bar{p}), p}\,\wedge}{\Diamond(\Box p \wedge \bar{p}), \Box p}\,4}{\Diamond(\Box p \wedge \bar{p}) \vee \Box p}\,\vee$$

A cyclic proof of Löb's axiom

$$\dfrac{\dfrac{\dfrac{\dfrac{\Box p \wedge \bar{p}, \Diamond(\Box p \wedge \bar{p}), p}{\Box p, \Diamond(\Box p \wedge \bar{p}), p}\,4 \qquad \bar{p}, \Diamond(\Box p \wedge \bar{p}), p}{\Box p \wedge \bar{p}, \Diamond(\Box p \wedge \bar{p}), p}\,\wedge}{\Diamond(\Box p \wedge \bar{p}), \Box p}\,4}{\Diamond(\Box p \wedge \bar{p}) \vee \Box p}\,\vee$$

Theorem (Shamkanov 2014)

K4 + circularity = Gödel-Löb logic

Gödel-Löb logic axiomatisable by

Boolean tautologies + $\Box$-distribution + Löb axiom + MP

A very general definition

Definition. A cyclic proof of Γ in $\mathscr{S}$ is a derivation tree which can have both axiomatic and **non-axiomatic** leaves as long as the latter satisfy the property:

A very general definition

Definition. A cyclic proof of Γ in $\mathscr{S}$ is a derivation tree which can have both axiomatic and **non-axiomatic** leaves as long as the latter satisfy the property:

1. if the non-axiomatic leaf is labelled by Δ then there is another node labelled by Δ which occurs earlier in the derivation tree
2. on every cycle of the induced graph **progress** is made

A very general definition

Definition. A cyclic proof of Γ in $\mathscr{S}$ is a derivation tree which can have both axiomatic and **non-axiomatic** leaves as long as the latter satisfy the property:

1. if the non-axiomatic leaf is labelled by Δ then there is another node labelled by Δ which occurs earlier in the derivation tree
2. on every cycle of the induced graph **progress** is made

What is progress? How is it determined?

Non-example

$$\dfrac{\dfrac{\dfrac{}{\bot}\,\text{weak.}}{\bot, \bot}}{\bot}\,\text{contr.}$$

Fixpoint-example

Fixpoint-example

- p is reachable $\quad R =_\mu p \vee \Diamond R$
- p is reachable in an even number of steps $\quad E =_\mu p \vee \Diamond\Diamond E$
- p is reachable in an odd number of steps $\quad O =_\mu \Diamond(p \vee \Diamond O)$

The following is valid: $\quad R \to E \vee O$

Fixpoint-example

- p is reachable $\quad R =_\mu p \vee \Diamond R$
- p is reachable in an even number of steps $\quad E =_\mu p \vee \Diamond\Diamond E$
- p is reachable in an odd number of steps $\quad O =_\mu \Diamond(p \vee \Diamond O)$

The following is valid: $R \to E \vee O$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{R \to E, O}{\Diamond R \to p, \Diamond E, \Diamond O}\,\Diamond \qquad p \to \cdots p \cdots}{(\Diamond R \vee p) \to p, \Diamond E, \Diamond O}\,\vee_l}{R \to \Diamond E, (p \vee \Diamond O)}\,R, \vee_r}{\Diamond R \to p, \Diamond\Diamond E, \Diamond(p \vee \Diamond O)}\,\Diamond}{\Diamond R \to p, \Diamond\Diamond E, O}\,O \qquad \dfrac{p \to p, \Diamond\Diamond E, O}{(p \vee \Diamond R) \to (p \vee \Diamond\Diamond E), O}\,\vee_l, \vee_r}{R \to E, O}\,R,E$$

Proof by induction vs cyclic proofs

- principle of induction
- local
- non-algorithmic

- principle of infinite decent
- global
- algorithmic
- symmetric treatment of inductive and co-inductive properties

Outline

Section 4

Modal μ-calculus

Syntax: $p \mid \bar{p} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid \Diamond\varphi$

Semantics: For directed labelled graph $T = (S, \to, \lambda)$:

$$\|p\|^T = \{u \in S \mid p \in \lambda(u)\}$$

$$\|\varphi \wedge \psi\|^T = \|\varphi\|_V^T \cap \|\psi\|^T$$

$$\|\Box\varphi\|^T = \{u \in S \mid \text{for all } v, \text{ if } u \to v \text{ then } v \in \|\varphi\|^T\ \}$$

Examples: $q \vee \Diamond q \vee \Diamond\Diamond q$; $\underbrace{\Box \cdots \Box}_{n} \Diamond p$.

Syntax: $p \mid \bar{p} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid \Diamond\varphi$

Semantics: For directed labelled graph $T = (S, \to, \lambda)$:

$$\|p\|^T = \{u \in S \mid p \in \lambda(u)\}$$
$$\|\varphi \wedge \psi\|^T = \|\varphi\|_V^T \cap \|\psi\|^T$$
$$\|\Box\varphi\|^T = \{u \in S \mid \text{for all } v, \text{ if } u \to v \text{ then } v \in \|\varphi\|^T\ \}$$

Examples: $q \vee \Diamond q \vee \Diamond\Diamond q$; $\underbrace{\Box \cdots \Box}_{n} \Diamond p$.

Expressive limitation: properties along arbitrary (finite or infinite) paths in a structure cannot be presented

Syntax: $p \mid \bar{p} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid \Diamond\varphi$

Semantics: For directed labelled graph $T = (S, \to, \lambda)$:

$$\|p\|^T = \{u \in S \mid p \in \lambda(u)\}$$
$$\|\varphi \wedge \psi\|^T = \|\varphi\|^T_V \cap \|\psi\|^T$$
$$\|\Box\varphi\|^T = \{u \in S \mid \text{for all } v, \text{ if } u \to v \text{ then } v \in \|\varphi\|^T\ \}$$

Examples: $q \vee \Diamond q \vee \Diamond\Diamond q$; $\underbrace{\Box \cdots \Box}_{n} \Diamond p$.

Expressive limitation: properties along arbitrary (finite or infinite) paths in a structure cannot be presented

Infinite path quantification:

1. $s \models \bigvee_n \Diamond^n q$ – q is reachable from s
2. $s \models \bigwedge_n \Box^n \Diamond p$ – All paths from s always satisfy $\Diamond p$

1. $s \models q \vee \Diamond q \vee \Diamond\Diamond q \vee \cdots$ iterating $\mathrm{x} \mapsto q \vee \Diamond \mathrm{x}$ on $\bot$

1. $s \models q \vee \Diamond q \vee \Diamond\Diamond q \vee \cdots$ iterating $\mathsf{x} \mapsto q \vee \Diamond \mathsf{x}$ on $\bot$

Given $\varphi(\mathsf{x})$ (positive in x) we introduce

- $\mu \mathsf{x}\, \varphi =$ least fixed point of $\mathsf{x} \mapsto \varphi(\mathsf{x})$ Liveness **inductive**
- $\nu \mathsf{x}\, \varphi =$ greatest fixed point of $\mathsf{x} \mapsto \varphi(\mathsf{x})$ Safety **co-inductive**

1. $s \models q \vee \Diamond q \vee \Diamond\Diamond q \vee \cdots$ iterating $\mathsf{x} \mapsto q \vee \Diamond \mathsf{x}$ on $\bot$

Given $\varphi(\mathsf{x})$ (positive in x) we introduce

- $\mu \mathsf{x}\, \varphi =$ least fixed point of $\mathsf{x} \mapsto \varphi(\mathsf{x})$ Liveness **inductive**
- $\nu \mathsf{x}\, \varphi =$ greatest fixed point of $\mathsf{x} \mapsto \varphi(\mathsf{x})$ Safety **co-inductive**

A note on expressiveness

$$\mu \mathsf{x}(p \vee \Box \mathsf{x}) : \text{all paths contain a } p$$
$$\mu \mathsf{x}(p \vee \Box \mathsf{x}) \neq p \vee \Box p \vee \Box\Box p \vee \cdots$$

1. $s \models q \vee \Diamond q \vee \Diamond\Diamond q \vee \cdots$ iterating $\mathsf{x} \mapsto q \vee \Diamond \mathsf{x}$ on $\bot$

Given $\varphi(\mathsf{x})$ (positive in x) we introduce

- $\mu \mathsf{x}\, \varphi$ = least fixed point of $\mathsf{x} \mapsto \varphi(\mathsf{x})$ Liveness **inductive**
- $\nu \mathsf{x}\, \varphi$ = greatest fixed point of $\mathsf{x} \mapsto \varphi(\mathsf{x})$ Safety **co-inductive**

A note on expressiveness

$$\mu \mathsf{x}(p \vee \Box \mathsf{x}) : \text{all paths contain a } p$$
$$\mu \mathsf{x}(p \vee \Box \mathsf{x}) \neq p \vee \Box p \vee \Box\Box p \vee \cdots$$

$$\mu \mathsf{x}(p \vee \Box \mathsf{x}) = p \vee \Box p \vee \Box\Box p \vee \cdots \vee \Box^{\omega} p \vee \cdots$$

Modal μ-calculus

Syntax $p \mid \overline{p} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond\varphi \mid \Box\varphi \mid \mathsf{x} \mid \mu\mathsf{x}\,\varphi \mid \nu\mathsf{x}\,\varphi$

Semantics Fix $T = (S, \to, \lambda)$. Given $\varphi(\mathsf{x})$ define

$$f_\varphi \colon 2^S \to 2^S$$
$$U \mapsto \|\varphi(U)\|^T$$

Modal μ-calculus

Syntax $p \mid \overline{p} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond\varphi \mid \Box\varphi \mid \mathsf{x} \mid \mu\mathsf{x}\,\varphi \mid \nu\mathsf{x}\,\varphi$

Semantics Fix $T = (S, \to, \lambda)$. Given $\varphi(\mathsf{x})$ define

$$f_\varphi \colon 2^S \to 2^S$$
$$U \mapsto \|\varphi(U)\|^T$$

$\|\mu\mathsf{x}\,\varphi\|^T =$ least fixed point of f_φ $\quad \|\nu\mathsf{x}\,\varphi\|^T =$ greatest fixed point of f_φ

$$= \bigcap \{U \subseteq S \mid f_\varphi(U) \subseteq U\} \qquad = \bigcup \{U \subseteq S \mid U \subseteq f_\varphi(U)\}$$

Modal μ-calculus

Syntax $p \mid \overline{p} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond\varphi \mid \Box\varphi \mid \mathsf{x} \mid \mu\mathsf{x}\,\varphi \mid \nu\mathsf{x}\,\varphi$

Semantics Fix $T = (S, \to, \lambda)$. Given $\varphi(\mathsf{x})$ define

$$f_\varphi \colon 2^S \to 2^S$$
$$U \mapsto \|\varphi(U)\|^T$$

$\|\mu\mathsf{x}\,\varphi\|^T =$ least fixed point of f_φ $\quad \|\nu\mathsf{x}\,\varphi\|^T =$ greatest fixed point of f_φ

$$= \bigcap \{U \subseteq S \mid f_\varphi(U) \subseteq U\} \qquad = \bigcup \{U \subseteq S \mid U \subseteq f_\varphi(U)\}$$

Duality Define $\overline{\varphi}$ as the De Morgan dual of φ:

$$\overline{\mu\mathsf{x}\varphi(\mathsf{x})} = \nu\mathsf{x}\overline{\varphi}(\mathsf{x}) \qquad \|\overline{\varphi}\|^T = S \setminus \|\varphi\|^K$$

ν is **looping/safety** and μ is **finite looping/liveness**

ν is **looping/safety** and μ is **finite looping/liveness**

- $\nu x(\Box x \wedge \varphi)$: **always** φ
- $\mu x(\Diamond x \vee \varphi)$: **reachable** φ

ν is **looping/safety** and μ is **finite looping/liveness**

- $\nu x(\Box x \wedge \varphi)$: **always** φ
- $\mu x(\Diamond x \vee \varphi)$: **reachable** φ
- $\mu x(\psi \vee (\varphi \wedge \Box x))$: φ **until** ψ
- $\nu x \Box(\varphi \wedge x)$: φ is **common knowledge**

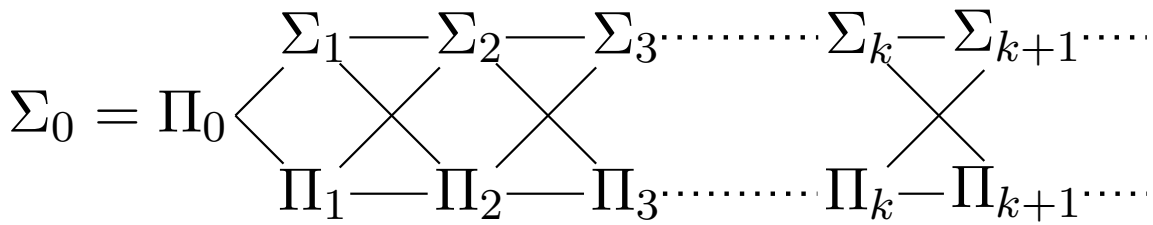
ν is **looping/safety** and μ is **finite looping/liveness**

- $\nu \mathsf{x}(\Box \mathsf{x} \wedge \varphi)$: **always** φ
- $\mu \mathsf{x}(\Diamond \mathsf{x} \vee \varphi)$: **reachable** φ
- $\mu \mathsf{x}(\psi \vee (\varphi \wedge \Box \mathsf{x}))$: φ **until** ψ
- $\nu \mathsf{x} \Box(\varphi \wedge \mathsf{x})$: φ is **common knowledge**
- $\nu \mathsf{x} \mu \mathsf{y}\big((p \vee \Diamond \mathsf{y}) \wedge \Diamond \mathsf{x}\big)$: a path along which p **holds infinitely often**

ν is **looping/safety** and μ is **finite looping/liveness**

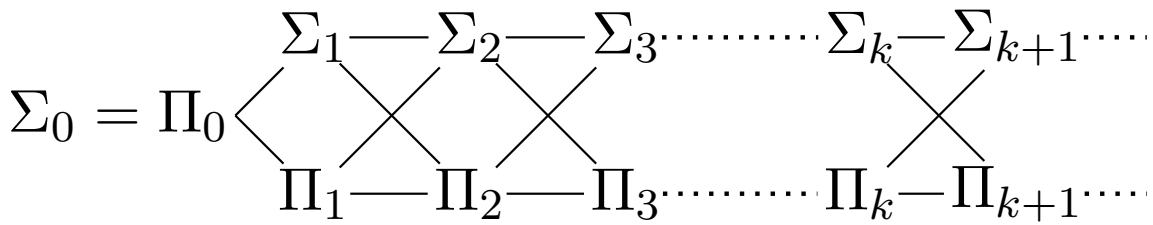
- $\nu \mathsf{x}(\Box \mathsf{x} \wedge \varphi)$: **always** φ
- $\mu \mathsf{x}(\Diamond \mathsf{x} \vee \varphi)$: **reachable** φ
- $\mu \mathsf{x}(\psi \vee (\varphi \wedge \Box \mathsf{x}))$: φ **until** ψ
- $\nu \mathsf{x} \Box(\varphi \wedge \mathsf{x})$: φ is **common knowledge**
- $\nu \mathsf{x} \mu \mathsf{y}\big((p \vee \Diamond \mathsf{y}) \wedge \Diamond \mathsf{x}\big)$: a path along which p **holds infinitely often**
- $\nu \mathsf{x}\big(\mu \mathsf{y}(p \vee \Diamond \mathsf{y}) \wedge \Diamond \mathsf{x}\big)$: a path along which p **is always reachable**

- Count alternations between the two quantifiers **Simple hierarchy**
- Count the *genuine* alternations between least and greatest fixed point quantifiers (**depth**) **Alternation hierarchy**



This hierarchy is strict (Bradfield 1998).

- Count alternations between the two quantifiers **Simple hierarchy**
- Count the *genuine* alternations between least and greatest fixed point quantifiers (**depth**) **Alternation hierarchy**



This hierarchy is strict (Bradfield 1998).

Connexion:

- LTL can be captured by μ-calculus formulas of depth 2
- CTL, PDL can be captured by alternation-free μ-calculus
- CTL*: depth 3

- Expressive: LTL, PDL, CTL, CTL*
- Robustly decidability: model checking, satisfiability

- Expressive: LTL, PDL, CTL, CTL*
- Robustly decidability: model checking, satisfiability
- Closed under bisimulation — but logic is not compact.

- Expressive: LTL, PDL, CTL, CTL*
- Robustly decidability: model checking, satisfiability
- Closed under bisimulation — but logic is not compact.
- Equivalent to: alternating parity automata, parity games

- Expressive: LTL, PDL, CTL, CTL*
- Robustly decidability: model checking, satisfiability
- Closed under bisimulation — but logic is not compact.
- Equivalent to: alternating parity automata, parity games
- Tree Model Property: every satisfiable formula has a tree model.
- Finite Model Property: every satisfiable formula has a finite model.

Model Checking Problem *Given an LTS T, a state s, and a closed modal mu-calculus formula φ, does $s \in \|\varphi\|^T_\varnothing$ hold?*

Decidable (when transition systems T is finite), though its exact complexity is a long standing open problem. The best known bound is $\mathbf{NP} \cap \mathbf{co\text{-}NP}$. The problem is conjectured to be in $\mathbf{P}$.

Model Checking Problem *Given an LTS T, a state s, and a closed modal mu-calculus formula φ, does $s \in \|\varphi\|^T_\varnothing$ hold?*

Decidable (when transition systems T is finite), though its exact complexity is a long standing open problem. The best known bound is $\mathbf{NP} \cap \mathbf{co\text{-}NP}$. The problem is conjectured to be in $\mathbf{P}$.

Satisfiability Problem: *Given a closed modal mu-calculus formula φ, is it satisfiable?* I.e. is there an LTS T with a state s such that $s \in \|\varphi\|^T_\varnothing$?

Decidable: **EXPTIME**-complete.

Model Checking Problem *Given an LTS T, a state s, and a closed modal mu-calculus formula φ, does $s \in \|\varphi\|^T_\varnothing$ hold?*

Decidable (when transition systems T is finite), though its exact complexity is a long standing open problem. The best known bound is $\mathbf{NP} \cap \mathbf{co\text{-}NP}$. The problem is conjectured to be in $\mathbf{P}$.

Satisfiability Problem: *Given a closed modal mu-calculus formula φ, is it satisfiable?* I.e. is there an LTS T with a state s such that $s \in \|\varphi\|^T_\varnothing$?

Decidable: **EXPTIME**-complete.

Tree Model Property: *If a formula is satisfiable, then it is satisfiable in an LTS which is a (bounded degree, possibly infinite) tree.*

Model Checking Problem *Given an LTS T, a state s, and a closed modal mu-calculus formula φ, does $s \in \|\varphi\|^T_\varnothing$ hold?*

Decidable (when transition systems T is finite), though its exact complexity is a long standing open problem. The best known bound is $\mathbf{NP} \cap \mathbf{co\text{-}NP}$. The problem is conjectured to be in $\mathbf{P}$.

Satisfiability Problem: *Given a closed modal mu-calculus formula φ, is it satisfiable?* I.e. is there an LTS T with a state s such that $s \in \|\varphi\|^T_\varnothing$?

Decidable: **EXPTIME**-complete.

Tree Model Property: *If a formula is satisfiable, then it is satisfiable in an LTS which is a (bounded degree, possibly infinite) tree.*

Small Model Property: *If a modal mu-calculus formula is satisfiable, then it is satisfiable by an LTS of size exponential in size of the formula.*

Outline

Section 5

Proof Systems

$$\frac{}{\varphi, \overline{\varphi}}\ \varphi \qquad \frac{\Gamma}{\Gamma, \varphi}\ \text{weak}$$

$$\frac{\Gamma, \varphi, \psi}{\Gamma, \varphi \vee \psi}\ \vee \qquad \frac{\Gamma, \varphi \qquad \Gamma, \psi}{\Gamma, \varphi \wedge \psi}\ \wedge \qquad \frac{\Gamma, \varphi}{\Diamond\Gamma, \Box\varphi}\ \text{mod}$$

$$\frac{\Gamma, \varphi(\sigma \mathsf{x} \varphi(\mathsf{x}))}{\Gamma, \sigma \mathsf{x} \varphi}\ \sigma \qquad \frac{\psi, \varphi(\overline{\psi})}{\psi, \nu \mathsf{x} \varphi}\ \text{ind} \qquad \frac{\Gamma, \overline{\varphi} \qquad \Gamma, \varphi}{\Gamma}\ \text{cut}$$

$$\frac{}{\varphi, \overline{\varphi}}\ \varphi \qquad \frac{\Gamma}{\Gamma, \varphi}\ \text{weak}$$

$$\frac{\Gamma, \varphi, \psi}{\Gamma, \varphi \vee \psi}\ \vee \qquad \frac{\Gamma, \varphi \qquad \Gamma, \psi}{\Gamma, \varphi \wedge \psi}\ \wedge \qquad \frac{\Gamma, \varphi}{\Diamond\Gamma, \Box\varphi}\ \text{mod}$$

$$\frac{\Gamma, \varphi(\sigma \mathsf{x} \varphi(\mathsf{x}))}{\Gamma, \sigma \mathsf{x} \varphi}\ \sigma \qquad \frac{\psi, \varphi(\overline{\psi})}{\psi, \nu \mathsf{x} \varphi}\ \text{ind} \qquad \frac{\Gamma, \overline{\varphi} \qquad \Gamma, \varphi}{\Gamma}\ \text{cut}$$

Abbreviation.

- " , " read as disjunction
- $\Gamma = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ finite set of formulas
- $\Diamond\Gamma := \{\Diamond\varphi_1, \Diamond\varphi_2, \ldots, \Diamond\varphi_n\}$
- $\overline{\Gamma} := \overline{\varphi}_1 \wedge \overline{\varphi}_2 \wedge \ldots \wedge \overline{\varphi}_n$
- $\sigma \in \{\mu, \nu\}$

Validity and proofs

A closed formula φ is **valid** iff $\|\varphi\|^T = S$ for every $T = (S, \to, \lambda)$.

Validity and proofs

A closed formula φ is **valid** iff $\|\varphi\|^T = S$ for every $T = (S, \to, \lambda)$.

Theorem (Kozen 1983; Walukiewicz 2000)

For every closed φ we have $\models \varphi$ *iff* $\mathsf{Koz} \vdash \varphi$.

Validity and proofs

A closed formula φ is **valid** iff $\|\varphi\|^T = S$ for every $T = (S, \to, \lambda)$.

Theorem (Kozen 1983; Walukiewicz 2000)

For every closed φ we have $\models \varphi$ *iff* $\mathsf{Koz} \vdash \varphi$.

Soundness Proved by Kozen (1983).
Completeness aconjuntive fragment; full μ-calculus

1. Completeness of disjunctive formulas: tableaux.
2. Provable equivalence between disjunctive and μ-formulas: tableaux, games, automata.

$$p \mid \overline{p} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond\varphi \mid \Box\varphi \mid \mathsf{x} \mid \mu\mathsf{x}\,\varphi \mid \nu\mathsf{x}\,\varphi$$

μ-quantifier: least fixed point

- $\varphi(\mu\mathsf{x}\varphi) \to \mu\mathsf{x}\varphi$
- $\varphi(\psi) \to \psi \vdash \mu\mathsf{x}\varphi \to \psi$

ν-quantifier: greatest fixed point

- $\nu\mathsf{x}\varphi \to \varphi(\nu\mathsf{x}\varphi)$
- $\psi \to \varphi(\psi) \vdash \psi \to \nu\mathsf{x}\varphi$

ν-regenration

$$\frac{\Gamma, \varphi(\nu\mathsf{x}\varphi(\mathsf{x}))}{\Gamma, \nu\mathsf{x}\varphi}\ \nu$$

ν-induction

$$\frac{\psi, \varphi(\overline{\psi})}{\psi, \nu\mathsf{x}\varphi}\ \text{ind}$$

Axioms and rules of proof system Fix

$$\frac{}{\varphi, \bar{\varphi}} \qquad \frac{\Gamma}{\Gamma, \varphi}\ \text{weak}$$

$$\frac{\Gamma, \varphi, \psi}{\Gamma, \varphi \vee \psi}\ \vee \qquad \frac{\Gamma, \varphi \qquad \Gamma, \psi}{\Gamma, \varphi \wedge \psi}\ \wedge \qquad \frac{\Gamma, \varphi}{\Diamond\Gamma, \Box\varphi}\ \text{mod}$$

$$\frac{\Gamma, \varphi(\mu x \varphi)}{\Gamma, \mu x \varphi}\ \mu \qquad \frac{\Gamma, \varphi(\nu x \varphi)}{\Gamma, \nu x \varphi}\ \nu$$

A **pre-tableau** is a finitely branching tree in axioms and rules of Fix.

Axioms and rules of proof system Fix

$$\frac{}{\varphi, \bar{\varphi}} \qquad \frac{\Gamma}{\Gamma, \varphi}\ \text{weak}$$

$$\frac{\Gamma, \varphi, \psi}{\Gamma, \varphi \vee \psi}\ \vee \qquad \frac{\Gamma, \varphi \qquad \Gamma, \psi}{\Gamma, \varphi \wedge \psi}\ \wedge \qquad \frac{\Gamma, \varphi}{\Diamond\Gamma, \Box\varphi}\ \text{mod}$$

$$\frac{\Gamma, \varphi(\mu x \varphi)}{\Gamma, \mu x \varphi}\ \mu \qquad \frac{\Gamma, \varphi(\nu x \varphi)}{\Gamma, \nu x \varphi}\ \nu$$

A **pre-tableau** is a finitely branching tree in axioms and rules of Fix.

Principal and **residual** formulas are defined analogous to modal logic.

Examples of pre-tableaux

The formula $(\nu \mathsf{x} \Diamond \mathsf{x}) \vee (\mu \mathsf{y} \Box \mathsf{y})$ has (essentially) one pre-tableau:

Examples of pre-tableaux

The formula $(\nu x \Diamond x) \vee (\mu y \Box y)$ has (essentially) one pre-tableau:

$$\dfrac{\dfrac{\dfrac{\dfrac{\nu x \Diamond x, \mu y \Box y}{\Diamond \nu x \Diamond x, \Box \mu y \Box y}\,\text{mod}}{\Diamond \nu x \Diamond x, \mu y \Box y}\,\mu}{\nu x \Diamond x, \mu y \Box y}\,\nu}{\nu x \Diamond x \vee \mu y \Box y}\,\vee$$

Fix a pre-tableau $t = \langle S, \to, \lambda \rangle$ for Γ and a path $\mathbb{P}$ through t. A finite sequence of formulas $\varphi_0, \varphi_1, \ldots, \varphi_n$ is a **trace** through $\mathbb{P}$ if

- $\varphi_i \in \lambda(\mathbb{P}(i))$ for each $i \leq n$;
- $\varphi_{i+1} = \varphi_i$ if φ_i is not principal in the rule from $\mathbb{P}(i)$ to $\mathbb{P}(i+1)$
- φ_{i+1} is the residual subformula of φ_i in the label of $\mathbb{P}(i+1)$.

An infinite seq. of formulas $(\varphi_i)_i$ is a **trace** if every initial seq. is a trace.

Fix a pre-tableau $t = \langle S, \to, \lambda \rangle$ for Γ and a path $\mathbb{P}$ through t. A finite sequence of formulas $\varphi_0, \varphi_1, \ldots, \varphi_n$ is a **trace** through $\mathbb{P}$ if

- $\varphi_i \in \lambda(\mathbb{P}(i))$ for each $i \leq n$;
- $\varphi_{i+1} = \varphi_i$ if φ_i is not principal in the rule from $\mathbb{P}(i)$ to $\mathbb{P}(i+1)$
- φ_{i+1} is the residual subformula of φ_i in the label of $\mathbb{P}(i+1)$.

An infinite seq. of formulas $(\varphi_i)_i$ is a **trace** if every initial seq. is a trace.

Lemma

For every infinite trace there exists a formula $\sigma x \varphi$ that appears infinitely often and is the shortest formula occurring infinitely often.

Example of traces

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\nu x \Diamond x, \mu y \Box y}{\Diamond \nu x \Diamond x, \Box \mu y \Box y}\,\text{mod}}{\Diamond \nu x \Diamond x, \mu y \Box y}\,\mu}{\nu x \Diamond x, \mu y \Box y}\,\nu}{\nu x \Diamond x \vee \mu y \Box y}\,\vee$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\nu x \Diamond x, \mu y \Box y}{\Diamond \nu x \Diamond x, \Box \mu y \Box y}\,\text{mod}}{\Diamond \nu x \Diamond x, \mu y \Box y}\,\mu}{\nu x \Diamond x, \mu y \Box y}\,\nu}{\nu x \Diamond x \vee \mu y \Box y}\,\vee$$

μ-traces and ν-traces

In each infinite trace the unique $\sigma x \varphi$ identified by the lemma will be referred to as the "**most significant formula**" of the trace.

μ-traces and ν-traces

In each infinite trace the unique $\sigma x \varphi$ identified by the lemma will be referred to as the "**most significant formula**" of the trace.

We call an infinite trace

- a **μ-trace** if $\sigma = \mu$
- a **ν-trace** if $\sigma = \nu$

μ-traces and ν-traces

In each infinite trace the unique $\sigma x \varphi$ identified by the lemma will be referred to as the "**most significant formula**" of the trace.

We call an infinite trace

- a **μ-trace** if $\sigma = \mu$
- a **ν-trace** if $\sigma = \nu$

A pre-tableau is a **tableau** if

1. the sequent at each leaf is an axiom, and
2. every infinite path contains a **ν-trace** (progressive thread).

Examples

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\nu x \Diamond x, \mu y \Box y}{\Diamond \nu x \Diamond x, \Box \mu y \Box y}\,\text{mod}}{\Diamond \nu x \Diamond x, \mu y \Box y}\,\mu}{\nu x \Diamond x, \mu y \Box y}\,\nu}{\nu x \Diamond x \vee \mu y \Box y}\,\vee \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\nu x \Diamond x, \mu y \Box y}{\Diamond \nu x \Diamond x, \Box \mu y \Box y}\,\text{mod}}{\Diamond \nu x \Diamond x, \mu y \Box y}\,\mu}{\nu x \Diamond x, \mu y \Box y}\,\nu}{\nu x \Diamond x \vee \mu y \Box y}\,\vee$$

Question What kind of traces do we have here?

It is not hard to check (semantically) that the following formula is valid.

$$\mu \mathsf{x} \nu \mathsf{y} \varphi(\mathsf{x}, \mathsf{y}) \to \nu \mathsf{y} \mu \mathsf{x} \varphi(\mathsf{x}, \mathsf{y})$$

Let $\psi_1 := \nu \mathsf{x} \mu \mathsf{y} \overline{\varphi}(\mathsf{x}, \mathsf{y})$ and $\psi_2 := \nu \mathsf{z} \mu \mathsf{w} \varphi(\mathsf{z}, \mathsf{w})$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\mu \mathsf{y} \overline{\varphi}(\psi_1, \mathsf{y}), \mu \mathsf{w} \varphi(\mathsf{w}, \psi_2)}{\psi_1, \mu \mathsf{w} \varphi(\mathsf{w}, \psi_2)} \nu_{\mathsf{x}} \qquad \mu \mathsf{y} \overline{\varphi}(\psi_1, \mathsf{y}), \psi_2}{\vdots}}{\overline{\varphi}(\psi_1, \mu \mathsf{y} \overline{\varphi}(\psi_1, \mathsf{y})), \varphi(\mu \mathsf{w} \varphi(\mathsf{w}, \psi_2), \psi_2)} \mu_{\mathsf{y}}, \mu_{\mathsf{w}}}{\dfrac{\mu \mathsf{y} \overline{\varphi}(\psi_1, \mathsf{y}), \mu \mathsf{w} \varphi(\mathsf{z}, \psi_2)}{\mu \mathsf{y} \overline{\varphi}(\psi_1, y), \psi_2} \nu_{\mathsf{x}}}}{\psi_1, \psi_2} \nu_{\mathsf{z}}$$

It is not hard to check (semantically) that the following formula is valid.

$$\mu x \nu y \varphi(x, y) \to \nu y \mu x \varphi(x, y)$$

Let $\psi_1 := \nu x \mu y \overline{\varphi}(x, y)$ and $\psi_2 := \nu z \mu w \varphi(z, w)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mu y \overline{\varphi}(\psi_1, y), \mu w \varphi(w, \psi_2)}{\psi_1, \mu w \varphi(w, \psi_2)}\,\nu_x \qquad \mu y \overline{\varphi}(\psi_1, y), \psi_2}{\vdots}}{\overline{\varphi}(\psi_1, \mu y \overline{\varphi}(\psi_1, y)), \varphi(\mu w \varphi(w, \psi_2), \psi_2)}\,\mu_y, \mu_w}{\mu y \overline{\varphi}(\psi_1, y), \mu w \varphi(z, \psi_2)}\,\nu_z}{\mu y \overline{\varphi}(\psi_1, y), \psi_2}\,\nu_x}{\psi_1, \psi_2}$$

Question significance order? kind of traces?

Theorem (Niwinski, Walukiewicz 1996)

For every closed formula φ, there exists a tableau-proof for φ iff φ is valid.

Theorem (Niwinski, Walukiewicz 1996)

For every closed formula φ, there exists a tableau-proof for φ iff φ is valid.

Theorem (Jungteerapanich, 2009; Stirling, 2014)

*φ is valid if and only if there is a **regular/cyclic** proof (in the axioms and rules of* Fix*).*

Theorem (Niwinski, Walukiewicz 1996)

For every closed formula φ, there exists a tableau-proof for φ iff φ is valid.

Theorem (Jungteerapanich, 2009; Stirling, 2014)

*φ is valid if and only if there is a **regular/cyclic** proof (in the axioms and rules of* Fix*).*

Main challenge Finding regular representations and turn the infinitary proofs into cyclic proofs.

Build an automaton $\mathcal{A}$ accepting precisely the ill-founded proofs. By Rabin Basis Theorem $\mathcal{A}$ accepts a regular tree, i.e. a cyclic proof.

Disadvantage: Need the automaton

Build an automaton $\mathcal{A}$ accepting precisely the ill-founded proofs. By Rabin Basis Theorem $\mathcal{A}$ accepts a regular tree, i.e. a cyclic proof.

Disadvantage: Need the automaton

Direct 'Annotate' sequents with more information so that repetitions along paths can be identified. Use it to mark a frontier (finite initial subtree) that when viewed as a cyclic derivation tree witnesses the validity

- modal μ-calculus: Jungteerapanich, 2009; Stirling, 2014
- LTL: Brünnler and Lange 2008; Kokkinis and Studer 2016
- CTL*: A., Leigh & Turata 2023
- μFOL: A., Enqvist & Leigh 2022
- PDL (w.i.p)

- cyclic proofs for extension of μ-calculus with
  - converse modalities
  - nominals
  - counting

- cyclic proofs for extension of μ-calculus with
  - converse modalities
  - nominals
  - counting
- cyclic proofs of fragment guarded fixed point logics

- cyclic proofs for extension of μ-calculus with
  - converse modalities
  - nominals
  - counting
- cyclic proofs of fragment guarded fixed point logics
- cyclic proofs for PDL

- cyclic proofs for extension of μ-calculus with
  - converse modalities
  - nominals
  - counting
- cyclic proofs of fragment guarded fixed point logics
- cyclic proofs for PDL
- intuitionistic logic
- higher type logic
- higher order logic
- probabilistic logic

- cyclic proofs for extension of μ-calculus with
  - converse modalities
  - nominals
  - counting
- cyclic proofs of fragment guarded fixed point logics
- cyclic proofs for PDL
- intuitionistic logic
- higher type logic
- higher order logic
- probabilistic logic

What sort of questions?

- cyclic proofs for extension of μ-calculus with
  - converse modalities
  - nominals
  - counting
- cyclic proofs of fragment guarded fixed point logics
- cyclic proofs for PDL
- intuitionistic logic
- higher type logic
- higher order logic
- probabilistic logic

What sort of questions?

- designing sound (and complete) axiomatisation
- investigating metalogical properties such as decidability, consistency, interpolation, cut-elimination, ordinal analysis, applications in other areas such as databases