Expressiveness of Temporal Logics

François Laroussinie and Nicolas Markey

Lab. Specification et Verification
ENS Cachan & CNRS, France

July 31, 2006
The Origins of Temporal Logics

Aristotle, Diodorus Cronus: (∼ 400 B.C.)

*Paradox of the future contingents:*

There will be a sea battle tomorrow

⇒ need for a notion of time in logics.
The Origins of Temporal Logics

- John Duns Scotus (1266-1308): modalities;
- William of Ockham (1285-1347): three-valued logics;
- Gottfried Leibniz (1646-1716): formal logic, modal logic;
The Origins of Temporal Logics

- Gottlob Frege (1848-1925): first-order logic;
- Clarence Irving Lewis (1883-1964): modal logics;
- Saul Kripke (1940-): modal logics;
- Hans Kamp (1940-): temporal logics;
- Amir Pnueli (1941-): temporal logics in computer science.
Definition

The syntax of *propositional logics* is defined as

\[ \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \]

where \( p \) ranges over a (finite) set of atomic propositions AP.

The semantics is given by truth tables, e.g., for \( p \land q \):

\[
\begin{array}{c|c|c}
\land & \top & \bot \\
\hline
\top & \top & \bot \\
\bot & \bot & \bot \\
\end{array}
\]
Definition

The syntax of *propositional logics* is defined as

\[ \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \]

where \( p \) ranges over a (finite) set of atomic propositions \( \text{AP} \).

The semantics is given by truth tables, e.g., for \( p \land q \):

<table>
<thead>
<tr>
<th>( \land )</th>
<th>( \top )</th>
<th>( \bot )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \top )</td>
<td>( \top )</td>
<td>( \bot )</td>
</tr>
<tr>
<td>( \bot )</td>
<td>( \bot )</td>
<td>( \bot )</td>
</tr>
<tr>
<td>( \bot )</td>
<td>( \bot )</td>
<td>( \bot )</td>
</tr>
</tbody>
</table>

Examples

the sky is blue \( \land \) the grass is green
Propositional Logic, Modal Logic and Temporal Logic

Definition
The syntax of *propositional logics* is defined as

\[ \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \]

where \( p \) ranges over a (finite) set of atomic propositions \( \text{AP} \).

The semantics is given by truth tables, e.g., for \( p \land q \):

\[
\begin{array}{c|c|c}
\land & \top & \bot \\
\hline
\top & \top & \bot \\
\bot & \bot & \bot \\
\bot & \bot & \bot \\
\end{array}
\]

Examples

\[ \neg (\text{sky} \land \text{grass}) \equiv \neg \text{sky} \lor \neg \text{grass} \]
Propositional Logic, Modal Logic and Temporal Logic

Definition

*First-order logic* is an extension of propositional logic with (first-order) quantification:

\[ \varphi ::= \ p(x) \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \exists x \in X. \ \varphi \mid \forall x \in X. \ \varphi \]

where \( p \) ranges over a finite set of predicates, \( X \) are sets, and \( x \) is an item of those sets.
**Definition**

*First-order logic* is an extension of propositional logic with (first-order) quantification:

\[ \varphi ::= p(x) \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \exists x \in X. \varphi \mid \forall x \in X. \varphi \]

where \( p \) ranges over a finite set of predicates, \( X \) are sets, and \( x \) is an item of those sets.

**Examples**

\[ \exists x \in \{\text{sky, grass}\}. \ (\text{is\_green}(x) \lor \text{is\_blue}(x)) \]
Definition

*First-order logic* is an extension of propositional logic with (first-order) quantification:

\[ \varphi ::= p(x) \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \exists x \in X. \varphi \mid \forall x \in X. \varphi \]

where \( p \) ranges over a finite set of predicates, \( X \) are sets, and \( x \) is an item of those sets.

Examples

\[ \exists x \in \{\text{banana, orange}\}. \ (\text{is\_green}(x) \lor \text{is\_blue}(x)) \]
**Definition**

*First-order logic* is an extension of propositional logic with (first-order) quantification:

\[ \varphi ::= p(x) \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \exists x \in X. \varphi \mid \forall x \in X. \varphi \]

where \( p \) ranges over a finite set of predicates, \( X \) are sets, and \( x \) is an item of those sets.

**Examples**

\[ \neg [\exists x \in X. \text{is\_green}(x)] \equiv \forall x \in X. \neg \text{is\_green}(x) \]
Propositional Logic, Modal Logic and Temporal Logic

Definition

Modal logic is an extension of propositional logic with “modalities” for expressing that something is possible or necessary:

$$\varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \lozenge \varphi \mid \square \varphi$$

where $p$ ranges over AP.
Propositional Logic, Modal Logic and Temporal Logic

Definition

Modal logic is an extension of propositional logic with “modalities” for expressing that something is possible or necessary:

\[ \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \Diamond \varphi \mid \Box \varphi \]

where \( p \) ranges over AP.

Examples

\[ \Diamond (it\ rains \land\ the\ sun\ shines) \]
Definition

Modal logic is an extension of propositional logic with “modalities” for expressing that something is possible or necessary:

\[ \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \diamond \varphi \mid \Box \varphi \]

where \( p \) ranges over AP.

Examples

\[ \neg \diamond (\text{it rains}) \equiv \Box (\neg \text{it rains}) \]
Modal logic formulas are interpreted over Kripke structures:

**Definition**

A *Kripke structure* is a 3-tuple \( \langle W, R, \ell \rangle \) where

- \( W \) is a non-empty set of *worlds*,
- \( R \) is a binary relation over \( W \), called *accessibility relation* (often required to be total),
- \( \ell : W \to 2^{AP} \) gives the valuation of atomic propositions in each world.
Modal logic formulas are interpreted over Kripke structures:

**Definition**

A *Kripke structure* is a 3-tuple \( \langle W, R, \ell \rangle \) where

- \( W \) is a non-empty set of *worlds*,
- \( R \) is a binary relation over \( W \), called *accessibility relation* (often required to be total),
- \( \ell : W \rightarrow 2^{AP} \) gives the valuation of atomic propositions in each world.

**Example**

```
red, blue  green, blue  red
```

\[ \text{red, blue} \rightarrow \text{green, blue} \rightarrow \text{red} \]
Temporal logics are special cases of modal logics that refer to the succession of events along time:

- ♦ is generally denoted with $F$, and is read “eventually”;
- □ is generally written $G$, and read “globally” or “always”.

*Temporal logics* are special cases of modal logics that refer to the succession of events along time:
Temporal logics are special cases of modal logics that refer to the succession of events along time:

- $\Diamond$ is generally denoted with $F$, and is read “eventually”,
- $\Box$ is generally written $G$, and read “globally” or “always”.

Example

$\neg F(\text{it rains}) \equiv G(\neg \text{it rains})$
Temporal logics in computer science

Used for specifying/describing behaviors of reactive systems.
Temporal logics in computer science

Used for specifying/describing behaviors of reactive systems.

Example (simplified model of a lift)

```
third floor
- down?
- up?

second floor
- down?
- up?

first floor
- down?
- up?

- cabin
```
Temporal logics in computer science

Used for specifying/describing behaviors of reactive systems.

Example (simplified model of a lift)

- Cabin
  - Third floor
  - Second floor
  - First floor

- Button
  - idle
  - go 1st floor
  - go 2nd floor
  - go 3rd floor

States:
- down?
- up?
- served
- request

Transitions:
- idle: go 1st floor
- 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
- idle: go 1st floor
- go 1st floor: serve
- serve: request
- request: idle
Temporal logics in computer science

Used for specifying/describing behaviors of reactive systems.

Example (simplified model of a lift)
Temporal logics in computer science

Used for specifying/describing behaviors of reactive systems.

Example (simplified model of a lift)

- Cabin
  - First floor
    - Down?
    - Up?
  - Second floor
    - Down?
    - Up?
  - Third floor
    - Down?
    - Up?

- Button
  - Call
    - Open?
  - Idle
    - Press!

- Doors
  - Open
    - Closed
  - Closed
    - Open?
Temporal logics in computer science

Used for specifying/describing behaviors of reactive systems.

Example (simplified model of a lift)

1. **Cabin**:
   - First floor
   - Second floor
   - Third floor
   - **Buttons**
     - Down
     - Up

2. **Button at Floor**:
   - Idle
   - Call
   - Press
   - Open

3. **Doors**:
   - Open
   - Closed
   - **Buttons**
     - Open
     - Close

4. **Controller**:
   - Up

The diagram illustrates the flow of operations and transitions between states in a lift system, using temporal logics to specify the behaviors.
Temporal logics in computer science

Used for specifying/describing behaviors of reactive systems.

Example (simplified model of a lift)

*Is it possible that the door is open at the first floor while the cabin is at the second floor?*
Temporal logics in computer science

Used for specifying/describing behaviors of reactive systems.

Example (simplified model of a lift)

Is it possible that the door is open at the first floor while the cabin is at the second floor?

Is any request eventually served?
Temporal logics in computer science

Used for specifying/describing behaviors of reactive systems.

Example (simplified model of a lift)

*Is it possible that the door is open at the first floor while the cabin is at the second floor?*

*Is any request eventually served?*

*Are the doors always closed when the cabin moves?*
Temporal logics in computer science

Used for specifying/describing behaviors of reactive systems.

Example (simplified model of a lift)

*Is it possible that the door is open at the first floor while the cabin is at the second floor?*

*Is any request eventually served?*

*Are the doors always closed when the cabin moves?*

Model-checking: automatic verification that the model satisfies those properties.
Linear-time and branching-time

Two different frameworks:

- **Linear-time framework:**
  - Deals with one single execution at a time;

- **Branching-time framework:**
  - Deals with the computation tree of the system;
  - Quantification on the possible evolutions of the system.

Example:
- Any request is eventually granted.
- It is always possible to go to the first floor.
Linear-time and branching-time

Two different frameworks:

- linear-time framework:
  - deals with one single execution at a time;

Example: Any request is eventually granted.

branching-time framework:
  - deals with the computation tree of the system;
  - quantification on the possible evolutions of the system.

Example: It is always possible to go to the first floor.
Linear-time and branching-time

Two different frameworks:

- linear-time framework:
  \[ \rightsquigarrow \] deals with one single execution at a time;

**Example**

*Any request is eventually granted.*
Linear-time and branching-time

Two different frameworks:

- linear-time framework:
  - deals with one single execution at a time;

Example

*Any request is eventually granted.*

- branching-time framework:
  - deals with the *computation tree* of the system;
  - quantification on the possible evolutions of the system.
Linear-time and branching-time

Two different frameworks:

- linear-time framework:
  - deals with one single execution at a time;

  **Example**
  
  *Any request is eventually granted.*

- branching-time framework:
  - deals with the *computation tree* of the system;
  - quantification on the possible evolutions of the system.

  **Example**
  
  *It is always possible to go to the first floor.*
Timed temporal logics

Timed logics include a quantitative notion of time:

Example
If the doors are not blocked, it is always possible to reach the first floor in at most 2 minutes. Any request is eventually served in at most 1 minute. Requires explicit timing constraints in the model (e.g. timed automata).
Timed temporal logics

Timed logics include a quantitative notion of time:

**Example**

*If the doors are not blocked, it is always possible to reach the first floor in at most 2 minutes.*

*Any request is eventually served in at most 1 minute.*
Timed temporal logics

Timed logics include a quantitative notion of time:

Example

*If the doors are not blocked, it is always possible to reach the first floor in at most 2 minutes.*

Any request is eventually served *in at most 1 minute.*

Requires explicit timing constraints in the model (e.g. timed automata).
Outline of the course

- **Monday**: linear-time temporal logics:
  - definitions;
  - relations with first-order logic;
- **Tuesday**: linear-time temporal logics:
  - relations with Büchi automata;
  - expressiveness hierarchies;
- **Wednesday**: branching-time temporal logics:
  - definition of CTL;
  - comparison with LTL, definition of CTL*;
- **Thursday**: branching-time temporal logics:
  - fragments of CTL*;
  - relations with automata theory;
- **Friday**: timed temporal logics:
  - definitions;
  - some results in the linear-time case.
Outline of today’s lecture

1 Introduction

2 The notion of expressiveness
   - Distinguishing power
   - Expressive power
   - Succinctness

3 Linear-time temporal logics
   - Definitions, basic formulas
   - LTL, PLTL and FO
   - Normal forms for LTL+Past
Outline of today’s lecture

1 Introduction

2 The notion of expressiveness
   - Distinguishing power
   - Expressive power
   - Succinctness

3 Linear-time temporal logics
   - Definitions, basic formulas
   - LTL, PLTL and FO
   - Normal forms for LTL+Past
Distinguishing power

**Definition**

Two structures \( s \) and \( s' \) of \( S \) are *equivalent* w.r.t. a logic \( L \) (written \( s \equiv_L s' \)) if they satisfy the same set of formulas of \( L \), i.e., if

\[
\{ \varphi \in L \mid s \models \varphi \} = \{ \varphi \in L \mid s' \models \varphi \}.
\]
Distinguishing power

Definition

Two structures $s$ and $s'$ of $S$ are equivalent w.r.t. a logic $\mathcal{L}$ (written $s \equiv_{\mathcal{L}} s'$) if they satisfy the same set of formulas of $\mathcal{L}$, i.e., if

$$\{ \varphi \in \mathcal{L} \mid s \models \varphi \} = \{ \varphi \in \mathcal{L} \mid s' \models \varphi \}.$$

Example

The following structures are equivalent w.r.t. a logic that can only express eventuality:

- Structure 1:
  - green
  - blue
  - red

- Structure 2:
  - green
  - red
  - blue

Diagram:

```
  green -> blue -> red

  green -> red -> blue
```
## Distinguishing power

**Definition**

A logic $L$ has *at least as much distinguishing power* as a logic $L'$ over a set of structures $S$ (denoted by $L \geq_d L'$) whenever

$$\forall s, s' \in S. \text{ if } s \equiv_L s' \text{ then } s \equiv_{L'} s'.$$
Distinguishing power

Definition
A logic $\mathcal{L}$ has *at least as much distinguishing power* as a logic $\mathcal{L}'$ over a set of structures $S$ (denoted by $\mathcal{L} \geq_d \mathcal{L}'$) whenever
\[
\forall s, s' \in S. \text{ if } s \equiv_\mathcal{L} s' \text{ then } s \equiv_{\mathcal{L}'} s'.
\]

Definition
Two logics $\mathcal{L}$ et $\mathcal{L}'$ have the same *distinguishing power* over a set of structures $S$ if $\mathcal{L} \geq_d \mathcal{L}'$ and $\mathcal{L}' \geq_d \mathcal{L}$. This is denoted by $\mathcal{L} \equiv_d \mathcal{L}'$. 
Distinguishing power

**Definition**

A logic \( \mathcal{L} \) has *at least as much distinguishing power* as a logic \( \mathcal{L}' \) over a set of structures \( S \) (denoted by \( \mathcal{L} \geq_d \mathcal{L}' \)) whenever

\[
\forall s, s' \in S. \quad \text{if } s \equiv_\mathcal{L} s' \text{ then } s \equiv_\mathcal{L}' s'.
\]

**Definition**

Two logics \( \mathcal{L} \) et \( \mathcal{L}' \) have the same *distinguishing power* over a set of structures \( S \) if \( \mathcal{L} \geq_d \mathcal{L}' \) and \( \mathcal{L}' \geq_d \mathcal{L} \). This is denoted by \( \mathcal{L} \equiv_d \mathcal{L}' \).

**Definition**

A logic \( \mathcal{L} \) has *strictly more distinguishing power* than a logic \( \mathcal{L}' \) over a set of structures \( S \) if \( \mathcal{L} \geq_d \mathcal{L}' \) and \( \mathcal{L} \not\equiv_d \mathcal{L}' \). This is denoted by \( \mathcal{L} >_d \mathcal{L}' \).
Expressive power

**Definition**

Two formulas $\varphi$ and $\psi$ are *equivalent* over a set $S$ of structures if

$$\forall s \in S. \ (s \models \varphi \iff s \models \psi).$$
# Expressive power

## Definition

Two formulas $\varphi$ and $\psi$ are **equivalent** over a set $S$ of structures if

$$\forall s \in S. (s \models \varphi \iff s \models \psi).$$

## Examples

Over the set of Kripke structures,

$$F_{\text{green}} \land F_{\text{red}} \not\equiv F(\text{green} \land \text{red}).$$
Expressive power

Definition
Two formulas $\varphi$ and $\psi$ are equivalent over a set $S$ of structures if

$$\forall s \in S. (s \models \varphi \iff s \models \psi).$$

Examples
Over the set of Kripke structures where green is always true,

$$F_{green} \land F_{red} \equiv F(green \land red).$$
Expressive power

Definition

A logic $\mathcal{L}$ is at least as expressive as a logic $\mathcal{L}'$ over a set of structures $S$ if for any formula in $\mathcal{L}'$, there exists a formula in $\mathcal{L}$ that is equivalent over $S$. This is written $\mathcal{L} \geq_e \mathcal{L}'$. 
Expressive power

Definition
A logic $\mathcal{L}$ is *at least as expressive* as a logic $\mathcal{L}'$ over a set of structures $\mathcal{S}$ if for any formula in $\mathcal{L}'$, there exists a formula in $\mathcal{L}$ that is equivalent over $\mathcal{S}$. This is written $\mathcal{L} \geq_e \mathcal{L}'$.

Definition
Two logics are *equally expressive* over $\mathcal{S}$ (denoted by $\mathcal{L} \equiv_e \mathcal{L}'$) if they are at least as expressive as each other.
Expressive power

Definition
A logic $\mathcal{L}$ is at least as expressive as a logic $\mathcal{L}'$ over a set of structures $S$ if for any formula in $\mathcal{L}'$, there exists a formula in $\mathcal{L}$ that is equivalent over $S$. This is written $\mathcal{L} \geq_e \mathcal{L}'$.

Definition
Two logics are equally expressive over $S$ (denoted by $\mathcal{L} \equiv_e \mathcal{L}'$) if they are at least as expressive as each other.

Definition
A logic $\mathcal{L}$ is strictly more expressive than a logic $\mathcal{L}'$ over $S$ (denoted by $\mathcal{L} >_e \mathcal{L}'$) if $\mathcal{L} \geq_e \mathcal{L}'$ and $\mathcal{L} \not\equiv_e \mathcal{L}'$. 
Distinguishing power vs expressive power

**Theorem**

*Expressive power is finer than distinguishing power.*
Distinguishing power vs expressive power

**Theorem**

*Expressive power is finer than distinguishing power.*

**Proof.**

- $L$ has the same expressive power as $L'$:

$$\forall \varphi' \in L'. \exists \varphi \in L. \forall s \in S. \ (s \models \varphi' \iff s \models \varphi)$$

and vice-versa.
Distinguishing power vs expressive power

Theorem

Expressive power is finer than distinguishing power.

Proof.

- $\mathcal{L}$ has the same expressive power as $\mathcal{L}'$:

$$\forall \varphi' \in \mathcal{L}'. \exists \varphi \in \mathcal{L}. \forall s \in S. \ (s \models \varphi' \iff s \models \varphi)$$

and vice-versa.

- $\mathcal{L}$ has the same distinguishing power as $\mathcal{L}'$:

$$\forall s, s' \in S. \forall \varphi' \in \mathcal{L}'. \exists \varphi \in \mathcal{L}. \ (s \models \varphi' \land s' \not\models \varphi') \Rightarrow (s \models \varphi \land s' \not\models \varphi)$$

and vice-versa.
Succinctness

When two logics are equally expressive, the sizes of equivalent formulas is another important criterion:

**Definition**

The *size* of a formula is the number of nodes of the tree that represents that formula.
Succinctness

When two logics are equally expressive, the sizes of equivalent formulas is another important criterion:

**Definition**

The *size* of a formula is the number of nodes of the tree that represents that formula.

**Example**

\[ \varphi = \text{green} \land F \text{red} \]

\[ |\varphi| = 4 \]
Succinctness

When two logics are equally expressive, the sizes of equivalent formulas is another important criterion:

**Definition**

The *size* of a formula is the number of nodes of the tree that represents that formula.

When two logics are equally expressive, we define the notion of succinctness:

**Definition**

A logic $\mathcal{L}$ is *more succinct* than a logic $\mathcal{L}'$ if, for any formula $\varphi \in \mathcal{L}$ and any equivalent formula $\varphi' \in \mathcal{L}'$, we have

$$|\varphi| \leq |\varphi'|$$
Outline of today’s lecture

1 Introduction

2 The notion of expressiveness
   - Distinguishing power
   - Expressive power
   - Succinctness

3 Linear-time temporal logics
   - Definitions, basic formulas
   - LTL, PLTL and FO
   - Normal forms for LTL+Past
Outline of today’s lecture

1. Introduction

2. The notion of expressiveness
   - Distinguishing power
   - Expressive power
   - Succinctness

3. Linear-time temporal logics
   - Definitions, basic formulas
   - LTL, PLTL and FO
   - Normal forms for LTL+Past
The linear-time framework

Linear-time temporal logic framework:
Express properties over each single execution of the system
The linear-time framework

Linear-time temporal logic framework:
Express properties over each single execution of the system

Examples

first floor
open₁
 go 3rd floor

first floor
closed₁
go 3rd floor

second floor
go 3rd floor

third floor
closed₃
go 3rd floor

third floor
open₃

\( G(\text{go 3rd floor} \Rightarrow F(\text{third floor})) \)
The linear-time framework

Linear-time temporal logic framework:
Express properties over each single execution of the system

Examples

\[ G(\text{open}_3 \Rightarrow \text{third floor}) \]
The linear-time framework

Linear-time temporal logic framework:
Express properties over each single execution of the system

Examples

\[(G F \text{open}_3) \Rightarrow (G F(\text{call}_3 \lor \text{go 3rd floor}))\]
Linear structures

Definition

A (labelled) linear structure over a set $AP$ of atomic propositions is a triple $S = \langle T, <, \ell \rangle$ where

- $T$ is an infinite set,
- $<$ is a linear order on $T$ s.t. $T$ has a minimal element, and
- $\ell: T \rightarrow 2^AP$ is a labelling function.
Linear structures

Definition

A **(labelled) linear structure** over a set $\mathit{AP}$ of atomic propositions is a triple $\mathcal{S} = \langle T, <, \ell \rangle$ where

- $T$ is an infinite set,
- $<$ is a linear order on $T$ s.t. $T$ has a minimal element, and
- $\ell : T \rightarrow 2^\mathit{AP}$ is a labelling function.

Examples

With $T = \mathbb{Z}^+$:

- 0: first floor open\textsubscript{1} go 3rd floor
- 1: first floor closed\textsubscript{1} go 3rd floor
- 2: second floor go 3rd floor
A *(labelled) linear structure* over a set \( AP \) of atomic propositions is a triple \( S = \langle T, <, \ell \rangle \) where

- \( T \) is an infinite set,
- \( < \) is a linear order on \( T \) s.t. \( T \) has a minimal element, and
- \( \ell : T \to 2^{AP} \) is a labelling function.

**Examples**

With \( T = \mathbb{R}^+ \):

- First floor
- Open
- Go 3rd floor

- First floor
- Closed
- Go 3rd floor

- Second floor
- Go 3rd floor
A \textit{(labelled) linear structure} over a set $\text{AP}$ of atomic propositions is a triple $\mathcal{S} = \langle T, <, \ell \rangle$ where

- $T$ is an infinite set,
- $<$ is a linear order on $T$ s.t. $T$ has a minimal element, and
- $\ell : T \rightarrow 2^{\text{AP}}$ is a labelling function.

A \textit{pointed linear structure} is a pair $m = \langle S, t \rangle$ where

- $S = \langle T, <, \ell \rangle$ is a linear structure,
- $t$ is an element of $T$. 
Linear structures

**Definition**

A *(labelled) linear structure* over a set $\text{AP}$ of atomic propositions is a triple $S = \langle T, <, \ell \rangle$ where

- $T$ is an infinite set,
- $<$ is a linear order on $T$ s.t. $T$ has a minimal element, and
- $\ell : T \to 2^{\text{AP}}$ is a labelling function.

**Definition**

A *(pointed linear structure)* is a pair $m = \langle S, t \rangle$ where

- $S = \langle T, <, \ell \rangle$ is a linear structure,
- $t$ is an element of $T$.

We write $\text{Lin}(T)$ for the set of pointed linear structures on $T$. $\text{Lin}^0(T)$ for the subset of *initially* pointed structures.
Linear-time modalities

Historically, two modalities:

\[ \mathbf{F} \varphi \text{ (or } \diamond \varphi \text{): } \langle S, t \rangle \models \mathbf{F} \varphi \iff \exists u > t. \langle S, u \rangle \models \varphi \]

("eventually" \( \varphi \))
Linear-time modalities

Historically, two modalities:

\[ \mathbf{F} \varphi \ (\text{or } \Diamond \varphi) : \langle S, t \rangle \models \mathbf{F} \varphi \iff \exists u > t. \langle S, u \rangle \models \varphi \]
("eventually" \( \varphi \))

\[ \mathbf{G} \varphi \ (\text{or } \Box \varphi) : \langle S, t \rangle \models \mathbf{G} \varphi \iff \forall u > t. \langle S, u \rangle \models \varphi \]
("always" \( \varphi \))
Linear-time modalities

Historically, two modalities:

\[ \mathbf{F} \varphi \text{ (or } \Diamond \varphi \text{)} : \langle S, t \rangle \models \mathbf{F} \varphi \iff \exists u > t. \langle S, u \rangle \models \varphi \]

("eventually" \( \varphi \))

\[ \mathbf{G} \varphi \text{ (or } \Box \varphi \text{)} : \langle S, t \rangle \models \mathbf{G} \varphi \iff \forall u > t. \langle S, u \rangle \models \varphi \]

("always" \( \varphi \))

Examples

Liveness properties: \( \mathbf{F} \text{ open}_i \)
Linear-time modalities

Historically, two modalities:

\[ F \varphi \text{ (or } \diamond \varphi) : \langle S, t \rangle \models F \varphi \iff \exists u > t. \langle S, u \rangle \models \varphi \]

(“eventually” \( \varphi \))

\[ G \varphi \text{ (or } \Box \varphi) : \langle S, t \rangle \models G \varphi \iff \forall u > t. \langle S, u \rangle \models \varphi \]

(“always” \( \varphi \))

Examples

Liveness properties: \( F \text{ open}_i \)

Safety properties: \( G(\text{open}_i \Rightarrow i\text{-th floor}) \)
Linear-time modalities

Historically, two modalities:

\[ F \varphi \] (or \( \Diamond \varphi \)) : \( \langle S, t \rangle \models F \varphi \iff \exists u > t. \langle S, u \rangle \models \varphi \)
(“eventually” \( \varphi \))

\[ G \varphi \] (or \( \Box \varphi \)) : \( \langle S, t \rangle \models G \varphi \iff \forall u > t. \langle S, u \rangle \models \varphi \)
(“always” \( \varphi \))

Examples

Duality:

\[ \mathbf{F} \varphi \equiv \neg \mathbf{G} \neg \varphi \]
Linear-time modalities

Historically, two modalities:

\( \mathbf{F} \varphi \) (or \( \diamond \varphi \)) : \( \langle S, t \rangle \models \mathbf{F} \varphi \iff \exists u > t. \langle S, u \rangle \models \varphi \)

(“eventually” \( \varphi \))

\( \mathbf{G} \varphi \) (or \( \Box \varphi \)) : \( \langle S, t \rangle \models \mathbf{G} \varphi \iff \forall u > t. \langle S, u \rangle \models \varphi \)

(“always” \( \varphi \))

Examples

Distributivity:

\[ \mathbf{F} \varphi \lor \mathbf{F} \psi \equiv \mathbf{F}(\varphi \lor \psi) \]

\[ \mathbf{F} \varphi \land \mathbf{F} \psi \not\equiv \mathbf{F}(\varphi \land \psi) \]
Linear-time modalities

Historically, two modalities:

F \varphi \ (or \ \diamond \varphi) : \langle S, t \rangle \models F \varphi \iff \exists u > t. \langle S, u \rangle \models \varphi
(“eventually” \ \varphi)

G \varphi \ (or \ \Box \varphi) : \langle S, t \rangle \models G \varphi \iff \forall u > t. \langle S, u \rangle \models \varphi
(“always” \ \varphi)

Examples

Distributivity:

G \varphi \lor G \psi \not\equiv G(\varphi \lor \psi)

G \varphi \land G \psi \equiv G(\varphi \land \psi)
Linear-time modalities

Historically, two modalities:

- **$F \varphi$ (or $\diamond \varphi$):** $\langle S, t \rangle \models F \varphi \iff \exists u > t. \langle S, u \rangle \models \varphi$
  ("eventually" $\varphi$)

- **$G \varphi$ (or $\Box \varphi$):** $\langle S, t \rangle \models G \varphi \iff \forall u > t. \langle S, u \rangle \models \varphi$
  ("always" $\varphi$)

Examples

Fairness properties:

- **$G F \varphi$**
  ("infinitely often" $\varphi$)
Linear-time modalities

Historically, two modalities:

\[ F \varphi \ (\text{or} \ \lozenge \varphi) : \langle S, t \rangle \models F \varphi \iff \exists u > t. \langle S, u \rangle \models \varphi \]

(“eventually” \( \varphi \))

\[ G \varphi \ (\text{or} \ \square \varphi) : \langle S, t \rangle \models G \varphi \iff \forall u > t. \langle S, u \rangle \models \varphi \]

(“always” \( \varphi \))

Examples

Fairness properties:

\[ G F \varphi \ \overset{\text{def}}{=} \ \infty \varphi \]

(“infinitely often” \( \varphi \))
Linear-time modalities

Historically, two modalities:

\[ F \varphi \text{ (or } \Diamond \varphi \text{)} : \langle S, t \rangle \models F \varphi \iff \exists u > t. \langle S, u \rangle \models \varphi \]
(“eventually” \( \varphi \))

\[ G \varphi \text{ (or } \Box \varphi \text{)} : \langle S, t \rangle \models G \varphi \iff \forall u > t. \langle S, u \rangle \models \varphi \]
(“always” \( \varphi \))

Examples

Non-strict modalities:

\[ \tilde{F} \varphi \overset{\text{def}}{=} \varphi \lor F \varphi \]
\[ \tilde{G} \varphi \overset{\text{def}}{=} \varphi \land G \varphi \]
Linear-time modalities

Past-time counterparts:

\( F^{-1} \varphi \) (or \( \Diamond \varphi \)) : \( \langle S, t \rangle \models F^{-1} \varphi \iff \exists u < t. \langle S, u \rangle \models \varphi \)

(“sometimes in the past” \( \varphi \))
Linear-time modalities

Past-time counterparts:

\[ \mathbf{F}^{-1} \varphi \text{ (or } \Diamond \varphi \text{) : } \langle S, t \rangle \models \mathbf{F}^{-1} \varphi \iff \exists u < t. \langle S, u \rangle \models \varphi \]

(“sometimes in the past” \( \varphi \))

\[ \mathbf{G}^{-1} \varphi \text{ (or } \Box \varphi \text{) : } \langle S, t \rangle \models \mathbf{G}^{-1} \varphi \iff \forall u < t. \langle S, u \rangle \models \varphi \]

(“always in the past” \( \varphi \))
Linear-time modalities

Past-time counterparts:

\[ F^{-1} \varphi \quad \text{(or } \Diamond \varphi) : \quad \langle S, t \rangle \models F^{-1} \varphi \iff \exists u < t. \langle S, u \rangle \models \varphi \]

(“sometimes in the past” \( \varphi \))

\[ G^{-1} \varphi \quad \text{(or } \Box \varphi) : \quad \langle S, t \rangle \models G^{-1} \varphi \iff \forall u < t. \langle S, u \rangle \models \varphi \]

(“always in the past” \( \varphi \))

Examples

Duality: \[ F^{-1} \varphi \equiv \neg G^{-1} \neg \varphi \]
Linear-time modalities

Past-time counterparts:

\[ F^{-1} \varphi \text{ (or } \Diamond \varphi) : \langle S, t \rangle \models F^{-1} \varphi \quad \iff \quad \exists u < t. \langle S, u \rangle \models \varphi \]

(“sometimes in the past” \( \varphi \))

\[ G^{-1} \varphi \text{ (or } \Box \varphi) : \langle S, t \rangle \models G^{-1} \varphi \quad \iff \quad \forall u < t. \langle S, u \rangle \models \varphi \]

(“always in the past” \( \varphi \))

Examples

Duality: \[ F^{-1} \varphi \equiv \neg G^{-1} \neg \varphi \]

Precedence properties: \[ G(\varphi \Rightarrow F^{-1} \psi) \]
Linear-time modalities

Past-time counterparts:

\[ \mathcal{F}^{-1} \varphi \ (\text{or } \diamond \varphi) : \langle S, t \rangle \models \mathcal{F}^{-1} \varphi \iff \exists u < t. \langle S, u \rangle \models \varphi \]

(“sometimes in the past” \( \varphi \))

\[ \mathcal{G}^{-1} \varphi \ (\text{or } \square \varphi) : \langle S, t \rangle \models \mathcal{G}^{-1} \varphi \iff \forall u < t. \langle S, u \rangle \models \varphi \]

(“always in the past” \( \varphi \))

Examples

Non-strict versions:

\[ \tilde{\mathcal{F}}^{-1} \varphi \ \overset{\text{def}}{=} \varphi \lor \mathcal{F}^{-1} \varphi \]

\[ \tilde{\mathcal{G}}^{-1} \varphi \ \overset{\text{def}}{=} \varphi \land \mathcal{G}^{-1} \varphi \]
Linear-time modalities

Past-time counterparts:

\[ \mathbf{F}^{-1} \varphi \ (\text{or} \ \Diamond \varphi) : \langle S, t \rangle \models \mathbf{F}^{-1} \varphi \iff \exists u < t. \langle S, u \rangle \models \varphi \]

(“sometimes in the past” \( \varphi \))

\[ \mathbf{G}^{-1} \varphi \ (\text{or} \ \Box \varphi) : \langle S, t \rangle \models \mathbf{G}^{-1} \varphi \iff \forall u < t. \langle S, u \rangle \models \varphi \]

(“always in the past” \( \varphi \))

Examples

\[ \mathbf{G}^{-1} \mathbf{F}^{-1} \varphi \equiv \bot \quad \text{except at origin} \]

\[ \mathbf{F}^{-1} \mathbf{G}^{-1} \varphi \equiv \top \quad \text{except at origin} \]
Linear-time modalities

Past-time counterparts:

\[ F^{-1} \varphi \text{ (or } \lozenge \varphi \text{) : } \langle S, t \rangle \models F^{-1} \varphi \iff \exists u < t. \langle S, u \rangle \models \varphi \]

(“sometimes in the past” \( \varphi \))

\[ G^{-1} \varphi \text{ (or } \blacksquare \varphi \text{) : } \langle S, t \rangle \models G^{-1} \varphi \iff \forall u < t. \langle S, u \rangle \models \varphi \]

(“always in the past” \( \varphi \))

Examples

“Initially”:

\[ \lnot G^{-1} \lnot F^{-1} \varphi \equiv \lnot F^{-1} \lnot G^{-1} \varphi \]
Linear-time modalities

Past-time counterparts:

\[ F^{-1} \varphi \text{ (or } \Diamond \varphi) \colon \langle S, t \rangle \models F^{-1} \varphi \iff \exists u < t. \langle S, u \rangle \models \varphi \]

(“sometimes in the past” \( \varphi \))

\[ G^{-1} \varphi \text{ (or } \Box \varphi) : \langle S, t \rangle \models G^{-1} \varphi \iff \forall u < t. \langle S, u \rangle \models \varphi \]

(“always in the past” \( \varphi \))

Examples

“Initially”:

\[ \tilde{G}^{-1} \tilde{F}^{-1} \varphi \equiv \tilde{F}^{-1} \tilde{G}^{-1} \varphi \overset{\text{def}}{=} I \varphi \]
Linear-time modalities

Past-time counterparts:

\( \mathbf{F}^{-1} \varphi \) (or \( \Diamond \varphi \)): \( \langle S, t \rangle \models \mathbf{F}^{-1} \varphi \iff \exists u < t. \langle S, u \rangle \models \varphi \)

(“sometimes in the past” \( \varphi \))

\( \mathbf{G}^{-1} \varphi \) (or \( \Box \varphi \)): \( \langle S, t \rangle \models \mathbf{G}^{-1} \varphi \iff \forall u < t. \langle S, u \rangle \models \varphi \)

(“always in the past” \( \varphi \))

Examples

“Initially”: \( \tilde{\mathbf{G}}^{-1} \tilde{\mathbf{F}}^{-1} \varphi \equiv \tilde{\mathbf{F}}^{-1} \tilde{\mathbf{G}}^{-1} \varphi \overset{\text{def}}{=} \mathbf{I} \varphi \)

“Until”: \( \mathbf{F} (\psi \land \mathbf{G}^{-1} \varphi) \)
“Until”?

Example

\[
green \text{ “until” } red \equiv F(red \land G^{-1} green)
\]
“Until”?

Example

\[ F(\text{red} \land \text{red “until” blue}) \not\equiv F(\text{red} \land F(\text{blue} \land G^{-1}\text{red})) \]
“Until”?  

Example

\[ F(\text{red} \land \text{red “until” blue}) \not\equiv F(\text{red} \land F(\text{blue} \land G^{-1}\text{red})) \]

Theorem (Kamp, 1968)

“Until” cannot be expressed using only \( F \), \( G \), \( F^{-1} \), and \( G^{-1} \).
“Until”?

Theorem (Kamp, 1968)

“Until” cannot be expressed using only \( F, G, F^{-1}, \) and \( G^{-1} \).
Theorem (Kamp, 1968)

“Until” cannot be expressed using only $F$, $G$, $F^{-1}$, and $G^{-1}$.

Proof (sketch).

Consider the following linear structure:

$A_n$
“Until”? 

Theorem (Kamp, 1968)

“Until” cannot be expressed using only $F$, $G$, $F^{-1}$, and $G^{-1}$.

Proof (sketch).

Consider the following linear structure:

Lemma

For any $n \in \mathbb{Z}^+$, formula

$$F(\text{blue } \land \text{blue “until” red})$$

holds along $A_n$ on a bounded subset of $\mathbb{R}^+$ containing $n$. 
“Until”?

**Theorem (Kamp, 1968)**

“*Until*” cannot be expressed using only $F$, $G$, $F^{-1}$, and $G^{-1}$.

**Proof (sketch).**

Consider the following linear structure:

![Linear structure diagram]

**Lemma**

Let $n \in \mathbb{Z}^+$, and $\varphi$ be a formula built on $F$, $G$, $F^{-1}$, and $G^{-1}$. Let $t \geq |\varphi|$ and $u \geq |\varphi|$ labelled with the same atomic propositions. Then

$$\langle A_n, t \rangle \models \varphi \iff \langle A_n, u \rangle \models \varphi$$
Lemma

Let \( n \in \mathbb{Z}^+ \), and \( \varphi \) be a formula built on \( F \), \( G \), \( F^{-1} \), and \( G^{-1} \). Let \( t \geq |\varphi| \) and \( u \geq |\varphi| \) labelled with the same atomic propositions. Then

\[
\langle A_n, t \rangle \models \varphi \iff \langle A_n, u \rangle \models \varphi
\]

Proof. By induction on the structure of the formula:
Lemma

Let \( n \in \mathbb{Z}^+ \), and \( \varphi \) be a formula built on \( F, G, F^{-1}, \) and \( G^{-1} \). Let \( t \geq |\varphi| \) and \( u \geq |\varphi| \) labelled with the same atomic propositions. Then

\[
\langle A_n, t \rangle \models \varphi \iff \langle A_n, u \rangle \models \varphi
\]

Proof. By induction on the structure of the formula:

- obvious for atomic propositions,
- straightforward for boolean combinators,
Lemma

Let \( n \in \mathbb{Z}^+ \), and \( \varphi \) be a formula built on \( F \), \( G \), \( F^{-1} \), and \( G^{-1} \). Let \( t \geq |\varphi| \) and \( u \geq |\varphi| \) labelled with the same atomic propositions. Then

\[
\langle A_n, t \rangle \models \varphi \iff \langle A_n, u \rangle \models \varphi
\]

Proof. By induction on the structure of the formula:

- if \( \varphi = F \psi \), then

\[
\langle A_n, t \rangle \models \varphi \Rightarrow \langle A_n, t' \rangle \models \psi \quad \text{for some } t' \geq t \geq |\psi| \\
\Rightarrow \langle A_n, u' \rangle \models \psi \quad \text{for any } u' \geq |\psi| \text{ labeled as } t' \\
\Rightarrow \langle A_n, u \rangle \models \varphi.
\]
“Until”? 

Now, if $F(\text{blue} \land \text{blue “until” red})$ can be expressed as a formula $\varphi$ built on $F$, $G$, $F^{-1}$, and $G^{-1}$, let $n = |\varphi|$. Then

- the set of positions along $A_n$ where $\varphi$ holds is bounded and contains $n$;
- since $\varphi$ holds at position $n$ along $A_n$, it also holds at any future position that is labeled by the same atomic propositions.

This is a contradiction.
Until and Since

\[ \varphi U \psi : \langle S, t \rangle \models \varphi U \psi \iff \exists u > t. (\langle S, u \rangle \models \psi \text{ and } \forall v > t. (v < u \Rightarrow \langle S, v \rangle \models \varphi)) \]
Until and Since

\[ \varphi \mathbf{U} \psi : \langle S, t \rangle \models \varphi \mathbf{U} \psi \iff \exists u > t. (\langle S, u \rangle \models \psi \text{ and } \forall v > t. (v < u \Rightarrow \langle S, v \rangle \models \varphi)) \]

(\varphi \ “until” \ \psi)

\[ \varphi \mathbf{S} \psi : \langle S, t \rangle \models \varphi \mathbf{S} \psi \iff \exists u < t. (\langle S, u \rangle \models \psi \text{ and } \forall v < t. (v > u \Rightarrow \langle S, v \rangle \models \varphi)) \]

(\varphi \ “since” \ \psi)
Until and Since

\( \varphi \mathbf{U} \psi : \langle S, t \rangle \models \varphi \mathbf{U} \psi \iff \exists u > t. (\langle S, u \rangle \models \psi \land \forall v > t. (v < u \Rightarrow \langle S, v \rangle \models \varphi) ) \)  

(\( \varphi \) “until” \( \psi \))

\( \varphi \mathbf{S} \psi : \langle S, t \rangle \models \varphi \mathbf{S} \psi \iff \exists u < t. (\langle S, u \rangle \models \psi \land \forall v < t. (v > u \Rightarrow \langle S, v \rangle \models \varphi) ) \)  

(\( \varphi \) “since” \( \psi \))

Examples

Equivalences:

\( \mathbf{F} \varphi \equiv \top \mathbf{U} \varphi \)

\( \mathbf{F}^{-1} \varphi \equiv \top \mathbf{S} \varphi \)
Until and Since

\[ \varphi \mathbf{U} \psi : \langle S, t \rangle \models \varphi \mathbf{U} \psi \iff \exists u > t. \ (\langle S, u \rangle \models \psi \text{ and } \forall v > t. \ (v < u \Rightarrow \langle S, v \rangle \models \varphi)) \]

\[ \varphi \mathbf{S} \psi : \langle S, t \rangle \models \varphi \mathbf{S} \psi \iff \exists u < t. \ (\langle S, u \rangle \models \psi \text{ and } \forall v < t. \ (v > u \Rightarrow \langle S, v \rangle \models \varphi)) \]

Examples

Non-strict modalities:

\[ \varphi \tilde{\mathbf{U}} \psi \overset{\text{def}}{=} \psi \lor (\varphi \land \varphi \mathbf{U} \psi) \]

\[ \varphi \tilde{\mathbf{S}} \psi \overset{\text{def}}{=} \psi \lor (\varphi \land \varphi \mathbf{S} \psi) \]
Until and Since

\[ \varphi \mathbf{U} \psi : \langle S, t \rangle \models \varphi \mathbf{U} \psi \iff \exists u > t. (\langle S, u \rangle \models \psi \text{ and } \forall v > t. (v < u \Rightarrow \langle S, v \rangle \models \varphi)) \]

(\varphi \ “until” \ \psi)

\[ \varphi \mathbf{S} \psi : \langle S, t \rangle \models \varphi \mathbf{S} \psi \iff \exists u < t. (\langle S, u \rangle \models \psi \text{ and } \forall v < t. (v > u \Rightarrow \langle S, v \rangle \models \varphi)) \]

(\varphi \ “since” \ \psi)

Examples

“Next” modality: \( \bot \mathbf{U} \varphi \equiv \mathbf{X} \varphi \) in discrete time
Until and Since

\[ \varphi \mathbf{U} \psi : \langle S, t \rangle \models \varphi \mathbf{U} \psi \iff \exists u > t. (\langle S, u \rangle \models \psi \text{ and } \forall v > t. (v < u \Rightarrow \langle S, v \rangle \models \varphi)) \]

(\varphi \text{ “until” } \psi)

\[ \varphi \mathbf{S} \psi : \langle S, t \rangle \models \varphi \mathbf{S} \psi \iff \exists u < t. (\langle S, u \rangle \models \psi \text{ and } \forall v < t. (v > u \Rightarrow \langle S, v \rangle \models \varphi)) \]

(\varphi \text{ “since” } \psi)

Examples

“Next” modality: \[ \bot \mathbf{U} \varphi \overset{\text{def}}{=} \mathbf{X} \varphi \] in discrete time
\[ \bot \mathbf{U} \varphi \equiv \bot \] in dense time
Until and Since

\[ \varphi U \psi : \langle S, t \rangle \models \varphi U \psi \iff \exists u > t. (\langle S, u \rangle \models \psi \text{ and } \forall v > t. (v < u \Rightarrow \langle S, v \rangle \models \varphi)) \]

(\varphi “until” \psi)

\[ \varphi S \psi : \langle S, t \rangle \models \varphi S \psi \iff \exists u < t. (\langle S, u \rangle \models \psi \text{ and } \forall v < t. (v > u \Rightarrow \langle S, v \rangle \models \varphi)) \]

(\varphi “since” \psi)

Examples

“Next” modality: \( \bot U \varphi \overset{\text{def}}{=} X \varphi \) in discrete time

“Previous” modality: \( \bot S \varphi \overset{\text{def}}{=} X^{-1} \varphi \) in discrete time
Some remarks about “dual until” and “weak until”

Modality $\mathbf{R}$ is defined as the dual of modality $\mathbf{U}$:

$$\varphi \mathbf{R} \psi \overset{\text{def}}{=} \neg (\neg \varphi) \mathbf{U} (\neg \psi).$$
Some remarks about “dual until” and “weak until”

Modality $R$ is defined as the dual of modality $U$:

$$\phi \ R \ \psi \ \overset{\text{def}}{=} \neg (\neg \phi) \ U (\neg \psi).$$

$\phi \ R \ \psi : \langle S, t \rangle \models \phi \ R \ \psi \ \iff \ \forall u > t. (\langle S, u \rangle \not\models \psi \ \Rightarrow \ \exists v > t. (v < u \ \land \ \langle S, v \rangle \models \phi))$
Some remarks about “dual until” and “weak until”

Modality $\mathbf{R}$ is defined as the dual of modality $\mathbf{U}$:

$$\varphi \mathbf{R} \psi \overset{\text{def}}{=} \neg (\neg \varphi) \mathbf{U} (\neg \psi).$$

Proposition

Over discrete time,

$$\varphi \mathbf{R} \psi \equiv G \psi \lor \psi \mathbf{U} (\varphi \land \psi).$$
Some remarks about “dual until” and “weak until”

Modality $W$ is a relaxed version of modality $U$:

$$\varphi W \psi \overset{\text{def}}{=} G \varphi \lor \varphi U \psi$$
Some remarks about “dual until” and “weak until”

Modality $W$ is a relaxed version of modality $U$:

$$\phi W \psi \overset{\text{def}}{=} G \phi \lor \phi U \psi$$

$\phi W \psi : \langle S, t \rangle \models \phi W \psi \iff \forall u > t. (\langle S, u \rangle \models \phi \lor \exists u > t. (\langle S, u \rangle \models \psi \land \forall v > t. (v < u \Rightarrow \langle S, v \rangle \models \phi)))$
Some remarks about “dual until” and “weak until”

Modality $W$ is a relaxed version of modality $U$:

$$\varphi W \psi \overset{\text{def}}{=} G \varphi \lor \varphi U \psi$$

$\varphi W \psi : \langle S, t \rangle \models \varphi W \psi \iff \forall u > t. (\langle S, u \rangle \models \varphi \lor \exists u > t. (\langle S, u \rangle \models \psi \land \forall v > t. (v < u \Rightarrow \langle S, v \rangle \models \varphi)))$

**Proposition**

*Over discrete time,*

$$\varphi W \psi \equiv \psi R (\varphi \lor \psi)$$
Given modalities $M_1$ to $M_n$ and a set $AP$ of atomic propositions, the logic $L_{AP}(M_1, \ldots, M_n)$ is defined by the following grammar:

$$L_{AP}(M_1, \ldots, M_n) \ni \varphi, \psi, \ldots ::= \top \mid p \mid \neg \varphi \mid \varphi \lor \psi \mid M_i(\varphi, \psi, \ldots)$$

where $p$ ranges over $AP$, and $i$ over $\{1, \ldots, n\}$. 
**Linear-time Temporal Logic**

**Definition**

Given modalities $M_1$ to $M_n$ and a set $AP$ of atomic propositions, the logic $L_{AP}(M_1, ..., M_n)$ is defined by the following grammar:

$$L_{AP}(M_1, ..., M_n) \ni \varphi, \psi, ... ::= \top | p | \neg \varphi | \varphi \lor \psi | M_i(\varphi, \psi, ...)$$

where $p$ ranges over $AP$, and $i$ over $\{1, ..., n\}$.

**Definition**

- $LTL^{+\text{Past}} = L(U, S)$
- $LTL = L(U)$
Linear-time Temporal Logic

Definition
Given modalities $M_1$ to $M_n$ and a set $AP$ of atomic propositions, the logic $L_{AP}(M_1, ..., M_n)$ is defined by the following grammar:

$L_{AP}(M_1, ..., M_n) \ni \varphi, \psi, ... ::= \top | p | \neg \varphi | \varphi \lor \psi | M_i(\varphi, \psi, ...)$

where $p$ ranges over $AP$, and $i$ over $\{1, ..., n\}$.

Definition
- $LTL^{+}\text{Past} = \mathcal{L}(U, S)$
- $LTL = \mathcal{L}(U)$
- $LTL = \mathcal{L}(\tilde{U}, X)$ often prefered in discrete-time.

⚠️ Both definitions of LTL are not equivalent w.r.t. succinctness.
What LTL$+$Past can express

LTL$+$Past can express properties such as:

- the time-line is discrete:

\[ G(F \top \implies \bot U \top) \]
What LTL+Past can express

LTL+Past can express properties such as:

- the time-line is discrete:
  \[ G(\mathbf{F} \top \Rightarrow \bot \mathbf{U} \top) \]

- (over discrete time) the linear structure is infinite:
  \[ G \mathbf{F} \top \]
What LTL+Past can express

LTL+Past can express properties such as:

- the time-line is discrete:
  \[ G ( F \top \Rightarrow \bot U \top) \]

- (over discrete time) the linear structure is infinite:
  \[ G F \top \]

LTL+Past can’t express properties such as:

- (over discrete time) a formula holds at every even position:
What LTL+Past can express

LTL+Past can express properties such as:

- the time-line is discrete:

\[ \mathbf{G (F T \Rightarrow \bot U T)} \]

- (over discrete time) the linear structure is infinite:

\[ \mathbf{G F T} \]

LTL+Past can’t express properties such as:

- (over discrete time) a formula holds at every even position:

\[ A_n \]

\[ B_n \]
What LTL+Past can express?

Proposition

$LTL+Past$ can encode the behavior of a (deterministic) linear-space Turing machine with a linear-size formula.
What LTL+Past can express?

Proposition

\[ \text{LTL}^+\text{Past can encode the behavior of a (deterministic) linear-space Turing machine with a linear-size formula.} \]

Proof (sketch).

- the content of the tape (of size \( n \)) and the tape head are encoded on \( n \) successive positions;

\[
\begin{array}{ccccccc}
a & b & a & a & c & \# & \#
\end{array}
\]

\[
\begin{array}{ccccccccccc}
\text{sep} & a & b & a & a & c & \# & \# & \text{sep}
\end{array}
\]

\( q \)
What LTL+Past can express?

**Proposition**

*LTL+Past can encode the behavior of a (deterministic) linear-space Turing machine with a linear-size formula.*

**Proof (sketch).**

- the content of the tape (of size $n$) and the tape head are encoded on $n$ successive positions;
- We write LTL+Past formulas ensuring that:

$$\sim \text{ the configuration separator repeats every } n + 2 \text{ position:}$$

$$sep \land G(sep \Rightarrow X(\neg sep \land X(\neg sep \land \ldots X sep\ldots))))$$

![Diagram of a Turing machine tape and head]
What LTL+Past can express?

**Proposition**

LTL+Past can encode the behavior of a (deterministic) linear-space Turing machine with a linear-size formula.

**Proof (sketch).**

- the content of the tape (of size $n$) and the tape head are encoded on $n$ successive positions;
- We write LTL+Past formulas ensuring that:
  - there is one tape head in each configuration:

$$G(sep \Rightarrow (\neg sep \cup (Q \land \neg Q \cup sep)))$$

```
<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>#</td>
<td>#</td>
<td></td>
</tr>
</tbody>
</table>
```

```
sep a b a a c # # sep
q
```
What LTL+Past can express?

Proposition

LTL+Past can encode the behavior of a (deterministic) linear-space Turing machine with a linear-size formula.

Proof (sketch).

• the content of the tape (of size $n$) and the tape head are encoded on $n$ successive positions;

• We write LTL+Past formulas ensuring that:

  $\Rightarrow$ the initial configuration is correct:

  $X(a \land q_0 \land X(b \land ...))$

\[
\begin{array}{cccccc}
  a & b & a & a & c & \# & \#
\end{array}
\]

\[
\begin{array}{cccccccc}
  \text{sep} & a & b & a & a & c & \# & \# & \text{sep}
\end{array}
\]

\[
q
\]
What LTL+Past can express?

**Proposition**

$LTL+Past$ can encode the behavior of a (deterministic) linear-space Turing machine with a linear-size formula.

**Proof (sketch).**

- the content of the tape (of size $n$) and the tape head are encoded on $n$ successive positions;
- We write $LTL+Past$ formulas ensuring that:

  $\sim\Rightarrow$ the transitions are applied correctly:

\[
G((a \land q) \Rightarrow X^n(q' \land X a'))
\]
What \( \text{LTL} + \text{Past} \) can express?

**Proposition**

\( \text{LTL} + \text{Past} \) can encode the behavior of a (deterministic) linear-space Turing machine with a linear-size formula.

**Proof (sketch).**

- the content of the tape (of size \( n \)) and the tape head are encoded on \( n \) successive positions;
- We write \( \text{LTL} + \text{Past} \) formulas ensuring that:

\[
G((b \land \neg Q) \Rightarrow X^{n+1} b)
\]

other letters are copied verbatim:
What \( LTL+\text{Past} \) can express?

**Proposition**

\( LTL+\text{Past} \) can encode the behavior of a (deterministic) linear-space Turing machine with a linear-size formula.

**Corollary**

Satisfiability of a \( LTL+\text{Past} \) formula is PSPACE-hard.

In fact, we have:

**Theorem (Sistla & Clarke, 1985)**

Satisfiability of a \( LTL+\text{Past} \) formula is PSPACE-complete.
Outline of today’s lecture

1 Introduction

2 The notion of expressiveness
   - Distinguishing power
   - Expressive power
   - Succinctness

3 Linear-time temporal logics
   - Definitions, basic formulas
   - LTL, PLTL and FO
   - Normal forms for LTL+Past
Proposition

Any $\mathit{LTL+Past}$ formula can be expressed in first-order logic.
LTL$+$Past and first-order logic

Proposition

Any LTL$+$Past formula can be expressed in first-order logic.

By a more careful examination:

Proposition

Any LTL$+$Past formula can be expressed in first-order logic using at most three variables.
LTL+Past and first-order logic

**Proposition**

Any LTL+Past formula can be expressed in first-order logic.

By a more careful examination:

**Proposition**

Any LTL+Past formula can be expressed in first-order logic using at most three variables.

**Example**

\( \langle A, 0 \rangle \models (\text{green } U \text{ blue}) U \text{ red} \) is equivalent to

\[
\exists x > 0. \left[ \langle A, x \rangle \models \text{red} \land \forall y > 0. \ (y < x \Rightarrow \\
(\exists z > y. \langle A, z \rangle \models \text{green} \land \\
\forall x > y. \ (x < z \Rightarrow \langle A, x \rangle \models \text{blue}))) \right]
\]
### Theorem (Kamp, 1968)

Over \( \text{Lin}(\mathbb{R}^+) \) and \( \text{Lin}(\mathbb{Z}^+) \), any first-order property can be expressed in \( \text{LTL}+\text{Past} \).
### Theorem (Kamp, 1968)

Over $\text{Lin}(\mathbb{R}^+)$ and $\text{Lin}(\mathbb{Z}^+)$, any first-order property can be expressed in LTL+Past.

⚠️ This result fails to hold over $\text{Lin}(\mathbb{Q}^+)$, where first-order logic has strictly more distinguishing power than LTL+Past.
Theorem (Gabbay et al., 1980)

Over $\text{Lin}^0(\mathbb{Z}^+)$, any first-order property can be expressed in LTL.
LTL+Past and first-order logic

Theorem (Gabbay et al., 1980)

Over $\text{Lin}^0(\mathbb{Z}^+)$, any first-order property can be expressed in LTL.

⚠️ This result fails to hold over $\text{Lin}^0(\mathbb{R}^+)$. 
LTL+Past and first-order logic

Theorem (Gabbay et al., 1980)

Over $\text{Lin}^0(\mathbb{Z}^+)$, any first-order property can be expressed in LTL.

⚠️ This result fails to hold over $\text{Lin}^0(\mathbb{R}^+)$. 

\begin{align*}
A & \quad g \quad g \quad g \quad g \quad g \quad g \\
B & \quad g \quad g \quad g \quad ggg \quad g \quad g \quad g
\end{align*}
Theorem (Gabbay et al., 1980)

Over $\text{Lin}^0(\mathbb{Z}^+)$, any first-order property can be expressed in LTL.

This result fails to hold over $\text{Lin}^0(\mathbb{R}^+)$. 

\begin{align*}
\mathcal{A} & \quad g \quad g \quad g \quad g \quad g \\
\mathcal{B} & \quad g \quad g \quad g \quad ggg \quad g \quad g \quad g 
\end{align*}

Lemma

No LTL formula can distinguish between $\langle \mathcal{A}, 0 \rangle$ and $\langle \mathcal{B}, 0 \rangle$. 
Theorem (Gabbay et al., 1980)

Over $\text{Lin}^0(\mathbb{Z}^+)$, any first-order property can be expressed in LTL.

This result fails to hold over $\text{Lin}^0(\mathbb{R}^+)$. 

A 
---
\begin{align*}
g & g & g & g & g & g
\end{align*}

B 
---
\begin{align*}
g & g & g & g & g & g
\end{align*}

Lemma

The following formula distinguishes between those structures:

$$\exists t > 0. \forall u < t. \exists v. (u < v < t \land \text{green}(v)).$$
LTL+Past and first-order logic

**Theorem (Gabbay et al., 1980)**

Over $\text{Lin}^0(\mathbb{Z}^+)$, any first-order property can be expressed in LTL.

**Corollary**

Over $\text{Lin}^0(\mathbb{Z}^+)$, LTL+Past and LTL are equally expressive.
**Succinctness issues**

<table>
<thead>
<tr>
<th>Theorem (Stockmeyer, 1974 + Sistla &amp; Clarke, 1985)</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Over</em> $\text{Lin}(\mathbb{R}^+)$ and $\text{Lin}(\mathbb{Z}^+)$, first-order logic can be non-elementarily more succinct than LTL+Past.</td>
</tr>
</tbody>
</table>
Succinctness issues

**Theorem (Stockmeyer, 1974 + Sistla & Clarke, 1985)**

Over Lin$(\mathbb{R}^+)$ and Lin$(\mathbb{Z}^+)$, first-order logic can be non-elementarily more succinct than LTL+Past.

**Sketch of proof.**

- Stockmeyer proved that some satisfiable first-order formulas don’t admit “small models”.
- On the other hand, Sistla and Clarke proved that satisfiable LTL+Past formulas always have exponential-size models.
Succinctness issues

Theorem (Stockmeyer, 1974 + Sistla & Clarke, 1985)

Over \( \text{Lin}(\mathbb{R}^+) \) and \( \text{Lin}(\mathbb{Z}^+) \), first-order logic can be non-elementarily more succinct than LTL+Past.

Theorem (Laroussinie, Markey, & Schnoebelen, 2002)

Over \( \text{Lin}^0(\mathbb{Z}^+) \), LTL+Past can be exponentially more succinct than LTL.
Succinctness issues

Theorem (Stockmeyer, 1974 + Sistla & Clarke, 1985)

Over $\text{Lin}(\mathbb{R}^+)$ and $\text{Lin}(\mathbb{Z}^+)$, first-order logic can be non-elementarily more succinct than $\text{LTL}+\text{Past}$.

Theorem (Laroussinie, Markey, & Schnoebelen, 2002)

Over $\text{Lin}^0(\mathbb{Z}^+)$, $\text{LTL}+\text{Past}$ can be exponentially more succinct than $\text{LTL}$.

Sketch of proof.

- Any LTL property can be captured by an exponential-size Büchi automaton (see tomorrow’s lecture).
- We build a family of properties that can be expressed by polynomial-size formulas in $\text{LTL}+\text{Past}$, but are only accepted by doubly-exponential size Büchi automata.
First-order logic with two variables

**Lemma**

Any formula in $\mathcal{L}(F, G, F^{-1}, G^{-1})$ can be expressed by a first-order formula using at most two variables.
First-order logic with two variables

**Lemma**

Any formula in $\mathcal{L}(F, G, F^{-1}, G^{-1})$ can be expressed by a first-order formula using at most two variables.

**Theorem (Etessami, Vardi, Wilke, 1997)**

First-order logic with two variables has the same expressive power as $\mathcal{L}(F, G, F^{-1}, G^{-1})$. 
First-order logic with two variables

**Lemma**

Any formula in $L(F, G, F^{-1}, G^{-1})$ can be expressed by a first-order formula using at most two variables.

**Theorem (Etessami, Vardi, Wilke, 1997)**

First-order logic with two variables has the same expressive power as $L(F, G, F^{-1}, G^{-1})$.

The effective translation was given originally for $\text{Lin}(\mathbb{Z}^+)$, but holds for any class of linear models. It involves an exponential blowup in the size of the formula.
Lemma

Any formula in $\mathcal{L}(F, G, F^{-1}, G^{-1})$ can be expressed by a first-order formula using at most two variables.

Theorem (Etessami, Vardi, Wilke, 1997)

First-order logic with two variables has the same expressive power as $\mathcal{L}(F, G, F^{-1}, G^{-1})$.

The effective translation was given originally for $\text{Lin}(\mathbb{Z}^+)$, but holds for any class of linear models. It involves an exponential blowup in the size of the formula.

Theorem (Etessami, Vardi, Wilke, 1997)

Over $\text{Lin}(\mathbb{Z}^+)$, first-order logic with two variables is exponentially more succinct than $\mathcal{L}(F, G, F^{-1}, G^{-1})$. 
Outline of today’s lecture

1. Introduction

2. The notion of expressiveness
   - Distinguishing power
   - Expressive power
   - Succinctness

3. Linear-time temporal logics
   - Definitions, basic formulas
   - LTL, PLTL and FO
   - Normal forms for LTL+Past
Separation of formulas

**Theorem (Gabbay, 1987)**

Over $\text{Lin}(\mathbb{Z}^+)$, any $\text{LTL}+\text{Past}$ formula can be written as a boolean combination of pure-past and pure-future formulas.
Separation of formulas

**Theorem (Gabbay, 1987)**

*Over \( \text{Lin}(\mathbb{Z}^+) \), any LTL+Past formula can be written as a boolean combination of pure-past and pure-future formulas.*

**Sketch of the proof.**

- (effective) recursive transformation of the formula;
Separation of formulas

Theorem (Gabbay, 1987)

Over $\text{Lin}(\mathbb{Z}^+)$, any LTL+Past formula can be written as a boolean combination of pure-past and pure-future formulas.

Sketch of the proof.

- (effective) recursive transformation of the formula;
- 8 transformation rules; For example:

\[
\text{red } S (\text{green } \land \text{ blue } U \text{ yellow}) \equiv
\]
Separation of formulas

**Theorem (Gabbay, 1987)**

Over $\text{Lin}(\mathbb{Z}^+)$, any $\text{LTL} + \text{Past}$ formula can be written as a boolean combination of pure-past and pure-future formulas.

**Sketch of the proof.**

- (effective) recursive transformation of the formula;
- 8 transformation rules; For example:

$$\text{red } S (\text{green } \land \text{ blue } U \text{ yellow}) \equiv$$
Separation of formulas

Theorem (Gabbay, 1987)

Over \( \text{Lin}(\mathbb{Z}^+) \), any \( \text{LTL}_+\text{Past} \) formula can be written as a boolean combination of pure-past and pure-future formulas.

Sketch of the proof.

- (effective) recursive transformation of the formula;
- 8 transformation rules; For example:

\[
\text{red } S (\text{green } \land \text{ blue } \bigcup \text{ yellow}) \equiv
\]
Separation of formulas

**Theorem (Gabbay, 1987)**

Over $\text{Lin}(\mathbb{Z}^+)$, any $\text{LTL+Past}$ formula can be written as a boolean combination of pure-past and pure-future formulas.

**Sketch of the proof.**

- (effective) recursive transformation of the formula;
- 8 transformation rules; For example:

$$\text{red } S (\text{green } \land \text{ blue } U \text{ yellow}) \equiv \text{red } S (\text{yellow } \land \text{ red } \land (\text{red } \land \text{ blue}) S \text{ green}) \lor$$
Separation of formulas

**Theorem (Gabbay, 1987)**

Over $\text{Lin}(\mathbb{Z}^+)$, any $\text{LTL} + \text{Past}$ formula can be written as a boolean combination of pure-past and pure-future formulas.

**Sketch of the proof.**

- (effective) recursive transformation of the formula;
- 8 transformation rules; For example:

$$\text{red } S (\text{green } \land \text{ blue } \mathbf{U} \text{ yellow}) \equiv \text{red } S (\text{yellow } \land \text{ red } \land (\text{red } \land \text{ blue}) S \text{ green}) \lor \text{yellow } \land (\text{red } \land \text{ blue}) S \text{ green } \lor \text{blue } \land (\text{red } \land \text{ blue}) S \text{ green}$$
Separation of formulas

Theorem (Gabbay, 1987)

Over $\text{Lin}(\mathbb{Z}^+)$, any $\text{LTL}+\text{Past}$ formula can be written as a boolean combination of pure-past and pure-future formulas.

Sketch of the proof.

- (effective) recursive transformation of the formula;
- 8 transformation rules; For example:

\[
\begin{align*}
\text{red } S (\text{green } \land \text{ blue } U \text{ yellow}) & \equiv \\
\text{red } S (\text{yellow } \land \text{ red } \land (\text{red } \land \text{ blue}) S \text{ green}) & \lor \\
\text{yellow } \land (\text{red } \land \text{ blue}) S \text{ green} & \lor \\
\text{blue } \land (\text{red } \land \text{ blue}) S \text{ green } \land \text{ blue } U \text{ yellow}
\end{align*}
\]
Separation of formulas

**Theorem (Gabbay, 1987)**

Over Lin($\mathbb{Z^+}$), any LTL+Past formula can be written as a boolean combination of pure-past and pure-future formulas.

*Sketch of the proof.*

- (effective) recursive transformation of the formula;
- 8 transformation rules;
- Termination: the number of alternations of $U$ and $S$ decreases at each step.
Theorem (Gabbay, 1987)

Over $\text{Lin}(\mathbb{Z}^+)$, any $\text{LTL}^+$ formula can be written as a boolean combination of pure-past and pure-future formulas.

Sketch of the proof.

- (effective) recursive transformation of the formula;
- 8 transformation rules;
- Termination: the number of alternations of $\textbf{U}$ and $\textbf{S}$ decreases at each step.

Theorem

This transformation induces a non-elementary blowup.
Theorem (Lichtenstein, Pnueli & Zuck, 1985)

Over $\text{Lin}^0(\mathbb{Z}^+)$, any formula in LTL+Past is equivalent to a formula under the following normal form:

$$\bigvee_{i=1..n} (F G \varphi_i \land G F \psi_i)$$

where $\varphi_i$ and $\psi_i$ are pure-past formulas.
Normal form for LTL+Past formulas

Theorem (Lichtenstein, Pnueli & Zuck, 1985)

Over Lin^0(\mathbb{Z}^+), any formula in LTL+Past is equivalent to a formula under the following normal form:

\[\bigvee_{i=1..n} (F G \varphi_i \land G F \psi_i)\]

where \(\varphi_i\) and \(\psi_i\) are pure-past formulas.

Example

\[
green \mathbf{U} \ red \equiv F(red \land G^{-1}green) \\
\equiv FG(F^{-1}(red \land G^{-1}green))
\]
Theorem (Lichtenstein, Pnueli & Zuck, 1985)

Over $\text{Lin}^0(\mathbb{Z}^+)$, any formula in LTL+Past is equivalent to a formula under the following normal form:

$$
\bigvee_{i=1..n} (F G \varphi_i \land G F \psi_i)
$$

where $\varphi_i$ and $\psi_i$ are pure-past formulas.

Example

$$
G(\text{green} \Rightarrow \text{green} \ U \text{red}) \equiv
$$
Normal form for LTL+Past formulas

Theorem (Lichtenstein, Pnueli & Zuck, 1985)

Over $\text{Lin}^0(\mathbb{Z}^+)$, any formula in LTL+Past is equivalent to a formula under the following normal form:

$$\bigvee_{i=1..n} (F G \varphi_i \land G F \psi_i)$$

where $\varphi_i$ and $\psi_i$ are pure-past formulas.

Example

$$G(green \Rightarrow green \cup red) \equiv$$
Normal form for LTL+Past formulas

Theorem (Lichtenstein, Pnueli & Zuck, 1985)

Over Lin⁰(\mathbb{Z}^+), any formula in LTL+Past is equivalent to a formula under the following normal form:

\[ \bigvee_{i=1..n} (F G \varphi_i \land G F \psi_i) \]

where \( \varphi_i \) and \( \psi_i \) are pure-past formulas.

Example

\[ G(\text{green} \Rightarrow \text{green} U \text{red}) \equiv G(\text{green} \lor \text{red} \lor (\neg \text{green}) S \text{red} \lor G^{-1} \neg \text{green}) \]
Normal form for LTL+Past formulas

Theorem (Lichtenstein, Pnueli & Zuck, 1985)

Over \( \text{Lin}^{0}(\mathbb{Z}^{+}) \), any formula in LTL+Past is equivalent to a formula under the following normal form:

\[
\bigvee_{i=1..n} (F \, G \, \varphi_{i} \land G \, F \, \psi_{i})
\]

where \( \varphi_{i} \) and \( \psi_{i} \) are pure-past formulas.

Example

\[
G(\text{green } \Rightarrow \text{ green } U \text{ red}) \equiv F \, G \, G^{-1}(\text{green } \lor \text{ red } \lor (\neg \text{ green }) \, S \, \text{ red } \lor G^{-1} \neg \text{ green}))
\]
Normal form for LTL+Past formulas

Theorem (Lichtenstein, Pnueli & Zuck, 1985)

Over Lin⁰(ℤ⁺), any formula in LTL+Past is equivalent to a formula under the following normal form:

$$\bigvee_{i=1..n} (F G \varphi_i \land G F \psi_i)$$

where \( \varphi_i \) and \( \psi_i \) are pure-past formulas.

Example

\[
G(green \Rightarrow green \ U \ red) \equiv \\
F G G^{-1}(green \lor red \lor (\neg green) S red \lor G^{-1} \neg green) \\
\land G F(G^{-1} \neg green \lor (\neg green S red))
\]
Expressiveness of Temporal Logics

François Laroussinie and Nicolas Markey

Lab. Specification et Verification
ENS Cachan & CNRS, France

August 1, 2006
Outline of today’s lecture

4 LTL+Past and the $\mu$-calculus

5 LTL+Past and Büchi automata
   - Büchi automata
   - From LTL+Past to Büchi automata
   - Büchi automata are more expressive
   - Alternating Büchi automata
   - Application: Succinctness of LTL+Past

6 Stuttering
   - The stuttering principle
   - The generalized stuttering principle

7 Ehrenfeucht-Fraïssé games
   - The rules of the game
   - EF games and the Until-Since hierarchy
Introduction to the second lecture

- In this second course, we focus on discrete time.
- As is usual in that case, we use the following definitions:
  - \( \text{LTL} = \mathcal{L}(U, X) \),
  - \( \text{LTL+Past} = \mathcal{L}(U, S, X, X^{-1}) \).

  Where \( U \) and \( S \) have their non-strict meaning, e.g.:
  
  \[ \varphi U \psi \text{ holds iff } \]
  
  - either \( \psi \) holds,
  - or \( \varphi \) holds, and \( \varphi U \psi \) holds in the next location,

  In other words, we have the following equivalence:

  \[ \varphi U \psi \equiv \psi \lor (\varphi \land X(\varphi U \psi)) \].

- Linear structures are seen as (infinite) words over the alphabet \( 2^{\mathcal{AP}} \).
Outline of today’s lecture

4 LTL+Past and the $\mu$-calculus

5 LTL+Past and Büchi automata
   - Büchi automata
   - From LTL+Past to Büchi automata
   - Büchi automata are more expressive
   - Alternating Büchi automata
   - Application: Succinctness of LTL+Past

6 Stuttering
   - The stuttering principle
   - The generalized stuttering principle

7 Ehrenfeucht-Fraïssé games
   - The rules of the game
   - EF games and the Until-Since hierarchy
LTL+Past and the $\mu$-calculus

Definition

The \textit{linear-time $\mu$-calculus} is an extension of $\mathcal{L}(X)$ with fixpoint operators:

\[
\mu\text{-calculus} \ni \varphi, \psi ::= \top \mid p \mid \neg p \mid Z \mid \varphi \lor \psi \mid \varphi \land \psi \mid X\psi \mid X^{-1}\psi \mid \mu Z \varphi \mid \nu Z \varphi
\]

where $p$ ranges over AP and $Z$ ranges over a finite set of variables.
LTL+Past and the $\mu$-calculus

Definition

The *linear-time $\mu$-calculus* is an extension of $\mathcal{L}(X)$ with fixpoint operators:

$$\mu\text{-calculus} \ni \varphi, \psi ::= \top \mid p \mid \neg p \mid Z \mid \varphi \lor \psi \mid \varphi \land \psi \mid X \psi \mid X^{-1} \psi \mid \mu Z \varphi \mid \nu Z \varphi$$

where $p$ ranges over AP and $Z$ ranges over a finite set of variables.

Example

$$\mu Z (\text{green} \lor X Z)$$
Let \( T \) be a linear structure. Given a formula \( \varphi(Z) \), the set of positions satisfying \( \mu Z \varphi(Z) \) is \[
\bigcap \{U \subseteq T \mid \varphi(U) \subseteq U\}
\]
LTL+Past and the $\mu$-calculus

**Theorem (Knaster, 1928 & Tarski, 1955)**

Let $T$ be a linear structure. Given a formula $\varphi(Z)$, the set of positions satisfying $\mu Z \varphi(Z)$ is

$$\bigcap \{U \subseteq T \mid \varphi(U) \subseteq U\}$$

Moreover, fixpoints can be computed iteratively:

**Theorem (Knaster, 1928 & Tarski, 1955)**

The set of positions satisfying $\mu Z \varphi(Z)$ is the limit of the following sequence:

$$\begin{align*}
\llbracket \mu Z \varphi(Z) \rrbracket_0 &= \emptyset \\
\llbracket \mu Z \varphi(Z) \rrbracket_{i+1} &= \{t \in T \mid \langle T, t \rangle \models \varphi(\llbracket \mu Z \varphi(Z) \rrbracket_i)\}
\end{align*}$$
LTL+Past and the $\mu$-calculus

Example

Consider the following linear structure (represented as a word):

$$T = g \ r \ r \ g \ g \ b \ r \ b \ g \ r \ r \ g \ r \ r \ r \ r \ r \ r \ r \ ...$$

and the following formula:

$$\mu Z (\text{green} \lor X Z).$$
Example

Consider the following linear structure (represented as a word):

\[ T = g \ r \ r \ g \ g \ b \ r \ b \ g \ r \ r \ g \ r \ r \ r \ r \ r \ r \ r \ r \ r \ r \ r \ r \ r \ r \ r \ r \ r \ ... \]

and the following formula:

\[ \mu Z (\text{green} \lor X Z) \].

Then:

\[ \llbracket \mu Z (g \lor X Z) \rrbracket_0 = \emptyset \]
Example

Consider the following linear structure (represented as a word):

\[ T = g \ r \ r \ g \ g \ b \ r \ b \ g \ r \ r \ g \ r \ r \ r \ r \ r \ r \ r \ ... \]

and the following formula:

\[ \mu Z \ (\text{green} \lor X \ Z). \]

Then:

\[ \mu Z \ (g \lor X \ Z) \equiv_0 \bot \]
Consider the following linear structure (represented as a word):

\[ T = \textcolor{green}{g} \textcolor{red}{r} \textcolor{red}{r} \textcolor{green}{g} \textcolor{green}{b} \textcolor{red}{r} \textcolor{red}{b} \textcolor{green}{g} \textcolor{red}{r} \textcolor{red}{r} \textcolor{green}{g} \textcolor{red}{r} \textcolor{red}{r} \textcolor{red}{r} \textcolor{red}{r} \textcolor{red}{r} \textcolor{red}{r} \ldots \]

and the following formula:

\[ \mu Z (\text{green} \lor X Z). \]

Then:

\[ [\mu Z (\text{g} \lor X Z)]_1 = \{ t \in T \mid \langle T, t \rangle \models g \lor X \bot \} \]
Example

Consider the following linear structure (represented as a word):

\[ T = \text{green red red green green blue red blue green red red red red red red red ...} \]

and the following formula:

\[ \mu Z (\text{green} \lor X Z). \]

Then:

\[ \mu Z (g \lor X Z) \equiv_1 g \]
Example

Consider the following linear structure (represented as a word):

\[ T = \text{g r r g g b r b g r r g r r r r r r ...} \]

and the following formula:

\[ \mu Z (\text{green} \lor X Z) \]

Then:

\[ \llbracket \mu Z (g \lor X Z) \rrbracket_2 = \{ t \in T \mid \langle T, t \rangle \models g \lor X g \} \]
Example

Consider the following linear structure (represented as a word):

\[ T = \text{g r r g g b r b g r r g r r r r r r r r r ...} \]

and the following formula:

\[ \mu Z (\text{green} \lor X Z). \]

Then:

\[ \mu Z (g \lor X Z) \equiv_2 g \lor X g \]
Example

Consider the following linear structure (represented as a word):

\[ T = \text{green} \text{ red red green green blue red blue green green red red red red ...} \]

and the following formula:

\[ \mu Z (\text{green} \lor X Z). \]

Then:

\[ \mu Z (g \lor X Z) \equiv_n g \lor X g \lor XX g \lor ... \lor X^{n-1} g \]
Example

Consider the following linear structure (represented as a word):

\[ T = \text{green} \quad \text{red} \quad \text{red} \quad \text{green} \quad \text{green} \quad \text{blue} \quad \text{red} \quad \text{blue} \quad \text{blue} \quad \text{green} \quad \text{green} \quad \text{red} \quad \text{red} \quad \text{red} \quad \text{red} \quad \text{red} \quad \text{red} \quad \text{red} \quad \text{red} \quad \ldots \]

and the following formula:

\[ \mu Z (\text{green} \lor X Z). \]

Then:

\[ \mu Z (g \lor X Z) \equiv F g \]
LTL+Past and the $\mu$-calculus

$$\varphi U \psi \equiv \psi \lor (\varphi \land X(\varphi U \psi)).$$

From this equivalence, we get:

$$\varphi U \psi \equiv \mu Z (\psi \lor (\varphi \land X Z))$$
LTL+Past and the μ-calculus

\[ \varphi U \psi \equiv \psi \lor (\varphi \land X(\varphi U \psi)) \].

From this equivalence, we get:

\[ \varphi U \psi \equiv \mu Z (\psi \lor (\varphi \land X Z)) \]

Thus:

**Proposition**

μ-calculus is at least as expressive as LTL+Past.
LTL+Past and the \( \mu \)-calculus

In fact:

**Proposition**

\( \mu \)-calculus is strictly more expressive than LTL+Past.
LTL+Past and the $\mu$-calculus

In fact:

Proposition

$\mu$-calculus is strictly more expressive than LTL+Past.

Proof.

The negation of

“green occurs at every even position”

is

“$\neg$green occurs at some even position”

i.e.

$\neg$green $\lor$ XX $\neg$green $\lor$ XXXX $\neg$green $\lor$ ...
LTL+Past and the $\mu$-calculus

In fact:

**Proposition**

$\mu$-calculus is strictly more expressive than LTL+Past.

**Proof.**

The negation of

"green occurs at every even position"

is

"$\neg$ green occurs at some even position"

i.e.

$$\neg \text{green} \lor XX \neg \text{green} \lor XXXX \neg \text{green} \lor \ldots$$

It can be written as

$$\mu Z (\neg \text{green} \lor XX Z)$$
Outline of today’s lecture

4 LTL+Past and the $\mu$-calculus

5 LTL+Past and Büchi automata
   - Büchi automata
   - From LTL+Past to Büchi automata
   - Büchi automata are more expressive
   - Alternating Büchi automata
   - Application: Succinctness of LTL+Past

6 Stuttering
   - The stuttering principle
   - The generalized stuttering principle

7 Ehrenfeucht-Fraïssé games
   - The rules of the game
   - EF games and the Until-Since hierarchy
Outline of today’s lecture

4 LTL+Past and the $\mu$-calculus

5 LTL+Past and Büchi automata
   - Büchi automata
   - From LTL+Past to Büchi automata
   - Büchi automata are more expressive
   - Alternating Büchi automata
   - Application: Succinctness of LTL+Past

6 Stuttering
   - The stuttering principle
   - The generalized stuttering principle

7 Ehrenfeucht-Fraïssé games
   - The rules of the game
   - EF games and the Until-Since hierarchy
LTL+Past and Büchi automata

Finite-state automata are a powerful formalism for defining languages.

Example

\[ G(\text{green} \Rightarrow F \text{red}) \]
LTL+Past and Büchi automata

Finite-state automata are a powerful formalism for defining languages.

Example

\[ G(\text{green} \Rightarrow \text{F red}) \]
LTL+Past and Büchi automata

Finite-state automata are a powerful formalism for defining languages.

Example

\[
G(\text{green} \Rightarrow F \text{red})
\]

What is the relationship between automata (on words) and (linear-time) temporal logics?
Büchi automata

**Definition**

A Büchi automaton is a 5-tuple $\mathcal{B} = \langle Q, Q_0, \Sigma, \rightarrow, F \rangle$ where

- $Q$ is the set of states (or locations) of the automaton,
- $Q_0 \subseteq Q$ is the set of initial states,
- $\Sigma$ is the alphabet,
- $\rightarrow \subseteq Q \times \Sigma \times Q$ is the transition relation,
- $F \subseteq Q$ is the set of repeated states
Büchi automata

Definition
A Büchi automaton is a 5-tuple $\mathcal{B} = \langle Q, Q_0, \Sigma, \rightarrow, F \rangle$ where
- $Q$ is the set of states (or locations) of the automaton,
- $Q_0 \subseteq Q$ is the set of initial states,
- $\Sigma$ is the alphabet,
- $\rightarrow \subseteq Q \times \Sigma \times Q$ is the transition relation,
- $F \subseteq Q$ is the set of repeated states.

Example

- $Q = \{q_0, q_1\}$, $Q_0 = \{q_0\}$,
- $\Sigma = \{\text{green, red}\}$,
- $\rightarrow = \{(q_0, \text{green}, q_1), (q_1, \text{green}, q_1), (q_1, \text{red}, q_0), (q_0, \text{red}, q_0)\}$,
- $F = \{q_0\}$. 

Diagram:

- States: $q_0$, $q_1$
- Transitions: $q_0 \xrightarrow{\text{green}} q_1$, $q_1 \xrightarrow{\text{green}} q_1$, $q_1 \xrightarrow{\text{red}} q_0$, $q_0 \xrightarrow{\text{red}} q_0$
Definition

An (infinite) word $w_0 w_1 \ldots$ is \textit{accepted} by a Büchi automaton $B$ if there exists an infinite sequence $\pi = (\ell_0, \ell_1, \ldots)$ of states s.t.:

- $\ell_0 \in Q_0$,
- for each $i$, $(\ell_i, w_i, \ell_{i+1}) \in \rightarrow$;
- at least one state in $F$ occurs infinitely often in $\pi$. 
Büchi automata

Definition

An (infinite) word $w_0 w_1 \ldots$ is *accepted* by a Büchi automaton $B$ if there exists an infinite sequence $\pi = (\ell_0, \ell_1, \ldots)$ of states s.t.:

- $\ell_0 \in Q_0$,
- for each $i$, $(\ell_i, w_i, \ell_{i+1}) \in \rightarrow$;
- at least one state in $F$ occurs infinitely often in $\pi$. 


Büchi automata

**Definition**

An (infinite) word $w_0 w_1 \ldots$ is *accepted* by a Büchi automaton $B$ if there exists an infinite sequence $\pi = (\ell_0, \ell_1, \ldots)$ of states s.t.:

- $\ell_0 \in Q_0$,
- for each $i$, $(\ell_i, w_i, \ell_{i+1}) \in \rightarrow$;
- at least one state in $F$ occurs infinitely often in $\pi$.

We write $\mathcal{L}(B)$ for the set of words accepted by $B$. 
Büchi automata

**Definition**

An (infinite) word \( w_0 \, w_1 \, \ldots \) is accepted by a Büchi automaton \( \mathcal{B} \) if there exists an infinite sequence \( \pi = (\ell_0, \ell_1, \ldots) \) of states s.t.:

- \( \ell_0 \in Q_0 \),
- for each \( i \), \( (\ell_i, w_i, \ell_{i+1}) \in \rightarrow \);
- at least one state in \( F \) occurs infinitely often in \( \pi \).

We write \( \mathcal{L}(\mathcal{B}) \) for the set of words accepted by \( \mathcal{B} \).

**Example**

\[ \text{green} \cdot \text{red}^\omega \in \mathcal{L}(\mathcal{B}), \]
\[ \text{green} \cdot \text{red} \cdot \text{green}^\omega \notin \mathcal{L}(\mathcal{B}). \]
Outline of today’s lecture

4 LTL+Past and the $\mu$-calculus

5 LTL+Past and Büchi automata
   - Büchi automata
   - From LTL+Past to Büchi automata
     - Büchi automata are more expressive
     - Alternating Büchi automata
     - Application: Succinctness of LTL+Past

6 Stuttering
   - The stuttering principle
   - The generalized stuttering principle

7 Ehrenfeucht-Fraïssé games
   - The rules of the game
   - EF games and the Until-Since hierarchy
From LTL+Past to Büchi automata

**Theorem (Lichtenstein, Pnueli, Zuck, 1985)**

Let $\varphi$ a formula in LTL+Past. There exists a Büchi automaton $B_\varphi$ s.t.

$$\forall w \in (2^{AP})^\omega. \quad w \in \mathcal{L}(B_\varphi) \iff w, 0 \models \varphi.$$  

**Sketch of proof.**

- each state of the automaton corresponds to a set of subformulas of $\varphi$ (and negations thereof),
- if a word $w$ is accepted from a location $q_0$, then any subformula represented by that state holds initially along $w$. 

From LTL+Past to Büchi automata

Theorem (Lichtenstein, Pnueli, Zuck, 1985)

Let \( \varphi \) a formula in LTL+Past. There exists a Büchi automaton \( B_\varphi \) s.t.

\[
\forall w \in (2^{AP})^\omega. \quad w \in \mathcal{L}(B_\varphi) \iff w, 0 \models \varphi.
\]

Sketch of proof.

- each state of the automaton corresponds to a set of subformulas of \( \varphi \) (and negations thereof),

- if a word \( w \) is accepted from a location \( q_0 \), then any subformula represented by that state holds initially along \( w \).
Theorem (Lichtenstein, Pnueli, Zuck, 1985)

Let $\varphi$ a formula in LTL+Past. There exists a Büchi automaton $B_{\varphi}$ s.t.

$$\forall w \in (2^{AP})^\omega. \quad w \in \mathcal{L}(B_{\varphi}) \iff w, 0 \models \varphi.$$ 

Sketch of proof.

- each state of the automaton corresponds to a set of subformulas of $\varphi$ (and negations thereof),
- if a word $w$ is accepted from a location $q_0$, then any subformula represented by that state holds initially along $w$. 

From LTL+Past to Büchi automata

Definition

The closure of \( \varphi \), denoted by \( \text{Cl}(\varphi) \), is the smallest set of formulas containing \( \varphi \) and closed under the following rules:

- \( \top \) and \( \bot \) are in \( \text{Cl}(\varphi) \),
- \( \neg \psi \in \text{Cl}(\varphi) \) iff \( \psi \in \text{Cl}(\varphi) \) (identifying \( \neg \neg \psi \) with \( \psi \)),
- if \( \psi_1 \land \psi_2 \) or \( \psi_1 \lor \psi_2 \) is in \( \text{Cl}(\varphi) \), then \( \psi_1 \in \text{Cl}(\varphi) \) and \( \psi_2 \in \text{Cl}(\varphi) \),
- if \( X \psi_1 \) is in \( \text{Cl}(\varphi) \), then so \( \psi_1 \),
- if \( \psi_1 U \psi_2 \) is in \( \text{Cl}(\varphi) \), then so are \( \psi_1, \psi_2 \), and \( X(\psi_1 U \psi_2) \),
- if \( X^{-1} \psi_1 \) is in \( \text{Cl}(\varphi) \), then so \( \psi_1 \),
- if \( \psi_1 S \psi_2 \) is in \( \text{Cl}(\varphi) \), then so are \( \psi_1, \psi_2 \), and \( X^{-1}(\psi_1 S \psi_2) \).
Proposition

The size of $\text{Cl}(\varphi)$ is at most $4|\varphi|$. 
Proposition

The size of $\text{Cl}(\varphi)$ is at most $4|\varphi|$.

Proof.
By induction of the structure of $\varphi$:

- clear if $\varphi$ is an atomic formula,
**Proposition**

The size of $\text{Cl}(\varphi)$ is at most $4|\varphi|$. 

**Proof.**

By induction of the structure of $\varphi$:

- clear if $\varphi$ is an atomic formula,
- if $\varphi = \psi_1 \land \psi_2$ or $\varphi = \psi_1 \lor \psi_2$, then

\[
\text{Cl}(\varphi) = \text{Cl}(\psi_1) \cup \text{Cl}(\psi_2) \cup \{\varphi, \neg \varphi\}.
\]
Proposition

The size of $\text{Cl}(\varphi)$ is at most $4|\varphi|$. 

Proof.
By induction of the structure of $\varphi$:

- clear if $\varphi$ is an atomic formula,
- if $\varphi = \psi_1 \land \psi_2$ or $\varphi = \psi_1 \lor \psi_2$, then
  
  $$\text{Cl}(\varphi) = \text{Cl}(\psi_1) \cup \text{Cl}(\psi_2) \cup \{\varphi, \neg \varphi\}.$$ 

- if $\varphi = \psi_1 \mathbf{U} \psi_2$, then
  
  $$\text{Cl}(\varphi) = \text{Cl}(\psi_1) \cup \text{Cl}(\psi_2) \cup \{\varphi, \neg \varphi, \mathbf{X} \varphi, \neg \mathbf{X} \varphi\}.$$
From LTL+Past to Büchi automata

**Proposition**

The size of $\text{Cl}(\varphi)$ is at most $4|\varphi|$.

**Proof.**

By induction of the structure of $\varphi$:

- clear if $\varphi$ is an atomic formula,
- if $\varphi = \psi_1 \land \psi_2$ or $\varphi = \psi_1 \lor \psi_2$, then
  \[ \text{Cl}(\varphi) = \text{Cl}(\psi_1) \cup \text{Cl}(\psi_2) \cup \{\varphi, \neg \varphi\}. \]
- if $\varphi = \psi_1 \mathbf{U} \psi_2$, then
  \[ \text{Cl}(\varphi) = \text{Cl}(\psi_1) \cup \text{Cl}(\psi_2) \cup \{\varphi, \neg \varphi, X \varphi, \neg X \varphi\}. \]
- the other cases are similar.
From LTL+Past to Büchi automata

Example

Consider formula $\varphi = G(\text{green} \Rightarrow (F \text{red} \lor G^{-1}\text{green}))$. Then:

$$\text{Cl}(\varphi) = \{\varphi, \neg \varphi, \text{green} \Rightarrow (F \text{red} \lor G^{-1}\text{green}), \neg(\text{green} \Rightarrow (F \text{red} \lor G^{-1}\text{green})), F \text{red} \lor G^{-1}\text{green}, \neg(F \text{red} \lor G^{-1}\text{green}), F \text{red}, \neg F \text{red}, X F \text{red}, \neg X F \text{red}, G^{-1}\text{green}, \neg G^{-1}\text{green}, X^{-1} G^{-1}\text{green}, \neg X^{-1} G^{-1}\text{green}, \text{green}, \neg \text{green}, \text{red}, \neg \text{red}, \top, \bot\}.$$
From LTL+Past to Büchi automata

**Definition**

A subset $S$ of $\text{Cl}(\varphi)$ is *maximal consistent* if:

- $\top \in S$,
- for any $\psi \in \text{Cl}(\varphi)$, $\psi \in S$ iff $\neg \psi \notin S$,
- for any $\psi = \psi_1 \land \psi_2 \in \text{Cl}(\varphi)$: $\psi \in S$ iff $\psi_1 \in S$ and $\psi_2 \in S$,
- for any $\psi = \psi_1 \lor \psi_2 \in \text{Cl}(\varphi)$: $\psi \in S$ iff $\psi_1 \in S$ or $\psi_2 \in S$,
- for any $\psi = \psi_1 \mathsf{U} \psi_2 \in \text{Cl}(\varphi)$:
  - $\psi \in S$ iff $\psi_2 \in S$, or both $\psi_1$ and $\mathsf{X}(\psi_1 \mathsf{U} \psi_2)$ are in $S$,
- for any $\psi = \psi_1 \mathsf{S} \psi_2 \in \text{Cl}(\varphi)$:
  - $\psi \in S$ iff $\psi_2 \in S$, or both $\psi_1$ and $\mathsf{X}^{-1}(\psi_1 \mathsf{S} \psi_2)$ are in $S$. 
From LTL+Past to Büchi automata

Example

The set

\[
\{ \varphi, \neg (\text{green} \Rightarrow (F \text{ red} \lor G^{-1} \text{ green})), \\
\neg (F \text{ red} \lor G^{-1} \text{ green}), \\
\neg F \text{ red}, \neg X F \text{ red}, \neg G^{-1} \text{ green}, \neg X^{-1} G^{-1} \text{ green}, \\
\text{green}, \neg \text{red} \}
\]

is maximal consistent.
Example

The set

\[
\{ \varphi, \neg (\text{green} \Rightarrow (F \text{ red } \lor G^{-1} \text{ green})), \\
\neg (F \text{ red } \lor G^{-1} \text{ green}), \\
\neg F \text{ red}, \neg X F \text{ red}, \neg G^{-1} \text{ green}, \neg X^{-1} G^{-1} \text{ green}, \\
\text{green}, \neg \text{red} \}
\]

is maximal consistent.

Proposition

There are at most \(2^{4|\varphi|}\) maximal consistent subsets of \(\text{Cl}(\varphi)\).
From LTL+Past to Büchi automata

Example

The set

\[
\{ \varphi, \neg (\text{green} \Rightarrow (F \text{ red} \lor G^{-1} \text{ green})), \\
\neg (F \text{ red} \lor G^{-1} \text{ green}), \\
\neg F \text{ red}, \neg X F \text{ red}, \neg G^{-1} \text{ green}, \neg X^{-1} G^{-1} \text{ green}, \\
\text{green}, \neg \text{red} \}
\]

is maximal consistent.

Proposition

There are at most \(2^{4|\varphi|}\) maximal consistent subsets of \(\text{Cl}(\varphi)\).

Maximal consistent subsets are the states of our Büchi automaton.
From LTL+Past to Büchi automata

Given two maximal consistent subsets $S$ and $T$ of $\text{Cl}(\varphi)$, and a “letter” $\sigma \subseteq \text{AP}$, there is a transition $(S, \sigma, T)$ iff:

- for any $p \in \text{AP}$, we have $p \in S$ iff $p \in \sigma$,
- for any subformula $X \varphi_1 \in \text{Cl}(\varphi)$:
  - $X \varphi_1$ is in $S$ iff $\varphi_1 \in T$,
- for any subformula $X^{-1} \varphi_1 \in \text{Cl}(\varphi)$:
  - $\varphi_1$ is in $S$ iff $X^{-1} \varphi_1 \in T$. 

Example

$S$: ...
- $\neg X \color{red}{F}$ red
- $\neg \color{green}{G}^{-1}$
- $\color{green}{G}$
- $\neg \color{red}{red}$

$T$: ...
- $\neg \color{red}{F}$
- $\neg X \color{red}{X^{-1}} \color{green}{G}^{-1}$
- $\neg \color{green}{red}$
Given two maximal consistent subsets $S$ and $T$ of $\text{Cl}(\varphi)$, and a "letter" $\sigma \subseteq \text{AP}$, there is a transition $(S, \sigma, T)$ iff:

- for any $p \in \text{AP}$, we have $p \in S$ iff $p \in \sigma$,
- for any subformula $X \varphi_1 \in \text{Cl}(\varphi)$:
  $X \varphi_1$ is in $S$ iff $\varphi_1 \in T$,
- for any subformula $X^{-1} \varphi_1 \in \text{Cl}(\varphi)$:
  $\varphi_1$ is in $S$ iff $X^{-1} \varphi_1 \in T$.

**Example**

$$
\begin{align*}
S: & \quad \ldots \\
& \quad \neg X F \text{ red} \\
& \quad \neg G^{-1} \text{ green} \\
& \quad \text{green} \\
\end{align*}
$$

$$
\begin{align*}
T: & \quad \ldots \\
& \quad \neg F \text{ red} \\
& \quad \neg G^{-1} \text{ green} \\
& \quad \neg X^{-1} \text{ green} \\
& \quad \neg \text{red} \\
\end{align*}
$$
From LTL+Past to Büchi automata

Example

For formula $G(\text{green} \Rightarrow (F \text{red} \lor G^{-1}\text{green}))$, we get (only “useful” states are displayed):
From LTL+Past to Büchi automata

We use (generalized) Büchi acceptance condition is used to enforce that eventualities eventually occur:

- For each subformula $\psi = \varphi_1 \mathbf{U} \varphi_2$, we write
  \[ F_\psi = \{ l \in Q \mid \varphi_2 \in l \text{ or } \psi \in l \} \]

- a word is accepted if it has a trajectory whose repeated states intersect $F_\psi$ for each $\mathbf{U}$-subformula $\psi$. 
From LTL+Past to Büchi automata

We use (generalized) Büchi acceptance condition is used to enforce that eventualities eventually occur:

- For each subformula $\psi = \varphi_1 \mathbin{U} \varphi_2$, we write

$$F_{\psi} = \{ l \in Q \mid \varphi_2 \in l \text{ or } \psi \in l \}$$

- a word is accepted if it has a trajectory whose repeated states intersect $F_{\psi}$ for each $\mathbin{U}$-subformula $\psi$.

- initial states are those where all $\mathcal{X}^{-1}$-subformulas are false.
Example

For formula $\text{G}(\text{green} \Rightarrow (\text{F red} \lor \text{G}^{-1} \text{green}))$, we get:
From LTL+Past to Büchi automata

**Theorem**

For any LTL+Past formula \( \varphi \), there exists a Büchi automaton \( A \) s.t.

- a word is accepted by \( A \) iff it satisfies \( \varphi \);
- \( A \) has at most \( 2^{4|\varphi|} \) states.

This result is extremely important in computer science: it provides a nice way of verifying that an automaton satisfies an LTL+Past formula. In particular:

**Theorem**

Satisfiability of an LTL+Past formula is PSPACE-complete.
Outline of today’s lecture

4 LTL+Past and the $\mu$-calculus

5 LTL+Past and Büchi automata
   - Büchi automata
   - From LTL+Past to Büchi automata
   - Büchi automata are more expressive
   - Alternating Büchi automata
   - Application: Succinctness of LTL+Past

6 Stuttering
   - The stuttering principle
   - The generalized stuttering principle

7 Ehrenfeucht-Fraïssé games
   - The rules of the game
   - EF games and the Until-Since hierarchy
Büchi automata are more expressive

**Theorem**

*Büchi automata are strictly more expressive than LTL+Past.*
Büchi automata are more expressive

**Theorem**

*Büchi automata are strictly more expressive than LTL+Past.*

*Proof.*

The following Büchi automaton accepts words where *green* holds (at least) at even positions:

![Büchi automaton diagram]

- **States:** `q0`, `q1`  
- **Initial state:** `q0`  
- **Final state:** `q0`  
- **Accepting condition:** At least one even position where *green* holds

- **Transition:** `q0` to `q1` on `green`  
- **Transition:** `q1` to `q0` on `*`
Outline of today’s lecture

4 LTL+Past and the $\mu$-calculus

5 LTL+Past and Büchi automata
   ● Büchi automata
   ● From LTL+Past to Büchi automata
   ● Büchi automata are more expressive
   ● Alternating Büchi automata
   ● Application: Succinctness of LTL+Past

6 Stuttering
   ● The stuttering principle
   ● The generalized stuttering principle

7 Ehrenfeucht-Fraïssé games
   ● The rules of the game
   ● EF games and the Until-Since hierarchy
Alternating Büchi automata

**Alternation**

Alternating automata are automata in which non-deterministic “choices” can be both disjunctive (as usual) and conjunctive.
Alternating Büchi automata

Alternation

Alternating automata are automata in which non-deterministic “choices” can be both disjunctive (as usual) and conjunctive.

Example
Alternating Büchi automata

Alternation

Alternating automata are automata in which non-deterministic “choices” can be both disjunctive (as usual) and conjunctive.

Example

![Diagram of an alternating Büchi automaton with states 1, 2, 3, and 4, and transitions labeled with red, green, and blue colors. States 1 and 2 are connected with a green transition, states 2 and 3 with a red transition, states 3 and 4 with a blue transition, and states 4 and 1 with a blue transition.]
Alternating Büchi automata

Alternation

Alternating automata are automata in which non-deterministic “choices” can be both disjunctive (as usual) and conjunctive.

Example
Alternating Büchi automata

Alternation

Alternating automata are automata in which non-deterministic “choices” can be both disjunctive (as usual) and conjunctive.

Example

\[
\begin{array}{c}
1 \quad \text{green} \\
\downarrow \quad \text{red} \\
3 \\
\downarrow \\
\ast
\end{array}
\quad \begin{array}{c}
2 \quad \ast \\
\downarrow \quad \text{blue} \\
4 \\
\downarrow \quad \text{blue}
\end{array}
\quad \begin{array}{c}
1 \\
\downarrow \\
1 \\
\downarrow \\
2
\end{array}
\]

\quad 

\text{green}
Alternating Büchi automata

Alternation

Alternating automata are automata in which non-deterministic “choices” can be both disjunctive (as usual) and conjunctive.

Example
Alternating Büchi automata

Alternation

Alternating automata are automata in which non-deterministic “choices” can be both disjunctive (as usual) and conjunctive.

Example

![Diagram of an alternating Büchi automaton]
Alternating Büchi automata

Alternation

Alternating automata are automata in which non-deterministic “choices” can be both disjunctive (as usual) and conjunctive.

Example
Alternating Büchi automata

**Alternation**

Alternating automata are automata in which non-deterministic “choices” can be both disjunctive (as usual) and conjunctive.

**Example**

![Diagram of a Büchi automaton with transitions labeled red, green, and blue.](attachment:diagram.jpg)

![Diagram of a tree with labeled nodes green and red.](attachment:tree.jpg)
Alternating Büchi automata

**Alternation**

Alternating automata are automata in which non-deterministic “choices” can be both disjunctive (as usual) and conjunctive.

**Example**

![Diagram of an alternating Büchi automaton with states 1, 2, 3, 4, and transitions labeled with colors (green, red, blue).]
Alternating Büchi automata

Alternation

Alternating automata are automata in which non-deterministic “choices” can be both disjunctive (as usual) and conjunctive.

Example
Alternating Büchi automata

**Alternation**

Alternating automata are automata in which non-deterministic “choices” can be both disjunctive (as usual) and conjunctive.

**Example**

![Diagram of an alternating Büchi automaton with states 1, 2, 3, and 4, and transitions labeled with colors (green, red, blue).]
Alternating Büchi automata

**Alternation**

Alternating automata are automata in which non-deterministic "choices" can be both disjunctive (as usual) and conjunctive.

**Example**

The diagram on the left illustrates an alternating automaton where states 1 and 2 can choose between green and red, and states 3 and 4 can choose between blue and green. The tree on the right shows the possible paths and choices in the automaton.
Alternating Büchi automata

Definition

An alternating automaton is $1$-weak if there exists a total order on its set of states such that transitions are always “decreasing”.

1-weak
Alternating Büchi automata

Definition

An alternating automaton is \(1\)-weak if there exists a total order on its set of states such that transitions are always “decreasing”.

Example

```
1 -- red --> 4 -- green --> 3 -- blue --> 2
```

\((\text{green} \land \mathcal{F}\text{blue}) \cup \text{red}\)
Alternating Büchi automata

**Definition**

An alternating automaton is **1-weak** if there exists a total order on its set of states such that transitions are always "decreasing".

**Example**

\[(\text{green} \land \neg \text{blue}) \cup \text{red}\]
Alternating Büchi automata

Theorem (Vardi, 1994)

Any LTL formula $\varphi$ can be transformed into a 1-weak alternating Büchi automaton $B_\varphi$ s.t.

- a word is accepted by $B_\varphi$ iff it satisfies $\varphi$,
- $B_\varphi$ has at most $|\varphi|$ states.
## Alternating Büchi automata

### Theorem (Vardi, 1994)

Any LTL formula $\varphi$ can be transformed into a 1-weak alternating Büchi automaton $B_\varphi$ s.t.

- a word is accepted by $B_\varphi$ iff it satisfies $\varphi$,
- $B_\varphi$ has at most $|\varphi|$ states.

Conversely:

### Theorem (Rohde, 1997)

A 1-weak alternating Büchi automaton $B$ can be transformed into an LTL formula $\varphi_B$ s.t.

- a word is accepted by $\varphi$ iff it satisfies $\varphi_B$,
- $\varphi_B$ has size exponential in the size of $B$. 
Outline of today’s lecture

4 LTL+Past and the $\mu$-calculus

5 LTL+Past and Büchi automata
   - Büchi automata
   - From LTL+Past to Büchi automata
   - Büchi automata are more expressive
   - Alternating Büchi automata
   - Application: Succinctness of LTL+Past

6 Stuttering
   - The stuttering principle
   - The generalized stuttering principle

7 Ehrenfeucht-Fraïssé games
   - The rules of the game
   - EF games and the Until-Since hierarchy
Succinctness of LTL+Past

Consider the following property, built on $\text{AP} = \{p_0, \ldots, p_n\}$:

$(\mathcal{P})$: any two states that agree on propositions $p_1$ to $p_n$ also agree on proposition $p_0$. 

Succinctness of LTL+Past

Consider the following property, built on $\text{AP} = \{p_0, \ldots, p_n\}$:

$(\mathcal{P})$: any two states that agree on propositions $p_1$ to $p_n$ also agree on proposition $p_0$.

It can be expressed in LTL by enumerating the possible valuations for $p_0$ to $p_n$:

$$\bigwedge_{(b_0, \ldots, b_n) \in \{\top, \bot\}^{n+1}} \left( F\left( \bigwedge_{i \geq 0} p_i = b_i \right) \Rightarrow G\left( \bigwedge_{i \geq 1} p_i = b_i \right) \Rightarrow p_0 = b_0 \right)$$

The size of this formula is exponential in $n$.
Succinctness of $\mathsf{LTL}+$Past

$(\mathcal{P})$: any two states that agree on propositions $p_1$ to $p_n$ also agree on proposition $p_0$.

Let $\mathcal{A}$ be a Büchi automaton corresponding to property $(\mathcal{P})$.

Let $\Sigma = \{a_0, a_1, \ldots, a_{2^n-1}\}$ be the subsets of $\{p_1, \ldots, p_n\}$.
Succinctness of LTL+Past

(\mathcal{P}): any two states that agree on propositions \( p_1 \) to \( p_n \) also agree on proposition \( p_0 \).

For each \( K \subseteq \{0, \ldots, 2^n - 1\} \), we define \( w_K = b_0 \ldots b_{2^n-1} \) with

\[
b_i = \begin{cases} 
a_i & \text{if } i \in K \\
a_i \cup \{p_0\} & \text{otherwise}
\end{cases}
\]
Succinctness of LTL\textsubscript{+}Past

\((\mathcal{P})\): any two states that agree on propositions \(p_1\) to \(p_n\) also agree on proposition \(p_0\).

For each \(K \subseteq \{0, \ldots, 2^n - 1\}\), we define \(w_K = b_0 \cdots b_{2^n-1}\) with

\[
b_i = \begin{cases} a_i & \text{if } i \in K \\ a_i \cup \{p_0\} & \text{otherwise} \end{cases}
\]

Lemma

There are \(2^{2^n}\) different such words.
Succinctness of $\text{LTL}+\text{Past}$

$(\mathcal{P})$: any two states that agree on propositions $p_1$ to $p_n$ also agree on proposition $p_0$.

For each $K \subseteq \{0, \ldots, 2^n - 1\}$, we define $w_K = b_0 \ldots b_{2^n-1}$ with

$$b_i = \begin{cases} a_i & \text{if } i \in K \\ a_i \cup \{p_0\} & \text{otherwise} \end{cases}$$

**Lemma**

For any $K \subseteq \{0, \ldots, 2^n - 1\}$, the word $w_K^\omega$ is accepted by $\mathcal{A}$. 
Succinctness of LTL+Past

(P): any two states that agree on propositions $p_1$ to $p_n$ also agree on proposition $p_0$.

For each $K \subseteq \{0, \ldots, 2^n - 1\}$, we define $w_K = b_0 \ldots b_{2^n - 1}$ with

$$b_i = \begin{cases} a_i & \text{if } i \in K \\ a_i \cup \{p_0\} & \text{otherwise} \end{cases}$$

Lemma

For any $K \subseteq \{0, \ldots, 2^n - 1\}$, the word $w_K^\omega$ is accepted by $A$.

Lemma

For any $K \neq K'$, the word $w_{K'} \cdot w_K^\omega$ is not accepted by $A$. 
Succinctness of LTL+Past

(P): any two states that agree on propositions $p_1$ to $p_n$ also agree on proposition $p_0$.

Lemma
For any $K \subseteq \{0, ..., 2^n - 1\}$, the word $w_K^\omega$ is accepted by $A$.

Lemma
For any $K \neq K'$, the word $w_{K'} \cdot w_K^\omega$ is not accepted by $A$.

For any $K \neq K'$, the states reached after reading $w_K$ and after reading $w_{K'}$ must be different.
Succinctness of LTL+Past

\((\mathcal{P})\): any two states that agree on propositions \(p_1\) to \(p_n\) also agree on proposition \(p_0\).

**Lemma**

For any \(K \subseteq \{0, ..., 2^n - 1\}\), the word \(w^K_\omega\) is accepted by \(A\).

**Lemma**

For any \(K \neq K'\), the word \(w^{K'} \cdot w^K_\omega\) is not accepted by \(A\).

For any \(K \neq K'\), the states reached after reading \(w_K\) and after reading \(w_{K'}\) must be different.

**Theorem**

Any Büchi automaton \(A\) characterizing property \((\mathcal{P})\) has at least \(2^{2^n}\) states.
Succinctness of LTL+Past

(P): any two states that agree on propositions $p_1$ to $p_n$ also agree on proposition $p_0$.

Theorem

Any Büchi automaton $A$ characterizing property (P) has at least $2^{2^n}$ states.

Corollary

Any LTL formula expressing property (P) has size at least $2^{n-1}$.
Succinctness of LTL+Past

Consider now the following property, slightly different:

\((P')\): any state that agrees on propositions \(p_1\) to \(p_n\) with the initial state also agrees on proposition \(p_0\).
Succinctness of LTL+Past

Consider now the following property, slightly different:

\((\mathcal{P'})\): any state that agrees on propositions \(p_1\) to \(p_n\) with the initial state also agrees on proposition \(p_0\).

This can be expressed in LTL+Past by the following (polynomial-size) formula:

\[
G \left( \bigwedge_{i \geq 1} p_i \iff F^{-1} G^{-1} p_i \right) \Rightarrow \left( p_0 \iff F^{-1} G^{-1} p_0 \right).
\]
Succinctness of LTL+Past

Consider now the following property, slightly different:

\((\mathcal{P}')\): any state that agrees on propositions \(p_1\) to \(p_n\) with the initial state also agrees on proposition \(p_0\).

This can be expressed in LTL+Past by the following (polynomial-size) formula:

\[
\begin{align*}
\mathbf{G}\left( \bigwedge_{i \geq 1} p_i \iff \mathbf{F}^{-1} \mathbf{G}^{-1} p_i \right) \Rightarrow (p_0 \iff \mathbf{F}^{-1} \mathbf{G}^{-1} p_0) \end{align*}
\]

Let \(\phi\) be an LTL formula expressing property \((\mathcal{P}')\). Then \(\mathbf{G} \phi\) precisely expresses property \((\mathcal{P})\), and thus has size at least \(2^{n-1}\). 
Outline of today’s lecture

4 LTL+Past and the $\mu$-calculus

5 LTL+Past and Büchi automata
   - Büchi automata
   - From LTL+Past to Büchi automata
   - Büchi automata are more expressive
   - Alternating Büchi automata
   - Application: Succinctness of LTL+Past

6 Stuttering
   - The stuttering principle
   - The generalized stuttering principle

7 Ehrenfeucht-Fraïssé games
   - The rules of the game
   - EF games and the Until-Since hierarchy
The stuttering principle

Definition
Let $w = w_0 w_1 \ldots$ be a word over AP. A letter $a = w_i$ is **stuttering** in $w$ if it appears several consecutive times, i.e., if $a = w_i = w_{i+1}$.
The stuttering principle

**Definition**

Let $w = w_0 w_1 \ldots$ be a word over AP. A letter $a = w_i$ is **stuttering** in $w$ if it appears several consecutive times, i.e., if $a = w_i = w_{i+1}$.

**Example**

\[
w = g \cdot r \cdot g \cdot g \cdot g \cdot \underbrace{g \cdot b \cdot r \ldots}_{\text{stuttering}}
\]
The stuttering principle

Definition
Let \( w = w_0 w_1 \ldots \) be a word over AP. A letter \( a = w_i \) is stuttering in \( w \) if it appears several consecutive times, i.e., if \( a = w_i = w_{i+1} \).

Definition
The relation \( \preceq \) is defined as follows:

\[ w \preceq w' \iff w \text{ is obtained from } w' \text{ by removing one copy of a stuttering letter.} \]
The stuttering principle

**Definition**
Let \( w = w_0 \ldots \) be a word over \( AP \). A letter \( a = w_i \) is *stuttering* in \( w \) if it appears several consecutive times, i.e., if \( a = w_i = w_{i+1} \).

**Definition**
The relation \( \preceq \) is defined as follows:

\[
w \preceq w' \iff w \text{ is obtained from } w' \text{ by removing one copy of a stuttering letter}.
\]

**Example**

\[
g \cdot r \cdot g \cdot g \cdot g \cdot b \cdot r \ldots \preceq g \cdot r \cdot g \cdot g \cdot g \cdot g \cdot b \cdot r \ldots
\]
**The stuttering principle**

**Definition**

*Stuttering equivalence* is the least equivalence relation that subsumes $\ll$.
The stuttering principle

**Definition**

*Stuttering equivalence* is the least equivalence relation that subsumes $\preccurlyeq$.

**Example**

The words $g \cdot (b \cdot b \cdot r)^\omega$ and $g \cdot (b \cdot r \cdot r)^\omega$ are stuttering equivalent.
The stuttering principle

**Definition**

*Stuttering equivalence* is the least equivalence relation that subsumes $\preceq$.

**Theorem**

*Two words are stuttering-equivalent iff they cannot be distinguished by any formula of $\mathcal{L}(U)$.*
The stuttering principle

<table>
<thead>
<tr>
<th>Definition</th>
<th><strong>Stuttering equivalence</strong> is the least equivalence relation that subsumes $\subseteq$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theorem</td>
<td>Two words are stuttering-equivalent iff they cannot be distinguished by any formula of $L(U)$.</td>
</tr>
<tr>
<td>Corollary</td>
<td>$L(U, X)$ has strictly more distinguishing power than $L(U)$.</td>
</tr>
</tbody>
</table>
The generalized stuttering principle

**Definition**

A subword $w[i, j]$ of a word $w$ is $(m, n)$-redundant if the subword $w[i + j, mj - m + 1 + n]$ is a prefix of $w[i, j]^\omega$. 
The generalized stuttering principle

**Definition**

A subword \( w[i, j] \) of a word \( w \) is \((m, n)\)-redundant if the subword \( w[i + j, mj - m + 1 + n] \) is a prefix of \( w[i, j]^{\omega} \).

**Example**

\[ r \ r \ g \ g \ r \ g \ r \ r \ g \ r \ r \ g \ r \ r \ g \ g \ g \ r \]
The generalized stuttering principle

Definition

A subword $w[i, j]$ of a word $w$ is $(m, n)$-redundant if the subword $w[i + j, mj - m + 1 + n]$ is a prefix of $w[i, j]^\omega$.

Example

$r$ $r$ $g$ $g$ $r$ $g$ $r$ $g$ $r$ $g$ $r$ $g$ $r$ $g$ $r$ $r$ $g$ $r$ $g$ $r$ $g$ $r$ $g$ $g$ $r$
The generalized stuttering principle

**Definition**

A subword $w[i, j]$ of a word $w$ is $(m, n)$-redundant if the subword $w[i + j, mj - m + 1 + n]$ is a prefix of $w[i, j]^\omega$.

**Example**

$$w[5, 3]^\omega = \text{r g r r g r r g r r g r r g r r g r r g r}$$

$$w[5, 3] = \text{r g g g r g r r g r r g r r g g r}$$
The generalized stuttering principle

Definition

A subword $w[i, j]$ of a word $w$ is $(m, n)$-redundant if the subword $w[i + j, mj - m + 1 + n]$ is a prefix of $w[i, j]^{\omega}$.

Example

$w[5, 3]^{\omega} =$

$w[5, 3]$ is $(0, 8)$-redundant, but also $(3, 2)$-redundant.
The generalized stuttering principle

Definition

Given two words $w$ and $w'$, and two integers $m$ and $n$, we define:

$$w \preceq_{m,n} w' \iff w \text{ is obtained from } w' \text{ by removing } (m, n)\text{-redundant subwords.}$$
The generalized stuttering principle

**Definition**

Given two words $w$ and $w'$, and two integers $m$ and $n$, we define:

$$w \preceq_{m,n} w' \iff w \text{ is obtained from } w' \text{ by removing } (m, n)\text{-redundant subwords.}$$

**Definition**

$(m, n)$-stuttering equivalence is the least equivalence relation that subsumes $\preceq_{m,n}$. 
The generalized stuttering principle

Definition
Given two words $w$ and $w'$, and two integers $m$ and $n$, we define:

$$w \preceq_{m,n} w' \iff w \text{ is obtained from } w' \text{ by removing } (m, n)-\text{redundant subwords.}$$

Definition
$(m, n)$-stuttering equivalence is the least equivalence relation that subsumes $\preceq_{m,n}$.

Example

$$rrgg\ rgr\ rgr\ ggr\ ... \preceq_{3,2} rrgg\ rgr\ rgr\ rgr\ rgr\ ggr\ ...$$
The generalized stuttering principle

**Definition**

We write $\mathcal{L}(U^m, X^n)$ for the fragment of LTL where nesting identical modalities is bounded (by $m$ and $n$ for $U$ and $X$, respectively).
The generalized stuttering principle

Example

The following formula is in $\mathcal{L}(U^2, X^4)$:

$$\text{green } U (X(\text{red } U \text{ green } \land X \text{ blue})) \lor X X X X \text{ red}$$
The generalized stuttering principle

Example

The following formula is in $\mathcal{L}(U^2, X^4)$:

$$\text{green } U (X(\text{red } U \text{ green } \land X \text{ blue})) \lor X X X X X \text{ red}$$
The generalized stuttering principle

**Theorem (Kučera, Strejček, 2005)**

*If two words are \((m, n)\)-stuttering equivalent, then they can’t be distinguished by formulas of \(\mathcal{L}(U^m, X^n)\).*
The generalized stuttering principle

<table>
<thead>
<tr>
<th>Theorem (Kučera, Strejček, 2005)</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>If two words are (m, n)-stuttering equivalent, then they can’t be distinguished by formulas of ( \mathcal{L}(U^m, X^n) ).</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corollary</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>For any m and n, there exists formulas in ( \mathcal{L}(U^m, X^n) ) that can be expressed neither in ( \mathcal{L}(U^{m-1}, X) ) nor in ( \mathcal{L}(U^m, X^{n-1}) ).</em></td>
</tr>
</tbody>
</table>
The generalized stuttering principle

**Example**

We illustrate the case where \( m = 3 \) and \( n = 2 \): let

\[
\varphi = F(green \land F(red \land F(blue \land XX \text{green}))).
\]

and

\[
w = \text{brg brg brg y}^\omega
\]

\[
w' = \text{brg brg y}^\omega
\]

Then \( w, 0 \models \varphi \) and \( w', 0 \nvdash \varphi \).

\( w \) and \( w' \) are \((2, 2)\)- and \((3, 1)\)-stutter equivalent, and thus can be distinguished neither by formulas in \( \mathcal{L}(U^2, X^2) \) nor by formulas in \( \mathcal{L}(U^3, X^1) \).
The generalized stuttering principle

**Theorem**

The family $\mathcal{L}(U^m, X^n)$ form a strict hierarchy w.r.t. expressive power.
The generalized stuttering principle

**Theorem**

The family \( \mathcal{L}(U^m, X^n) \) form a strict hierarchy w.r.t. expressive power.

⚠️ It seems natural to conjecture that

if a property can be expressed in \( \mathcal{L}(U^{m+1}, X^n) \) and in \( \mathcal{L}(U^m, X^{n+1}) \), then it can be expressed in \( \mathcal{L}(U^m, X^n) \).

This is **false**: for instance, \( F(green \land F \neg green) \) and \( F(green \land X \neg green) \) are equivalent, but cannot be expressed in \( \mathcal{L}(U^1) \).
Outline of today’s lecture

4 LTL+Past and the $\mu$-calculus

5 LTL+Past and Büchi automata
   - Büchi automata
   - From LTL+Past to Büchi automata
   - Büchi automata are more expressive
   - Alternating Büchi automata
   - Application: Succinctness of LTL+Past

6 Stuttering
   - The stuttering principle
   - The generalized stuttering principle

7 Ehrenfeucht-Fraïssé games
   - The rules of the game
   - EF games and the Until-Since hierarchy
Ehrenfeucht-Fraïssé games

EF games are 2-player games used to show that two linear structures $T$ and $U$ can (or cannot) be distinguished by a logic.
Ehrenfeucht-Fraïssé games

EF games are 2-player games used to show that two linear structures $T$ and $U$ can (or cannot) be distinguished by a logic.

**Definition**

A *configuration* of the game is a couple $\langle t, u \rangle$ where $t \in T$ and $u \in U$. 
Ehrenfeucht-Fraïssé games

EF games are 2-player games used to show that two linear structures $T$ and $U$ can (or cannot) be distinguished by a logic.

**Definition**

A *configuration* of the game is a couple $\langle t, u \rangle$ where $t \in T$ and $u \in U$.

**Definition**

From a configuration $\langle t_0, u_0 \rangle$, the rules of a $k$-round EF game is defined recursively as follows:

- when $k = 0$, player $A$ wins if $t_0$ and $u_0$ are labeled by exactly the same atomic propositions;
- when $k \geq 1$, two cases may arise:
  - if $t_0$ and $u_0$ are not labeled by the same atomic propositions, then the game stops and player $A$ is declared the winner;
  - otherwise, player $A$ plays an $U$-move or a $S$-move.
Ehrenfeucht-Fraïssé games

EF games are 2-player games used to show that two linear structures $T$ and $U$ can (or cannot) be distinguished by a logic.

**Definition**

An $U$-move from configuration $\langle t, u \rangle$ is played as follows:

- player $A$ selects the structure he wants to play on (say $T$), and an element $t'$ of that structure s.t. $t \leq t'$;
- player $B$ responds by choosing an element $u'$ in the other structure s.t. $u \leq u'$;
- player $A$ has then two choices:
  - either he sets the new configuration to $\langle t', u' \rangle$;
  - or he selects a position $u''$ in $U$ s.t. $u \leq u'' < u'$, player $B$ chooses $t''$ in $T$ with $t \leq t'' < t'$, and the new configuration is $\langle t'', u'' \rangle$.

$S$-moves are symmetric.
Ehrenfeucht-Fraïssé games

Example

Initial configuration
Ehrenfeucht-Fraïssé games

Example

Player $\mathcal{A}$ plays an $\mathbf{U}$-move on the first structure.
Ehrenfeucht-Fraïssé games

Example

Player $B$ responds on the other structure.
Ehrenfeucht-Fraïssé games

Example

This is the new configuration.
Ehrenfeucht-Fraïssé games

Example

Player $A$ plays an $U$-move on the first structure.
Ehrenfeucht-Fraïssé games

Example

Player B responds on the other structure...
Ehrenfeucht-Fraïssé games

Example

Player $B$ responds on the other structure... and loses.
Ehrenfeucht-Fraïssé games

Example

Initial configuration
Example

Player \( A \) plays an \( U \)-move on the first structure.
Ehrenfeucht-Fraïssé games

<table>
<thead>
<tr>
<th>Example</th>
<th>Player $B$ responds on the other structure.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

Ehrenfeucht-Fraïssé games

Example

This is the new configuration.
Ehrenfeucht-Fraïssé games

Example

Player $\mathcal{A}$ plays a $S$-move on the second structure.
Ehrenfeucht-Fraïssé games

Example

Player $B$ responds on the other structure.
Ehrenfeucht-Fraïssé games

Example

Player $A$ picks an intermediary position on the first structure.
Ehrenfeucht-Fraïssé games

Example

Player B responds...
Ehrenfeucht-Fraïssé games

Example

Player $B$ responds... and loses.
We write $\mathcal{L}(\{U, S\}^k)$ for the fragment of $\mathcal{L}(U, S)$ where the temporal height is bounded by $k$. 
EF games and the Until-Since hierarchy

**Definition**

We write $\mathcal{L}(\{\mathbf{U}, \mathbf{S}\}^k)$ for the fragment of $\mathcal{L}(\mathbf{U}, \mathbf{S})$ where the temporal height is bounded by $k$.

**Example**

$$\text{green } \mathbf{U} ((\text{green } \mathbf{S} \text{ red}) \mathbf{S} (\text{red } \mathbf{U} \text{ green } \land (\text{green } \mathbf{U} \text{ blue}))))$$
Definition

We write $\mathcal{L}(\{U, S\}^k)$ for the fragment of $\mathcal{L}(U, S)$ where the temporal height is bounded by $k$.

Example

$\text{green } U ((\text{green } S \text{ red}) \ S (\text{red } U \text{ green } \land \ (\text{green } U \text{ blue})))$
EF games and the Until-Since hierarchy

**Definition**

We write $\mathcal{L}(\{\text{U, S}\}^k)$ for the fragment of $\mathcal{L}(\text{U, S})$ where the temporal height is bounded by $k$.

**Example**

```
green \text{U} ((\text{green S red}) \text{S} (\text{red U green} \land (\text{green U blue})))
```

belongs to $\mathcal{L}(\{\text{U, S}\}^4)$
EF games and the Until-Since hierarchy

**Definition**

We write $\mathcal{L}({\{U, S}\}^k)$ for the fragment of $\mathcal{L}(U, S)$ where the temporal height is bounded by $k$.

**Theorem (Etessami & Wilke, 2000)**

*Player $B$ has a winning strategy in the $k$-round game from a configuration $\langle t, u \rangle$ iff, for any formula $\varphi \in \mathcal{L}({\{U, S}\}^k)$, we have*

$$\langle T, t \rangle \models \varphi \iff \langle U, u \rangle \models \varphi.$$
EF games and the Until-Since hierarchy

Now, consider the following two structures:

\[ T_k = \left( (r \ b)^{k-1} \ g \ b \right)^{k-1} (r \ b)^{k} \ g \ b \ ( (r \ b)^{k-1} \ g \ b )^{k-1} y^\omega \]

\[ U_k = \left( (r \ b)^{k-1} \ g \ b \right)^{k-1} (r \ b)^{k-1} \ g \ b \ ( (r \ b)^{k-1} \ g \ b )^{k-1} y^\omega \]
EF games and the Until-Since hierarchy

Now, consider the following two structures:

\[ T_k = ((r \ b)^{k-1} \ g \ b)^{k-1} (r \ b)^{k-1} \ g \ b ((r \ b)^{k-1} \ g \ b)^{k-1} y^\omega \]
\[ U_k = ((r \ b)^{k-1} \ g \ b)^{k-1} (r \ b)^{k-1} \ g \ b ((r \ b)^{k-1} \ g \ b)^{k-1} y^\omega \]

For instance:

\[ T_3 = rbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgby^\omega \]
\[ U_3 = rbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgby^\omega \]
EF games and the Until-Since hierarchy

Now, consider the following two structures:

\[ T_k = ((r \ b)^{k-1} \ g \ b)^{k-1} (r \ b)^k \ g \ b ((r \ b)^{k-1} \ g \ b)^{k-1} y^\omega \]

\[ U_k = ((r \ b)^{k-1} \ g \ b)^{k-1} (r \ b)^{k-1} g \ b ((r \ b)^{k-1} \ g \ b)^{k-1} y^\omega \]

For instance:

\[ T_3 = rbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgby^\omega \]

\[ U_3 = rbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbrbgby^\omega \]

Lemma

Player B has a winning strategy in the k-round game based on \( T_k \) and \( U_k \) from configuration \( \langle 0, 0 \rangle \).
EF games and the Until-Since hierarchy

Now, consider the following two structures:

\[
T_k = ((r b)^{k-1} g b)^{k-1} (r b)^k g b ((r b)^{k-1} g b)^{k-1} y^\omega \\
U_k = ((r b)^{k-1} g b)^{k-1} (r b)^{k-1} g b ((r b)^{k-1} g b)^{k-1} y^\omega
\]

For instance:

\[
T_3 = r br b g b r b r b g b r b r b g b r b r b g b r b r b g b g b y^\omega \\
U_3 = r br b g b r b r b g b r b r b g b r b r b g b r b r b g b r b g b y^\omega
\]
EF games and the Until-Since hierarchy

Now, consider the following two structures:

\[ T_k = (((r \ b)^{k-1} \ g \ b)^{k-1} (r \ b)^k \ g \ b \ ((r \ b)^{k-1} \ g \ b)^{k-1}) y^\omega \]
\[ U_k = (((r \ b)^{k-1} \ g \ b)^{k-1} (r \ b)^{k-1} \ g \ b \ ((r \ b)^{k-1} \ g \ b)^{k-1}) y^\omega \]

For instance:

\[ T_3 = rbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbgby^\omega \]
\[ U_3 = rbrbgbrbrbgbrbrbgbrbrbgbrbrbgbrbgby^\omega \]

**Lemma**

There exists a formula \( \varphi \in \mathcal{L}(\{U, S\}^{k+1}) \) that distinguishes between \( \langle T_k, 0 \rangle \) and \( \langle U_k, 0 \rangle \).

For instance:

\[ \top \ U (b \land (b \ S (r \land r \ S b)) \land (b \ U (r \land r \ U (b \land b \ U \ r)))) \]
**EF games and the Until-Since hierarchy**

**Theorem (Etessami & Wilke, 2000)**

The hierarchy $\mathcal{L}(\{U, S\}^k)$ is strict (w.r.t. distinguishing power).
EF games and the Until-Since hierarchy

**Theorem (Etessami & Wilke, 2000)**

The hierarchy $\mathcal{L}(\{U, S\}^k)$ is strict (w.r.t. distinguishing power).

Similar techniques can be used to prove that:

**Theorem (Etessami & Wilke, 2000)**

The hierarchy $\mathcal{L}(\{U, S\}^k, \{X, F, X^{-1}, F^{-1}\}^r)$ is strict (w.r.t. distinguishing power).
Expressiveness of Temporal Logics

François Laroussinie and Nicolas Markey

Lab. Specification et Verification
ENS Cachan & CNRS, France

August 2, 2006
Branching time

With Branching-time Temporal logics, formulas are interpreted over states of a tree-like structure (or a Kripke structure).

A state may have several successors!

Thus properties may express that a state has a successor satisfying a given formula and another one satisfying another formula.

Example

When specifying reactive system (or a program):

- with linear-time temporal logic, the system is seen as a set of executions.
- with branching-time temporal logic, the system is seen as a Kripke structure.

(here: we only consider discrete time !)
With Branching-time Temporal logics, formulas are interpreted over states of a tree-like structure (or a Kripke structure).

A state may have several successors!

Thus properties may express that a state has a successor satisfying a given formula and another one satisfying another formula.

Example

When specifying reactive system (or a program):

- with linear-time temporal logic, the system is seen as a set of executions.
- with branching-time temporal logic, the system is seen as a Kripke structure.

(here: we only consider discrete time !)
In this lecture, we will consider the following questions:

- Which kind of properties can we express with BT-TL?
- What is the difference between BT-TL and linear-time TL?
- There are a lot of logics in BT... Why?
- What is the relationship of BT TL with automata theory?
Tree-like structure

Definition

A tree is a set $T \subseteq \mathbb{N}^*_0$ such that if $x \cdot c \in T$ with $x \in \mathbb{N}^*_0$ and $c \in \mathbb{N}_0$, we have:

- $x \in T$
- for all $1 \leq c' < c$, $x \cdot c' \in T$

A tree is as a partially ordered set of nodes s.t. the set of predecessors of any node is finite, totally ordered and with a common minimal element (the root $\varepsilon$).

Let $\Sigma$ be an alphabet.

Definition

A $\Sigma$-labeled tree is a pair $\langle T, l \rangle$ where:

- $T$ is a tree
- $l : T \rightarrow \Sigma$ maps each node with a letter in $\Sigma$. 
Tree-like structure

Definition
A tree is a set $T \subseteq \mathbb{N}^*_0$ such that if $x \cdot c \in T$ with $x \in \mathbb{N}^*_0$ and $c \in \mathbb{N}_0$, we have:

- $x \in T$
- for all $1 \leq c' < c$, $x \cdot c' \in T$

A tree is as a partially ordered set of nodes s.t. the set of predecessors of any node is finite, totally ordered and with a common minimal element (the root $\varepsilon$).

Let $\Sigma$ be an alphabet.

Definition
A $\Sigma$-labeled tree is a pair $\langle T, l \rangle$ where:

- $T$ is a tree
- $l : T \rightarrow \Sigma$ maps each node with a letter in $\Sigma$. 
**Tree-like structure**

**Definition**
A tree is a set $T \subseteq \mathbb{N}^*_0$ such that if $x \cdot c \in T$ with $x \in \mathbb{N}^*_0$ and $c \in \mathbb{N}_{>0}$, we have:
- $x \in T$
- for all $1 \leq c' < c$, $x \cdot c' \in T$

A tree is as a partially ordered set of nodes s.t. the set of predecessors of any node is finite, totally ordered and with a common minimal element (the **root** $\varepsilon$).

Let $\Sigma$ be an alphabet.

**Definition**
A $\Sigma$-labeled tree is a pair $\langle T, l \rangle$ where:
- $T$ is a tree
- $l : T \rightarrow \Sigma$ maps each node with a letter in $\Sigma$. 
Let $\text{AP}$ be a set of atomic propositions.

**Definition**

A Kripke structure is a tuple $\mathcal{S} = \langle Q, q_{\text{init}}, R, l \rangle$ where

- $Q$ is a set of states,
- $q_{\text{init}} \in Q$ is the initial state,
- $R \subseteq Q \times Q$ is a total transition relation,
- $l : Q \rightarrow 2^\text{AP}$ labels every state with the propositions it satisfies.

(+$\ $ fairness constraints$)$

($qRq'$ is usually denoted $q \rightarrow q'$)
Branching time

A Kripke structure...

And its unwinding...
Branching time

A Kripke structure...

And its unwinding...
Branching time

A Kripke structure...

And its unwinding...
Branching time

A Kripke structure...

And its unwinding...
Branching time

A Kripke structure...

And its unwinding...

Branching time

A Kripke structure...

And its unwinding...
Definition (Clarke & Emerson, 1981)

\[ \varphi, \psi ::= P_1 | P_2 | \ldots | \neg \varphi | \varphi \land \psi | E X \varphi | A X \varphi \\
| E \varphi U \psi | A \varphi U \psi \]

\( P_1, P_2, \ldots \in AP. \)

Notation: \( \mathcal{B}(X, U) \).

Formulas are interpreted over states of a structure \( S \).

\textbf{Exec}(q)\) denotes the set of (infinite) executions from \( q \).

Given an execution \( \rho = q_0 \rightarrow q_1 \rightarrow q_2 \ldots \), we have:

- \( \rho(i) \) denotes the \( i \)-th state (i.e. \( q_i \)),
- \( \rho|_i \) denotes the prefix \( q_1 \rightarrow \ldots \rightarrow q_i \),
- \( \rho^i \) is the \( i \)-th suffix: \( q_i \rightarrow q_{i+1} \rightarrow \ldots \).
**Computation Tree Logic – CTL**

**Definition (Clarke & Emerson, 1981)**

\[ \varphi, \psi ::= P_1 \mid P_2 \mid \ldots \mid \neg \varphi \mid \varphi \land \psi \mid E X \varphi \mid A X \varphi \mid E \varphi U \psi \mid A \varphi U \psi \]

\(P_1, P_2, \ldots \in AP.\)

Notation: \(B(X, U).\)

Formulas are interpreted over states of a structure \(S.\)

\textbf{Exec}(q)\ denotes the set of (infinite) executions from \(q.\)

Given an execution \(\rho = q_0 \rightarrow q_1 \rightarrow q_2 \ldots,\) we have:

- \(\rho(i)\) denotes the \(i\)-th state (i.e. \(q_i\)),
- \(\rho|_i\) denotes the prefix \(q_1 \rightarrow \ldots \rightarrow q_i,\)
- \(\rho^i\) is the \(i\)-th suffix: \(q_i \rightarrow q_{i+1} \rightarrow \ldots.\)
Computation Tree Logic – \textit{CTL}

\[ S = \langle Q, q_{\text{init}}, R, I \rangle \text{ or } S = (T, I) \]

\textbf{Definition}

These clauses define when a state \( q \) in \( S \) satisfies a formula \( \varphi \), written \( q \models_S \varphi \):

- \( q \models_S P \) iff \( P \in I(q) \)
- \( q \models_S \varphi \land \psi \) iff \( q \models_S \varphi \) and \( q \models_S \psi \)
- \( q \models_S \neg \varphi \) iff \( q \not\models_S \varphi \)

- \( q \models_S \text{EX} \varphi \) iff \( \exists \rho \in \text{Exec}(q) \text{ s.t. } \rho(1) \models_S \varphi \)
- \( q \models_S \text{AX} \varphi \) iff \( \forall \rho \in \text{Exec}(q) \text{ we have } \rho(1) \models_S \varphi \)
- \( q \models_S \text{E} \varphi \text{ U } \psi \) iff \( \exists \rho \in \text{Exec}(q) \text{ s.t. } \exists i \geq 0, \rho(i) \models_S \psi \text{ and } \forall 0 \leq j < i, \text{ we have } \rho(j) \models_S \varphi \)
- \( q \models_S \text{A} \varphi \text{ U } \psi \) iff \( \forall \rho \in \text{Exec}(q), \exists i \geq 0, \text{ s.t. } \rho(i) \models_S \psi \text{ and } \forall 0 \leq j < i, \text{ we have } \rho(j) \models_S \varphi \)
Abbreviations

$\top, \bot, \varphi \lor \psi, \varphi \Rightarrow \psi \ldots$

- $\textbf{EF} \varphi \overset{\text{def}}{=} \textbf{E} \top \textbf{U} \varphi : \text{“It is possible to reach a state satisfying } \varphi \text{”}$

- $\textbf{AF} \varphi \overset{\text{def}}{=} \textbf{A} \top \textbf{U} \varphi : \text{“Along any path, there exists a state satisfying } \varphi \text{”}$

- $\textbf{EG} \varphi \overset{\text{def}}{=} \neg \textbf{AF} \neg \varphi : \text{“There is a path where } \varphi \text{ holds for any state”}$

- $\textbf{AG} \varphi \overset{\text{def}}{=} \neg \textbf{EF} \neg \varphi : \text{“} \varphi \text{ holds for any reachable state”}$

We write $S \models \varphi$ when $q_{\text{init}} \models S \varphi$. 
Abbreviations

\( \top, \bot, \varphi \lor \psi, \varphi \Rightarrow \psi \ldots \)

- \( \mathbf{E} \mathbf{F} \varphi \overset{\text{def}}{=} \mathbf{E} \top \mathbf{U} \varphi \): “It is possible to reach a state satisfying \( \varphi \)”
- \( \mathbf{A} \mathbf{F} \varphi \overset{\text{def}}{=} \mathbf{A} \top \mathbf{U} \varphi \): “Along any path, there exists a state satisfying \( \varphi \)”
- \( \mathbf{E} \mathbf{G} \varphi \overset{\text{def}}{=} \neg \mathbf{A} \mathbf{F} \neg \varphi \): “There is a path where \( \varphi \) holds for any state”
- \( \mathbf{A} \mathbf{G} \varphi \overset{\text{def}}{=} \neg \mathbf{E} \mathbf{F} \neg \varphi \): “\( \varphi \) holds for any reachable state”

We write \( S \models \varphi \) when \( q_{\text{init}} \models S \varphi \).
Abbreviations

\( \top, \bot, \varphi \lor \psi, \varphi \Rightarrow \psi \ldots \)

- \( \text{EF} \varphi \overset{\text{def}}{=} \text{E} \top \text{U} \varphi \): “It is possible to reach a state satisfying \( \varphi \)”
- \( \text{AF} \varphi \overset{\text{def}}{=} \text{A} \top \text{U} \varphi \): “Along any path, there exists a state satisfying \( \varphi \)”
- \( \text{EG} \varphi \overset{\text{def}}{=} \neg \text{AF} \neg \varphi \): “There is a path where \( \varphi \) holds for any state”
- \( \text{AG} \varphi \overset{\text{def}}{=} \neg \text{EF} \neg \varphi \): “\( \varphi \) holds for any reachable state”

We write \( S \models \varphi \) when \( q_{\text{init}} \models S \varphi \).
Examples of $CTL$ formulae

Example

- “It is possible that the door is open at the first floor while the cabin is at the second floor”

$$\textbf{E} \textbf{F} \left( open_1 \land cabin_2 \right)$$

- “Any request is eventually served”

$$\bigwedge_{i=1...n} \textbf{A} \textbf{G} \left( call_i \Rightarrow \textbf{A} \textbf{F} (cabin_i \land open_i) \right)$$

- “If a request for the $i$-th floor is done when the cabin is at the $i$-th floor, the request is satisfied immediately”

$$\bigwedge_{i=1...n} \textbf{A} \textbf{G} \left( (call_i \land cabin_i) \Rightarrow (open_i \lor \textbf{A} \textbf{X} open_i) \right)$$
Examples of $CTL$ formulae

Example

- “It is possible that the door is open at the first floor while the cabin is at the second floor”

  $$\text{E F } \left( open_1 \land cabin_2 \right)$$

- “Any request is eventually served”

  $$\bigwedge_{i=1\ldots n} \text{AG } \left( call_i \Rightarrow \text{AF } \left( cabin_i \land open_i \right) \right)$$

- “If a request for the $i$-th floor is done when the cabin is at the $i$-th floor, the request is satisfied immediately”

  $$\bigwedge_{i=1\ldots n} \text{AG } \left( \left( call_i \land cabin_i \right) \Rightarrow \left( open_i \lor \text{AX } open_i \right) \right)$$
Examples of $CTL$ formulae

**Example**

- “It is possible that the door is open at the first floor while the cabin is at the second floor”
  \[ E F \left( open_1 \land cabin_2 \right) \]

- “Any request is eventually served”
  \[ \bigwedge_{i=1...n} AG \left( call_i \Rightarrow AF \left( cabin_i \land open_i \right) \right) \]

- “If a request for the $i$-th floor is done when the cabin is at the $i$-th floor, the request is satisfied immediately”
  \[ \bigwedge_{i=1...n} AG \left( (call_i \land cabin_i) \Rightarrow (open_i \lor AX open_i) \right) \]
Outline

8. Distinguishing power

9. Comparison with $LTL$

10. $CTL^*$

11. Expressivity of the fragments of $CTL^*$
   - $UB$ and $UB^+$
   - $CTL$ vs $CTL^+$
   - $ECTL$ vs $CTL$
   - $ECTL^+$, $CTL^*$, and beyond
Outline

8 Distinguishing power

9 Comparison with $LTL$

10 $CTL^*$

11 Expressivity of the fragments of $CTL^*$
   - $UB$ and $UB^+$
   - $CTL$ vs $CTL^+$
   - $ECTL$ vs $CTL$
   - $ECTL^+$, $CTL^*$, and beyond
Distinguishing power

Two Kripke structures satisfy the same $LTL$ formulas iff they have the same set of executions (i.e. they are trace-equivalent).

\[ \forall \Phi \in LTL, \quad S \models \Phi \iff S' \models \Phi \]

But:

\[ S \models \mathbf{E} \mathbf{X} \left( \mathbf{E} \mathbf{X} P_2 \land \mathbf{E} \mathbf{X} P_3 \right) \quad \text{and} \quad S' \not\models \mathbf{E} \mathbf{X} \left( \mathbf{E} \mathbf{X} P_2 \land \mathbf{E} \mathbf{X} P_3 \right) \]

Which behavioral equivalence is characterized by $CTL$?
Distinguishing power

Two Kripke structures satisfy the same $LTL$ formulas iff they have the same set of executions (i.e. they are trace-equivalent).

\[ \forall \Phi \in LTL, \quad S \models \Phi \iff S' \models \Phi \]

But:

\[ S \models \text{EX} (\text{EX} P_2 \land \text{EX} P_3) \quad \text{and} \quad S' \not\models \text{EX} (\text{EX} P_2 \land \text{EX} P_3) \]

Which behavioral equivalence is characterized by $CTL$?
Distinguishing power

Two Kripke structures satisfy the same LTL formulas iff they have the same set of executions (i.e. they are trace-equivalent).

\[ \forall \Phi \in LTL, \; S \models \Phi \iff S' \models \Phi \]

But:

\[ S \models EX (EX P_2 \land EX P_3) \quad \text{and} \quad S' \not\models EX (EX P_2 \land EX P_3) \]

Which behavioral equivalence is characterized by CTL?
Behavioral equivalences (Van Glabbeek, 1990)

- bisimulation
  - 2-nested simulation
    - ready simulation
      - ready trace equivalence
        - failure trace equivalence
        - simulation
          - completed trace equivalence
            - trace equivalence
              - [LTL]
    - failure equivalence
      - readiness equivalence
        - possible-futures equivalence

- trace equivalence
  - [LTL]
Distinguishing power

**Definition**

Let $S_1 = \langle Q_1, q_{init}^1, R_1, l_1 \rangle$ and $S_2 = \langle Q_2, q_{init}^2, R_2, l_2 \rangle$ be two Kripke structures.

A relation $\mathcal{R} \subseteq Q_1 \times Q_2$ is a **bisimulation** iff $q_1 \mathcal{R} q_2$ implies:

- $l_1(q_1) = l_2(q_2)$,
- $\forall q_1 \rightarrow q'_1, \exists q_2 \rightarrow q'_2$ such that $q'_1 \mathcal{R} q'_2$,
- $\forall q_2 \rightarrow q'_2, \exists q_1 \rightarrow q'_1$ such that $q'_1 \mathcal{R} q'_2$.

$S_1$ and $S_2$ are bisimilar (written $S_1 \approx S_2$) iff there exists a bisimulation $\mathcal{R}$ s.t. $q_{init}^1 \mathcal{R} q_{init}^2$. 
Bisimulation vs $CTL$ (Hennessy, 1980)

$q \equiv_{CTL} r \overset{\text{def}}{=} \left( \forall \varphi \in CTL, q \models \varphi \iff r \models \varphi \right)$

**Proposition**

$q \approx r \Rightarrow q \equiv_{CTL} r$

**Proposition**

For Kripke structures with finite branching, we have:

$q \equiv_{CTL} r \Rightarrow q \approx r$

NB: the modality $E X$ is sufficient ($B(X)$).

$CTL^\infty \overset{\text{def}}{=} \text{infinitary } CTL$

**Proposition**

$q \equiv_{CTL^\infty} r \iff q \approx r$
Bisimulation vs $CTL$ (Hennessy, 1980)

$$q \equiv_{CTL} r \overset{\text{def}}{=} \left( \forall \varphi \in CTL, q \models \varphi \iff r \models \varphi \right)$$

**Proposition**

$$q \approx r \Rightarrow q \equiv_{CTL} r$$

**Proposition**

*For Kripke structures with finite branching, we have:*

$$q \equiv_{CTL} r \Rightarrow q \approx r$$

NB: the modality $E X$ is sufficient ($B(X)$).

$CTL^\infty \overset{\text{def}}{=} \text{infinitary } CTL$

**Proposition**

$$q \equiv_{CTL^\infty} r \iff q \approx r$$
Bisimulation vs $CTL$ (Hennessy, 1980)

$q \equiv_{CTL} r \overset{\text{def}}{=} \left( \forall \varphi \in CTL, q \models \varphi \iff r \models \varphi \right)$

**Proposition**

$q \approx r \implies q \equiv_{CTL} r$

**Proposition**

*For Kripke structures with finite branching, we have:*

$q \equiv_{CTL} r \implies q \approx r$

NB: the modality $\mathbf{E X}$ is sufficient ($\mathcal{B}(X)$).

$CTL^\infty \overset{\text{def}}{=} \text{infinitary } CTL$

**Proposition**

$q \equiv_{CTL^\infty} r \iff q \approx r$
Characteristic formulas for \textit{finite} KS

\textbf{Theorem (Browne, 1988)}

\textit{Given a finite KS }$S$\textit{, there is a CTL formula }$\Phi_S$\textit{ such that for any finite KS }$S'$\textit{, we have:}

\[ S' \models \Phi_S \iff S \approx S' \]

\textbullet\ Given a state }$q$\text{ of }$S$\text{ and }$n \in \mathbb{N}$\text{, we define }$\Psi^n(q)$\text{ as follows:

\begin{align*}
\Psi_0(q) & \text{ def } = \bigwedge_{P \in l(q)} P \land \bigwedge_{P \notin l(q)} \neg P \\
\Psi_{n+1}(q) & \text{ def } = \Psi_0(q) \land \bigwedge_{q \rightarrow q'} (\mathbb{E} X \Psi_n(q')) \land \mathbb{A} X (\bigvee_{q \rightarrow q'} \Psi_n(q'))
\end{align*}

\textbullet\ For any }$q$\text{ and }$q'$\text{ in }$S$, $q' \models \Psi^n(q)$ \iff the computation trees of depth }$n$\text{ rooted in }$q$\text{ and }$q'$\text{ correspond.
Characteristic formulas for finite KS

**Theorem (Browne, 1988)**

*Given a finite KS $S$, there is a CTL formula $\Phi_S$ such that for any finite KS $S'$, we have:*

$$S' \models \Phi_S \iff S \approx S'$$

- Given a state $q$ of $S$ and $n \in \mathbb{N}$, we define $\Psi^n(q)$ as follows:

**Definition**

\[
\begin{align*}
\psi^0(q) & \overset{\text{def}}{=} \bigwedge_{P \in l(q)} P \land \bigwedge_{P \not\in l(q)} \neg P \\
\psi^{n+1}(q) & \overset{\text{def}}{=} \psi^0(q) \land \bigwedge_{q \to q'} \left( \text{E X} \psi^n(q') \right) \land \text{AX} \left( \bigvee_{q \to q'} \psi^n(q') \right)
\end{align*}
\]

- For any $q$ and $q'$ in $S$, $q' \models \psi^n(q)$ iff the computation trees of depth $n$ rooted in $q$ and $q'$ correspond.
Characteristic formulas for finite KS

Theorem (Browne, 1988)

Given a finite KS $S$, there is a CTL formula $\Phi_S$ such that for any finite KS $S'$, we have:

$$S' \models \Phi_S \iff S \simeq S'$$

- Given a state $q$ of $S$ and $n \in \mathbb{N}$, we define $\Psi^n(q)$ as follows:

**Definition**

$$\Psi^0(q) \overset{\text{def}}{=} \bigwedge_{P \in I(q)} P \land \bigwedge_{P \notin I(q)} \neg P$$

$$\Psi^{n+1}(q) \overset{\text{def}}{=} \Psi^0(q) \land \bigwedge_{q \rightarrow q'} \left( \text{E} X \Psi^n(q') \right) \land \text{A} X \left( \bigvee_{q \rightarrow q'} \Psi^n(q') \right)$$

- For any $q$ and $q'$ in $S$, $q' \models \Psi^n(q)$ iff the computation trees of depth $n$ rooted in $q$ and $q'$ correspond.
Characteristic formulas for finite KS

Theorem (Browne, 1988)

Given a finite KS $S$, there is a CTL formula $\Phi_S$ such that for any finite KS $S'$, we have:

$$S' \models \Phi_S \iff S \approx S'$$

- Given a state $q$ of $S$ and $n \in \mathbb{N}$, we define $\Psi^n(q)$ as follows:

**Definition**

$$\Psi^0(q) \overset{\text{def}}{=} \bigwedge_{P \in \ell(q)} P \land \bigwedge_{P \notin \ell(q)} \neg P$$

$$\Psi^{n+1}(q) \overset{\text{def}}{=} \Psi^0(q) \land \bigwedge_{q \rightarrow q'} \left( \mathbf{E} X \Psi^n(q') \right) \land \mathbf{A} X \left( \bigvee_{q \rightarrow q'} \Psi^n(q') \right)$$

- For any $q$ and $q'$ in $S$, $q' \models \Psi^n(q)$ iff the computation trees of depth $n$ rooted in $q$ and $q'$ correspond.
Characteristic formulas for finite KS

- There exists a number $c$ for $S$ s.t. for any state $q$ and $q'$ in $S$, we have:

$$ q' \models \Psi^c(q) \iff q \approx q' $$

- We define $\Phi_S$ as follows:

Definition

$$ \Phi_S \overset{\text{def}}{=} \psi^c(q_{\text{init}}) \land \bigwedge_{q \in Q} \mathbf{AG} (\psi^c(q) \Rightarrow \bigwedge_{q \rightarrow q'} \mathbf{EX} \psi^c(q') \land \mathbf{AX} \bigvee_{q \rightarrow q'} \psi^c(q') ) $$

Clearly $S' \models \Phi_S \iff S \approx S'$

(the same approach can be used to characterize equivalence w.r.t. stuttering with $CTL \setminus \text{X}$)
Characteristic formulas for finite KS

- There exists a number $c$ for $S$ s.t. for any state $q$ and $q'$ in $S$, we have:

  $q' \models \psi^c(q) \iff q \approx q'$

- We define $\Phi_S$ as follows:

  \[
  \Phi_S \overset{\text{def}}{=} \psi^c(q_{\text{init}}) \land \bigwedge_{q \in Q} AG \left( \psi^c(q) \Rightarrow \bigwedge_{q \rightarrow q'} EX \psi^c(q') \land AX \bigvee_{q \rightarrow q'} \psi^c(q') \right)
  \]

  Clearly $S' \models \Phi_S \iff S \approx S'$

  (the same approach can be used to characterize equivalence w.r.t. stuttering with $CTL \setminus X$)
Characteristic formulas for *finite* KS

- There exists a number $c$ for $S$ s.t. for any state $q$ and $q'$ in $S$, we have:
  
  \[
  q' \models \Psi^c(q) \iff q \approx q'
  \]

- We define $\Phi_S$ as follows:

  \[
  \Phi_S \overset{\text{def}}{=} \Psi^c(q_{\text{init}}) \land \bigwedge_{q \in Q} \mathbf{A} \mathbf{G} \left( \Psi^c(q) \Rightarrow \bigwedge_{q \rightarrow q'} \mathbf{E} \mathbf{X} \Psi^c(q') \land \mathbf{A} \mathbf{X} \bigvee_{q \rightarrow q'} \Psi^c(q') \right)
  \]

  Clearly $S' \models \Phi_S \iff S \approx S'$

  (the same approach can be used to characterize equivalence w.r.t. stuttering with $\mathit{CTL} \setminus \mathbf{X}$)
Characteristic formulas for finite KS

- There exists a number $c$ for $S$ s.t. for any state $q$ and $q'$ in $S$, we have:

$$q' \models \Psi_c(q) \iff q \approx q'$$

- We define $\Phi_S$ as follows:

\[
\Phi_S \overset{\text{def}}{=} \Psi_c(q_{\text{init}}) \land \bigwedge_{q \in Q} \mathbf{A} \mathbf{G} \left( \Psi_c(q) \Rightarrow \bigwedge_{q \rightarrow q'} \mathbf{E} \mathbf{X} \Psi_c(q') \lor \mathbf{A} \mathbf{X} \bigvee_{q \rightarrow q'} \Psi_c(q') \right)
\]

Clearly $S' \models \Phi_S \iff S \approx S'$

(the same approach can be used to characterize equivalence w.r.t. stuttering with $\mathbf{CTL} \setminus \mathbf{X}$)
Characteristic formulas for finite KS

- There exists a number $c$ for $S$ s.t. for any state $q$ and $q'$ in $S$, we have:
  \[ q' \models \psi^c(q) \iff q \approx q' \]

- We define $\Phi_S$ as follows:

\[
\Phi_S \overset{\text{def}}{=} \psi^c(q_{\text{init}}) \land \bigwedge_{q \in Q} A\ G\left(\psi^c(q) \Rightarrow \bigwedge_{q \rightarrow q'} E\ X\ \psi^c(q') \land A\ X\ \bigvee_{q \rightarrow q'} \psi^c(q')\right)
\]

Clearly $S' \models \Phi_S \iff S \approx S'$

(the same approach can be used to characterize equivalence w.r.t. stuttering with $\text{CTL} \setminus X$)
Characteristic formulas for finite KS

- There exists a number $c$ for $S$ s.t. for any state $q$ and $q'$ in $S$, we have:

$$q' \models \Psi^c(q) \iff q \approx q'$$

- We define $\Phi_S$ as follows:

\[
\Phi_S \overset{\text{def}}{=} \Psi^c(q_{\text{init}}) \land \bigwedge_{q \in Q} \mathbf{A G} \left( \Psi^c(q) \Rightarrow \bigwedge_{q \rightarrow q'} \mathbf{E X} \Psi^c(q') \land \mathbf{A X} \bigvee_{q \rightarrow q'} \Psi^c(q') \right)
\]

Clearly $S' \models \Phi_S \iff S \approx S'$

(the same approach can be used to characterize equivalence w.r.t. stuttering with $\text{CTL} \setminus \text{X}$)
Outline

8 Distinguishing power

9 Comparison with \textit{LTL}

10 \textit{CTL}\textsuperscript{*}

11 Expressivity of the fragments of \textit{CTL}\textsuperscript{*}
   - UB and $UB^+$
   - $CTL$ vs $CTL^+$
   - ECTL vs CTL
   - $ECTL^+$, $CTL^*$, and beyond
Comparison with *LTL*

For a $\Phi \in LTL$, we write $S \models \Phi$ iff $\rho \models \Phi \ \forall \rho \in \text{Exec}(S)$ (i.e. $S \models A \Phi$).

**Proposition**

*LTL is NOT as expressive as CTL.*

The previous *CTL*-formulas can distinguish KS that verify the same *LTL* formulae. Another ex.: $A \ G \ E \ F \ P$

**Proposition**

*There is no CTL formula equivalent to $A \ F \ G \ P$.*

(proof in next section)

NB: $A \ F \ A \ G \ P \Rightarrow A \ F \ G \ P$ but $A \ F \ G \ P \not\Rightarrow A \ F \ A \ G \ P$.

**Proposition**

*LTL and CTL are uncomparable.*
Comparison with *LTL*

For a $\Phi \in LTL$, we write $S \models \Phi$ iff $\rho \models \Phi$ $\forall \rho \in$ Exec($S$) (i.e. $S \models A\Phi$).

**Proposition**

*LTL* is *NOT* as expressive as *CTL*.

The previous *CTL*-formulas can distinguish KS that verify the same *LTL* formulae.

Another ex.: $A\ G\ E\ F\ P$

**Proposition**

*There is no* *CTL* formula equivalent to $A\ F\ G\ P$.

(proof in next section)

NB: $A\ F\ A\ G\ P \Rightarrow A\ F\ G\ P$ but $A\ F\ G\ P \not\Rightarrow A\ F\ A\ G\ P$.

**Proposition**

*LTL* and *CTL* are uncomparable.
Comparison with \textit{LTL}

For a $\Phi \in \textit{LTL}$, we write $S \models \Phi$ iff $\rho \models \Phi \ \forall \rho \in \text{Exec}(S)$ (\textit{i.e.} $S \models \text{A}\Phi$).

\textbf{Proposition}

\textit{LTL is NOT as expressive as CTL.}

The previous \textit{CTL}-formulas can distinguish KS that verify the same \textit{LTL} formulae. Another ex.: \textbf{A G E F P}

\textbf{Proposition}

\textit{There is no CTL formula equivalent to A F G P.}

(proof in next section)

NB: \textbf{A F A G P} $\Rightarrow$ \textbf{A F G P} but \textbf{A F G P} $\not\Rightarrow$ \textbf{A F A G P}.

\textbf{Proposition}

\textit{LTL and CTL are uncomparable.}
Outline

8 Distinguishing power

9 Comparison with $LTL$

10 $CTL^*$

11 Expressivity of the fragments of $CTL^*$
   - $UB$ and $UB^+$
   - $CTL$ vs $CTL^+$
   - $ECTL$ vs $CTL$
   - $ECTL^+$, $CTL^*$, and beyond
Definition of $CTL^*$ (Emerson & Halpern, 1986)

Idea: merging linear-time TL and branching-time TL...

Definition

$CTL^* \ni \varphi, \psi \ ::= \ P_1 \mid P_2 \mid \ldots \mid \neg \varphi \mid \varphi \land \psi \mid E\varphi_p \mid A\varphi_p$

$CTL^*_p \ni \varphi_p, \psi_p \ ::= \ \varphi \mid \neg \varphi_p \mid \varphi_p \land \psi_p \mid X\varphi_p \mid \varphi_p U \psi_p$

with $P \in AP$

State formulae ($CTL^*$) are interpreted over states of a KS.
Path formulae ($CTL^*_p$) are interpreted over executions in a KS.
Semantics of $CTL^*$

**Definition**

\[
\begin{align*}
q \models_s P & \quad \text{iff} \quad P \in l(q) \\
q \models_s \varphi \land \psi & \quad \text{iff} \quad q \models_s \varphi \text{ and } q \models_s \psi \\
q \models_s \neg \varphi & \quad \text{iff} \quad q \not\models_s \varphi \\
q \models_s \exists \varphi_p & \quad \text{iff} \quad \exists \rho \in \text{Exec}(q) \text{ s.t. } \rho \models_s \varphi_p \\
q \models_s \forall \varphi_p & \quad \text{iff} \quad \forall \rho \in \text{Exec}(q) \text{ we have } \rho \models_s \varphi_p \\
\rho \models_s P & \quad \text{iff} \quad P \in l(\rho(0)) \\
\rho \models_s \varphi_p \land \psi_p & \quad \text{iff} \quad \rho \models_s \varphi \text{ and } \rho \models_s \psi \\
\rho \models_s \neg \varphi_p & \quad \text{iff} \quad \rho \not\models_s \varphi_p \\
\rho \models_s \varphi_p \lor \psi_p & \quad \text{iff} \quad \exists i \geq 0, \rho^i \models_s \psi_p \text{ and } \\
& \quad \forall 0 \leq j < i, \text{ we have } \rho^j \models_s \varphi_p \\
\rho \models_s \forall \varphi_p & \quad \text{iff} \quad \rho^1 \models_s \varphi_p \\
\end{align*}
\]
Fragments of $CTL^*$ – 1

Definition (Ben-Ari, Manna & Pnueli, 1983)

$$UB \ni \varphi, \psi ::= P_1 \mid P_2 \mid \ldots \mid \neg \varphi \mid \varphi \land \psi \mid E \varphi_p \mid A \varphi_p$$

$$UB_p \ni \varphi_p ::= X \varphi \mid F \varphi$$

Definition

$$UB^+ \ni \varphi, \psi ::= P_1 \mid P_2 \mid \ldots \mid \neg \varphi \mid \varphi \land \psi \mid E \varphi_p \mid A \varphi_p$$

$$UB_p^+ \ni \varphi_p, \psi_p ::= \neg \varphi_p \mid \varphi_p \land \psi_p \mid X \varphi \mid F \varphi$$

$A(F P_1 \Rightarrow F P_2)$ is in $UB^+$ but not in $UB$. 
Fragments of $CTL^* - 2$

Definition (Clarke & Emerson, 1981)

$$CTL \ni \varphi, \psi ::= P_1 \mid P_2 \mid \ldots \mid \neg \varphi \mid \varphi \land \psi \mid E\varphi_p \mid A\varphi_p$$

$$CTL_p \ni \varphi_p ::= X\varphi \mid \varphi U \psi$$

Definition (Emerson & Halpern, 1985)

$$CTL^+ \ni \varphi, \psi ::= P_1 \mid P_2 \mid \ldots \mid \neg \varphi \mid \varphi \land \psi \mid E\varphi_p \mid A\varphi_p$$

$$CTL^+_p \ni \varphi_p, \psi_p ::= \neg \varphi_p \mid \varphi_p \land \psi_p \mid X\varphi \mid \varphi U \psi$$

$A(P_1 U P_2 \Rightarrow P_3 U P_4)$ is in $CTL^+$ but not in $CTL$. 
Fragments of $CTL^*$ – 3

“infinitely often” : $\mathsf{F} \overset{\text{def}}{=} \mathsf{G} \mathsf{F}$

**Definition (Emerson & Emerson, 1986)**

$ECTL \ni \varphi, \psi ::= P_1 | P_2 | \ldots | \neg \varphi | \varphi \land \psi | E\varphi_p | A\varphi_p$

$ECTL_p \ni \varphi_p ::= X \varphi | \varphi U \psi | \mathsf{F} \varphi$

$ECTL^+ \ni \varphi, \psi ::= P_1 | \ldots | \neg \varphi | \varphi \land \psi | E\varphi_p | A\varphi_p$

$ECTL^+_p \ni \varphi_p, \psi_p ::= \neg \varphi_p | \varphi_p \land \psi_p | X \varphi | \varphi U \psi | \mathsf{F} \varphi$

$E P_1 U (P_2 \land E \mathsf{F} P_3)$ is in $ECTL$.

$E(P_1 U P_2 \land \mathsf{F} P_3 \land \mathsf{F} P_4)$ is in $ECTL^+$ but not in $ECTL$. 
Outline

8 Distinguishing power

9 Comparison with LTL

10 CTL*

11 Expressivity of the fragments of CTL*
   - $UB$ and $UB^+$
   - $CTL$ vs $CTL^+$
   - $ECTL$ vs $CTL$
   - $ECTL^+$, $CTL^*$, and beyond
Syntactic inclusions

$L \rightarrow L'$: “$L$ is included in $L'$”

$UB \rightarrow CTL \rightarrow ECTL \rightarrow CTL^*$

$UB^+ \rightarrow CTL^+ \rightarrow ECTL^+$

And all these logics have the same distinguishing power for structures with finite branching.
Outline

8 Distinguishing power

9 Comparison with $LTL$

10 $CTL^*$

11 Expressivity of the fragments of $CTL^*$
  - $UB$ and $UB^+$
  - $CTL$ vs $CTL^+$
  - $ECTL$ vs $CTL$
  - $ECTL^+$, $CTL^*$, and beyond
Expressivity of $UB$

$UB$ formulas are built from atomic propositions, boolean operators and the following modalities: $E X$, $A X$, $E F$, $A F$, $E G$ and $A G$.

As we have:

\[
\begin{align*}
A X \varphi & \equiv \neg E X \neg \varphi \\
E G \varphi & \equiv \neg A F \neg \varphi \\
A G \varphi & \equiv \neg E F \neg \varphi
\end{align*}
\]

we only consider modalities $E X$, $E F$ and $A F$.

But $UB^+$ does not contain a finite number of modalities.
For ex.: $E (F P_1 \land F P_2 \land G \neg P_3) \in UB^+$

$UB^+$ and $UB$ have the same distinguishing power...
But $UB^+$ is more expressive than $UB$...
Expressivity of $UB$

$UB$ formulas are built from atomic propositions, boolean operators and the following modalities: $E X$, $A X$, $E F$, $A F$, $E G$ and $A G$.

As we have:

$$\begin{align*}
A X \varphi & \equiv \neg E X \neg \varphi \\
E G \varphi & \equiv \neg A F \neg \varphi \\
A G \varphi & \equiv \neg E F \neg \varphi
\end{align*}$$

we only consider modalities $E X$, $E F$ and $A F$.

But $UB^+$ does not contain a finite number of modalities. For ex.: $E(F P_1 \land F P_2 \land G \neg P_3) \in UB^+$

$UB^+$ and $UB$ have the same distinguishing power...
But $UB^+$ is more expressive than $UB$...
Expressivity of \( UB \)

\( UB \) formulas are built from atomic propositions, boolean operators and the following modalities: \( E X \), \( A X \), \( E F \), \( A F \), \( E G \) and \( A G \).

As we have:

\[
\begin{align*}
A X \varphi & \equiv \neg E X \neg \varphi \\
E G \varphi & \equiv \neg A F \neg \varphi \\
A G \varphi & \equiv \neg E F \neg \varphi
\end{align*}
\]

we only consider modalities \( E X \), \( E F \) and \( A F \).

But \( UB^+ \) does not contain a finite number of modalities.

For ex.: \( E( F P_1 \land F P_2 \land G \neg P_3 ) \in UB^+ \)

\( UB^+ \) and \( UB \) have the same distinguishing power...

But \( UB^+ \) is more expressive than \( UB \). . .
UB less expr. than $UB^+$ (Emerson & Halpern, 1985)

\[ \models P_1 \land \neg P_2 \quad \models \neg P_1 \land \neg P_2 \quad \models P_1 \land P_2 \]

\[ \alpha_i \models \mathcal{E}(G \rho_1 \land \mathcal{F} \rho_2) \quad \text{but} \quad \alpha'_i \not\models \mathcal{E}(G \rho_1 \land \mathcal{F} \rho_2) \]

**Lemma**

\[ \forall \varphi \in UB, \ |\varphi| \leq i \Rightarrow \left( \alpha_i \models \varphi \iff \alpha'_i \models \varphi \right) \]
UB less expr. than $UB^+$ (Emerson & Halpern, 1985)

\[ |\alpha_i| = P_1 \land \neg P_2 \quad |\alpha_1'| = \neg P_1 \land \neg P_2 \quad |\alpha_0| = P_1 \land P_2 \]

\[ \alpha_i \models E\left( G\ P_1 \land F\ P_2 \right) \quad \text{but} \quad \alpha_1' \not\models E\left( G\ P_1 \land F\ P_2 \right) \]

Lemma

\[ \forall \varphi \in UB, \ |\varphi| \leq i \implies (\alpha_i \models \varphi \iff \alpha_i' \models \varphi) \]
**UB** less expr. than **UB**$^+$ (Emerson & Halpern, 1985)

\[
\begin{aligned}
\alpha_i &\models P_1 \land \neg P_2 & \quad \beta_i &\models \neg P_1 \land \neg P_2 & \quad \gamma_0 &\models P_1 \land P_2 \\
\end{aligned}
\]

\[
\alpha_i \models E\left( G P_1 \land F P_2 \right) \text{ but } \alpha'_i \nmodels E\left( G P_1 \land F P_2 \right)
\]

**Lemma**

\[
\forall \varphi \in UB, |\varphi| \leq i \Rightarrow \left( \alpha_i \models \varphi \iff \alpha'_i \models \varphi \right)
\]
\textit{UB} less expr. than \textit{UB}^+ (Emerson & Halpern, 1985)

\[
\begin{align*}
\alpha_i & \models P_1 \land \lnot P_2 & \beta_i & \models \lnot P_1 \land \lnot P_2 & \gamma_0 & \models P_1 \land P_2 \\
\alpha_i & \models E\left( G \ P_1 \land F \ P_2 \right) & \beta_i & \models E\left( G \ P_1 \land F \ P_2 \right) & \gamma_0 & \models E\left( G \ P_1 \land F \ P_2 \right)
\end{align*}
\]

Lemma

\[
\forall \varphi \in UB, \ |\varphi| \leq i \Rightarrow \left( \alpha_i \models \varphi \iff \alpha_i' \models \varphi \right)
\]

\textit{Proof}: by induction on \(|\varphi|\).
Direct for atomic propositions and boolean operators.
UB less expr. than $UB^+$ (Emerson & Halpern, 1985)

\[ \alpha_i \models P_1 \land \neg P_2 \quad \models \neg P_1 \land \neg P_2 \quad \models P_1 \land P_2 \]

\[ \alpha_i \models \mathbf{E}\left( G \, P_1 \land F \, P_2 \right) \] but \[ \alpha'_i \not\models \mathbf{E}\left( G \, P_1 \land F \, P_2 \right) \]

**Lemma**

\[ \forall \phi \in UB, \ |\phi| \leq i \Rightarrow (\alpha_i \models \phi \iff \alpha'_i \models \phi) \]

**Proof:** by induction on $|\phi|$.

\( \phi = \mathbf{E} \mathbf{X} \psi \).

If $\alpha_i \models \mathbf{E} \mathbf{X} \psi$ then either $\beta_i \models \psi$, $\alpha_i \models \psi$ or $\alpha_{i-1} \models \psi$.

Induction hypothesis allows to deduce $\alpha'_i \models \mathbf{E} \mathbf{X} \psi$. 

UB less expr. than UB⁺ (Emerson & Halpern, 1985)

\[ \alpha_i \models E\left( G P_1 \land F P_2 \right) \quad \text{but} \quad \alpha'_i \not\models E\left( G P_1 \land F P_2 \right) \]

Lemma

\[ \forall \varphi \in UB, |\varphi| \leq i \Rightarrow \left( \alpha_i \models \varphi \iff \alpha'_i \models \varphi \right) \]

Proof: by induction on \( |\varphi| \).

\( \varphi = E F \psi. \)

If \( \alpha_i \models E F \psi \), then either \( \alpha_i \models \psi \) (and the i.h. can be applied) or another state (also reachable from \( \alpha'_i \)) satisfies \( \psi \).
**UB less expr. than UB^+ (Emerson & Halpern, 1985)**

\[
\alpha_i \models P_1 \land \neg P_2 \quad \alpha'_i \models \neg P_1 \land \neg P_2 \quad \alpha \models P_1 \land P_2
\]

\[
\alpha_i \models E\left( G\ P_1 \land F\ P_2 \right) \quad \text{but} \quad \alpha'_i \not\models E\left( G\ P_1 \land F\ P_2 \right)
\]

**Lemma**

\[
\forall \varphi \in UB, |\varphi| \leq i \Rightarrow \left( \alpha_i \models \varphi \iff \alpha'_i \models \varphi \right)
\]

**Proof:** by induction on $|\varphi|$.

$\varphi = \textbf{A} F \psi$.

If $\alpha_i \models \textbf{A} F \psi$, then $\alpha_i \models \psi$ and from the i.h. we have $\alpha'_i \models \psi$ and then $\alpha'_i \models \textbf{A} F \psi$. 
**UB** less expr. than **UB**\(^+\) (Emerson & Halpern, 1985)

\[
\text{for } \alpha_i \models \mathbf{E} \left( \mathbf{G} P_1 \land \mathbf{F} P_2 \right) \text{ but } \alpha'_i \not\models \mathbf{E} \left( \mathbf{G} P_1 \land \mathbf{F} P_2 \right)
\]

**Lemma**

\[
\forall \varphi \in \text{UB}, \ |\varphi| \leq i \Rightarrow (\alpha_i \models \varphi \iff \alpha'_i \models \varphi)
\]

**Conclusion:** there is no **UB** formula equivalent to \(\mathbf{E}(\mathbf{G} P_1 \land \mathbf{F} P_2)\).
**UB**$^+$ vs **CTL** – (Emerson & Halpern, 1985)

Theorem

**CTL is strictly more expressive than UB**$^+$!

- Let $\Phi \overset{\text{def}}{=} \exists P_1 \cup P_2$.
- Assume there exists an **UB**$^+$ formula $\Psi$ equivalent to $\Phi$.
- Let $\Psi'$ be the formula $\Psi$ where any path quantifiers $\exists$ and $\forall$ have been removed: $\Psi' \in \mathcal{L}(F, G, X)$.
- For any path $\rho$, we clearly have $\rho(1) \models \Psi$ iff $\rho \models \Psi'$.
- Then $\Psi'$ is equivalent to $P_1 \cup P_2$.
- But $\mathcal{L}(F, G, X) < \mathcal{L}(U, X)$!!

There is no **UB**$^+$ formula equivalent to $\exists P_1 \cup P_2$.

From the expressivity point of view, we have:

$UB < UB^+ < CTL$
Theorem

*CTL is strictly more expressive than UB*^+!*!

- Let \( \Phi \overset{\text{def}}{=} E P_1 U P_2 \).
- Assume there exists an UB^+ formula \( \Psi \) equivalent to \( \Phi \).
- Let \( \Psi' \) be the formula \( \Psi \) where any path quantifiers \( E \) and \( A \) have been removed: \( \Psi' \in \mathcal{L}(F, G, X) \).
- For any path \( \rho \), we clearly have \( \rho(1) \models \Psi \) iff \( \rho \models \Psi' \).
- Then \( \Psi' \) is equivalent to \( P_1 U P_2 \).
- But \( \mathcal{L}(F, G, X) \prec \mathcal{L}(U, X) \).

There is no UB^+ formula equivalent to \( E P_1 U P_2 \).

From the expressivity point of view, we have:

\[
UB < UB^+ < CTL
\]
\( UB^+ \) vs \( CTL \) – (Emerson & Halpern, 1985)

**Theorem**

\( CTL \) is strictly more expressive than \( UB^+ \)!

- Let \( \Phi \overset{\text{def}}{=} E \, P_1 \cup P_2 \).
- Assume there exists an \( UB^+ \) formula \( \Psi \) equivalent to \( \Phi \).
- Let \( \Psi' \) be the formula \( \Psi \) where any path quantifiers \( E \) and \( A \) have been removed : \( \Psi' \in \mathcal{L}(F, G, X) \).
- For any path \( \rho \), we clearly have \( \rho(1) \models \Psi \) iff \( \rho \models \Psi' \).
- Then \( \Psi' \) is equivalent to \( P_1 \cup P_2 \).
- But \( \mathcal{L}(F, G, X) < \mathcal{L}(U, X) \).

There is no \( UB^+ \) formula equivalent to \( E \, P_1 \cup P_2 \).

From the expressivity point of view, we have:

\[ UB < UB^+ < CTL \]
**UB\(^+\) vs CTL** – (Emerson & Halpern, 1985)

**Theorem**

*CTL is strictly more expressive than UB\(^+\)!*

- Let \(\Phi \overset{\text{def}}{=} E \ P_1 \ U \ P_2\).
- Assume there exists an UB\(^+\) formula \(\Psi\) equivalent to \(\Phi\).
- Let \(\Psi'\) be the formula \(\Psi\) where any path quantifiers \(E\) and \(A\) have been removed: \(\Psi' \in \mathcal{L}(F, G, X)\).
- For any path \(\rho\), we clearly have \(\rho(1) \models \Psi\) iff \(\rho \models \Psi'\).
- Then \(\Psi'\) is equivalent to \(P_1 \ U \ P_2\).
- But \(\mathcal{L}(F, G, X) < \mathcal{L}(U, X)\)!!

There is no UB\(^+\) formula equivalent to \(E \ P_1 \ U \ P_2\).

From the expressivity point of view, we have:

\[ UB < UB^+ < CTL \]
$UB^+$ vs $CTL$ – (Emerson & Halpern, 1985)

**Theorem**

*CTL is strictly more expressive than $UB^+$!*

- Let $\Phi \overset{\text{def}}{=} E P_1 U P_2$.
- Assume there exists an $UB^+$ formula $\Psi$ equivalent to $\Phi$.
- Let $\Psi'$ be the formula $\Psi$ where any path quantifiers $E$ and $A$ have been removed: $\Psi' \in L(F, G, X)$.
- For any path $\rho$, we clearly have $\rho(1) \models \Psi$ iff $\rho \models \Psi'$.
- Then $\Psi'$ is equivalent to $P_1 U P_2$.
- But $L(F, G, X) < L(U, X)$ !!

There is no $UB^+$ formula equivalent to $E P_1 U P_2$.

From the expressivity point of view, we have:

$UB < UB^+ < CTL$
Theorem

**CTL is strictly more expressive than UB**$^+$!

- Let $\Phi \overset{\text{def}}{=} \mathbf{E} P_1 \mathbf{U} P_2$.
- Assume there exists an $UB^+$ formula $\Psi$ equivalent to $\Phi$.
- Let $\Psi'$ be the formula $\Psi$ where any path quantifiers $\mathbf{E}$ and $\mathbf{A}$ have been removed: $\Psi' \in \mathcal{L}(\mathbf{F}, \mathbf{G}, \mathbf{X})$.
- For any path $\rho$, we clearly have $\rho(1) \models \Psi$ iff $\rho \models \Psi'$.
- Then $\Psi'$ is equivalent to $P_1 \mathbf{U} P_2$.
- But $\mathcal{L}(\mathbf{F}, \mathbf{G}, \mathbf{X}) < \mathcal{L}(\mathbf{U}, \mathbf{X})$!!

There is no $UB^+$ formula equivalent to $\mathbf{E} P_1 \mathbf{U} P_2$.

From the expressivity point of view, we have:

$$UB < UB^+ < CTL$$
\( UB^+ \) vs \( CTL \) – (Emerson & Halpern, 1985)

**Theorem**

*CTL is strictly more expressive than \( UB^+ \)!*

- Let \( \Phi \overset{\text{def}}{=} E P_1 U P_2 \).
- Assume there exists an \( UB^+ \) formula \( \Psi \) equivalent to \( \Phi \).
- Let \( \Psi' \) be the formula \( \Psi \) where any path quantifiers \( E \) and \( A \) have been removed: \( \Psi' \in \mathcal{L}(F, G, X) \).
- For any path \( \rho \), we clearly have \( \rho(1) \models \Psi \iff \rho \models \Psi' \).
- Then \( \Psi' \) is equivalent to \( P_1 U P_2 \).
- But \( \mathcal{L}(F, G, X) < \mathcal{L}(U, X) \) !

There is no \( UB^+ \) formula equivalent to \( E P_1 U P_2 \).

From the expressivity point of view, we have:

\[ UB < UB^+ < CTL \]
UB$^+$ vs CTL – (Emerson & Halpern, 1985)

**Theorem**

*CTL is strictly more expressive than UB$^+$.***

- Let $\Phi \overset{\text{def}}{=} E \ P_1 \ U \ P_2$.
- Assume there exists an $UB^+$ formula $\Psi$ equivalent to $\Phi$.
- Let $\Psi'$ be the formula $\Psi$ where any path quantifiers $E$ and $A$ have been removed: $\Psi' \in \mathcal{L}(F, G, X)$.
- For any path $\rho$, we clearly have $\rho(1) \models \Psi$ iff $\rho \models \Psi'$.
- Then $\Psi'$ is equivalent to $P_1 \ U \ P_2$.
- But $\mathcal{L}(F, G, X) < \mathcal{L}(U, X)$ !!

There is no $UB^+$ formula equivalent to $E \ P_1 \ U \ P_2$.

From the expressivity point of view, we have:

$$UB < UB^+ < CTL$$
**UB⁺ vs CTL – (Emerson & Halpern, 1985)**

**Theorem**

*CTL is strictly more expressive than UB⁺!* 

- Let $\Phi \overset{\text{def}}{=} E P_1 U P_2$.
- Assume there exists an $UB⁺$ formula $\Psi$ equivalent to $\Phi$.
- Let $\Psi'$ be the formula $\Psi$ where any path quantifiers $E$ and $A$ have been removed: $\Psi' \in \mathcal{L}(F, G, X)$.
- For any path $\rho$, we clearly have $\rho(1) \models \Psi$ iff $\rho \models \Psi'$.
- Then $\Psi'$ is equivalent to $P_1 U P_2$.
- But $\mathcal{L}(F, G, X) < \mathcal{L}(U, X)$ !!

There is no $UB⁺$ formula equivalent to $E P_1 U P_2$.

From the expressivity point of view, we have:

$UB < UB⁺ < CTL$
**Theorem**

**CTL is strictly more expressive than UB⁺!**

- Let $\Phi \stackrel{\text{def}}{=} E\ P_1 \ U\ P_2$.
- Assume there exists an $UB⁺$ formula $\Psi$ equivalent to $\Phi$.
- Let $\Psi'$ be the formula $\Psi$ where any path quantifiers $E$ and $A$ have been removed: $\Psi' \in \mathcal{L}(F, G, X)$.
- For any path $\rho$, we clearly have $\rho(1) \models \Psi$ iff $\rho \models \Psi'$.
- Then $\Psi'$ is equivalent to $P_1 \ U \ P_2$.
- But $\mathcal{L}(F, G, X) < \mathcal{L}(U, X)$ !!

There is no $UB⁺$ formula equivalent to $E\ P_1 \ U\ P_2$.

From the expressivity point of view, we have:

$$UB < UB⁺ < CTL$$
Outline

8  Distinguishing power

9  Comparison with \( LTL \)

10  \( CTL^* \)

11  Expressivity of the fragments of \( CTL^* \)
   - \( UB \) and \( UB^+ \)
   - \( CTL \) vs \( CTL^+ \)
   - \( ECTL \) vs \( CTL \)
   - \( ECTL^+ \), \( CTL^* \), and beyond
Expressivity of $CTL$

$CTL = \mathcal{B}(X, U)$.

$CTL$ formulae are built from atomic propositions, boolean operators and the following modalities: $EX$, $EU$ and $AU$.

**Theorem**

$A \varphi U \psi \equiv AF \psi \land \neg EU \neg \psi \land (\neg \psi \land \neg \varphi)$

$\Rightarrow$ The $TL$ based on the modalities $EX$, $EU$ and $AF$ is as expressive as $CTL$.

But:

**Theorem (Laroussinie, 1995)**

It is not possible to express $EP_1 U P_2$ with $EX$, $AU$ and $EF$.

$\Rightarrow$ $CTL$ is strictly more expressive than the $TL$ based on the modalities $EX$, $AU$ and $EF$.

Is it possible to translate $CTL^+$ formulae in $CTL$?
Expressivity of \( CTL \)

\[ CTL = \mathcal{B}(X, U). \]

\( CTL \) formulae are built from atomic propositions, boolean operators and the following modalities: \( EX, EU \) and \( AU \).

**Theorem**

\[ A\phi U \psi \equiv AF\psi \land \neg E\neg\psi U (\neg\psi \land \neg\varphi) \]

\( \Rightarrow \) The TL based on the modalities \( EX, EU \) and \( AF \) is as expressive as \( CTL \).

But:

**Theorem (Laroussinie, 1995)**

*It is not possible to express \( EP_1 U P_2 \) with \( EX, AU \) and \( EF \).*

\( \Rightarrow \) \( CTL \) is strictly more expressive than the TL based on the modalities \( EX, AU \) and \( EF \).

Is it possible to translate \( CTL^+ \) formulae in \( CTL \)?
**Expressivity of CTL**

\[ CTL = B(X, U) \].

\( CTL \) formulae are built from atomic propositions, boolean operators and the following modalities: \( \texttt{E X} \), \( \texttt{E U} \) and \( \texttt{A U} \).

**Theorem**

\[
\texttt{A} \varphi \texttt{ U } \psi \equiv \texttt{A F } \psi \land \neg \texttt{E } \neg \psi \texttt{ U } (\neg \psi \land \neg \varphi)
\]

\[ \Rightarrow \text{ The TL based on the modalities } \texttt{E X}, \texttt{E U} \text{ and } \texttt{A F} \text{ is as expressive as CTL.} \]

**But:**

**Theorem (Laroussinie, 1995)**

*It is not possible to express \( \texttt{E P}_1 \texttt{ U } P_2 \) with \( \texttt{E X}, \texttt{A U} \text{ and } \texttt{E F} \).*

\[ \Rightarrow \text{ CTL is strictly more expressive than the TL based on the modalities } \texttt{E X}, \texttt{A U} \text{ and } \texttt{E F}. \]

Is it possible to translate \( CTL^+ \) formulae in \( CTL \)?
**CTL vs CTL**

\[
E\left( P_1 \textbf{ U } P_2 \land P_3 \textbf{ U } P_4 \land G P_5 \right) \equiv \\
E(P_1 \land P_3 \land P_5) \textbf{ U } \left( P_2 \land E(P_3 \land P_5) \textbf{ U } (P_4 \land EG P_5) \right) \lor \\
E(P_1 \land P_3 \land P_5) \textbf{ U } \left( P_4 \land E(P_1 \land P_5) \textbf{ U } (P_2 \land EG P_5) \right)
\]

Theorem (Emerson & Halpern, 1985)

- Any CTL\(^+\) formula can be translated in an equivalent CTL formula.
- CTL is as expressive as CTL\(^+\).
**CTL vs CTL⁺**

\[
E\left( P_1 \text{ U } P_2 \land P_3 \text{ U } P_4 \land G P_5 \right) \equiv \\
E(P_1 \land P_3 \land P_5) \text{ U } \left( P_2 \land E(P_3 \land P_5) \text{ U } (P_4 \land EG P_5) \right) \lor \\
E(P_1 \land P_3 \land P_5) \text{ U } \left( P_4 \land E(P_1 \land P_5) \text{ U } (P_2 \land EG P_5) \right)
\]

*Theorem (Emerson & Halpern, 1985)*

- Any CTL⁺ formula can be translated in an equivalent CTL formula.
- CTL is as expressive as CTL⁺.
**CTL vs CTL⁺**

\[
\begin{align*}
E( & P_1 U P_2 \land P_3 U P_4 \land G P_5 ) \equiv \\
E( & P_1 \land P_3 \land P_5 ) U \left( P_2 \land E( P_3 \land P_5 ) U (P_4 \land EG P_5 ) \right) \lor \\
E( & P_1 \land P_3 \land P_5 ) U \left( P_4 \land E( P_1 \land P_5 ) U (P_2 \land EG P_5 ) \right)
\end{align*}
\]

**Theorem (Emerson & Halpern, 1985)**

- Any CTL⁺ formula can be translated in an equivalent CTL formula.
- CTL is as expressive as CTL⁺.
**CTL vs CTL**

But:

**Theorem (Wilke, 1999 and Adler & Immerman, 1999)**

*CTL* $^+$ is exponentially more succinct than *CTL*.

Any *CTL* formula equivalent to $\Phi_n = E(FP_1 \land \ldots \land FP_n)$ is of length $\Omega(2^n/\sqrt{n})$.

- For any $\varphi \in *CTL*$, there exist an Alternating Tree Aut. $A_{\varphi}$ recognizing the $Mod(\varphi)$ s.t. $|A_{\varphi}|$ is linear in $|\varphi|$.
- Every ATA recognizing $Mod(\Phi_n)$ has at least $\left(\begin{array}{c} n \\ \lceil n/2 \rceil \end{array}\right)$ states
**CTL vs CTL**

But:

**Theorem (Wilke, 1999 and Adler & Immerman, 1999)**

*CTL*\(^+\) is exponentially more succinct than *CTL*.

Any *CTL* formula equivalent to \(\Phi_n = E(F P_1 \land \ldots \land F P_n)\) is of length \(\Omega(2^n/\sqrt{n})\).

- For any \(\varphi \in CTL\), there exist an Alternating Tree Aut. \(A_\varphi\) recognizing the *Mod*(\(\varphi\)) s.t. \(|A_\varphi|\) is linear in \(|\varphi|\).
- Every ATA recognizing *Mod*(\(\Phi_n\)) has at least \(\left\lfloor \frac{n}{2} \right\rfloor\) states.
Outline

8 Distinguishing power
9 Comparison with $LTL$
10 $CTL^*$

11 Expressivity of the fragments of $CTL^*$
   - $UB$ and $UB^+$
   - $CTL$ vs $CTL^+$
   - $ECTL$ vs $CTL$
   - $ECTL^+$, $CTL^*$, and beyond
**CTL vs ECTL**

\[ ECTL = \mathcal{B}(X, U, F) \].

\( ECTL \) formulae are built from AP, boolean operators and the following modalities: \( EX, EU, AU, E\bar{F} \) and \( A\bar{F} \).

The modalities \( E\bar{F} \) and \( A\bar{F} \) have been introduced to express fairness properties.

**Theorem**

\[
\begin{align*}
A\bar{F} \varphi & \quad \text{def} \quad AGF \varphi \equiv AGAF \varphi \\
\end{align*}
\]

But \( E\bar{F} \varphi \equiv EGF \varphi \neq EGEF \varphi \)

\((E\bar{F} \varphi \Rightarrow EGEF \varphi \quad \text{but} \quad E\bar{F} \varphi \not\equiv EGEF \varphi)\)

\( E\bar{F} P \) cannot be expressed in \( CTL \)…
CTL vs ECTL

$$ECTL = B(X, U, \bar{F}).$$

ECTL formulae are built from AP, boolean operators and the following modalities: $E X$, $E U$, $A U$, $EF$ and $AF$.

The modalities $EF$ and $AF$ have been introduced to express fairness properties.

**Theorem**

$$AF \varphi \overset{\text{def}}{=} AGF \varphi \equiv AGAF \varphi$$

But $EF \varphi \equiv EGF \varphi \neq EGEF \varphi$

$$EF \varphi \Rightarrow EGEF \varphi \quad \text{but} \quad EF \varphi \nRightarrow EGEF \varphi$$

$EF P$ cannot be expressed in CTL...
**CTL vs ECTL**

\[
ECTL = \mathcal{B}(X, U, \mathcal{F})
\]

ECTL formulae are built from AP, boolean operators and the following modalities: \( EX, EU, AU, EF \) and \( AF \).

The modalities \( EF \) and \( AF \) have been introduced to express fairness properties.

**Theorem**

\[
AF \varphi \quad \overset{\text{def}}{=} \quad AGF \varphi \quad \equiv \quad AGAF \varphi
\]

But \( EF \varphi \equiv EGF \varphi \neq EGEF \varphi \)

\[
\left( EF \varphi \Rightarrow EGEF \varphi \quad \text{but} \quad EF \varphi \nRightarrow EGEF \varphi \right)
\]

\( EF \ P \) cannot be expressed in \( CTL \). . .
**CTL vs ECTL**

$ECTL = B(\mathbf{X}, \mathbf{U}, \mathbf{F})$.

$ECTL$ formulae are built from $A\mathbf{P}$, boolean operators and the following modalities: $E\mathbf{X}$, $E\mathbf{U}$, $A\mathbf{U}$, $E\mathbf{F}$ and $A\mathbf{F}$.

The modalities $E\mathbf{F}$ and $A\mathbf{F}$ have been introduced to express fairness properties.

**Theorem**

$$A\mathbf{F} \varphi \overset{\text{def}}{=} A\mathbf{G} F \varphi \equiv A\mathbf{G} A F \varphi$$

But $E\mathbf{F} \varphi \equiv E\mathbf{G} F \varphi \not\equiv E\mathbf{G} E F \varphi$

$$\left( E\mathbf{F} \varphi \Rightarrow E\mathbf{G} E F \varphi \text{ but } E\mathbf{F} \varphi \not\equiv E\mathbf{G} E F \varphi \right)$$

$E\mathbf{F} P$ cannot be expressed in $CTL$...
\[ \forall i \geq 1, \quad \alpha_i \not\models \text{EF} P \text{ and } \alpha'_i \models \text{EF} P \]

\[ \forall i \geq 1, \forall \varphi \in \text{CTL}, \ |\varphi| \leq i \quad \Rightarrow \quad \{ \alpha_i \models \varphi \iff \alpha'_i \models \varphi \}
\beta_i \models \varphi \iff \beta'_i \models \varphi \]
**EF** \( P \) and **CTL** (Emerson & Halpern, 1986)

\[
\forall i \geq 1, \quad \alpha_i \not\models \mathbf{EF} P \quad \text{and} \quad \alpha'_i \models \mathbf{EF} P
\]

\[\forall i \geq 1, \ \forall \varphi \in \text{CTL}, \ |\varphi| \leq i \quad \Rightarrow \quad \left\{ \begin{array}{ll}
\alpha_i \models \varphi & \iff \alpha'_i \models \varphi \\
\beta_i \models \varphi & \iff \beta'_i \models \varphi
\end{array} \right.\]
**EF** $P$ and **CTL** (Emerson & Halpern, 1986)

\[
\forall i \geq 1, \quad \alpha_i \not\models \text{EF} \ P \quad \text{and} \quad \alpha'_i \models \text{EF} \ P
\]

\[
\forall i \geq 1, \forall \varphi \in \text{CTL}, \ |\varphi| \leq i \quad \Rightarrow \quad \left\{ \begin{array}{c}
\alpha_i \models \varphi \iff \alpha'_i \models \varphi \\
\beta_i \models \varphi \iff \beta'_i \models \varphi
\end{array} \right.
\]
$\forall i \geq 1, \ alpha_i \not\models^{\infty} EF P$ and $\alpha'_i \models^{\infty} EF P$

$\forall i \geq 1, \forall \varphi \in CTL, |\varphi| \leq i \Rightarrow \begin{cases} \alpha_i \models \varphi \iff \alpha'_i \models \varphi \\ \beta_i \models \varphi \iff \beta'_i \models \varphi \end{cases}$
$\mathbf{EF} P$ and $CTL$ (Emerson & Halpern, 1986)

$\forall i \geq 1$, $\alpha_i \not\models \mathbf{EF} P$ and $\alpha'_i \models \mathbf{EF} P$

$\forall i \geq 1$, $\forall \varphi \in CTL$, $|\varphi| \leq i \Rightarrow \left\{ \begin{array}{ll} \alpha_i \models \varphi \iff \alpha'_i \models \varphi \\ \beta_i \models \varphi \iff \beta'_i \models \varphi \end{array} \right.$
$\mathcal{EF} \ P$ and $CTL$ (Emerson & Halpern, 1986)

\[
\begin{align*}
\forall i \geq 1, \quad & \alpha_i \not\models \mathcal{EF} \ P \quad \text{and} \quad \alpha'_i \models \mathcal{EF} \ P \\
\forall i \geq 1, \forall \varphi \in CTL, \ |\varphi| \leq i \quad & \Rightarrow \ \left\{ \begin{array}{cl}
\alpha_i \models \varphi & \iff \alpha'_i \models \varphi \\
\beta_i \models \varphi & \iff \beta'_i \models \varphi
\end{array} \right.
\end{align*}
\]
Outline

8 Distinguishing power

9 Comparison with $LTL$

10 $CTL^*$

11 Expressivity of the fragments of $CTL^*$
   - $UB$ and $UB^+$
   - $CTL$ vs $CTL^+$
   - $ECTL$ vs $CTL$
   - $ECTL^+$, $CTL^*$, and beyond
**ECTL⁺, CTL⁺ and beyond (Emerson, 1990)**

**Theorem (Emerson & Halpern, 1986)**

\[ E(\infty F P_1 \land \infty F P_2) \text{ cannot be expressed with ECTL.} \]

\[ \Rightarrow \text{ECTL⁺ is strictly more expressive than ECTL.} \]

**Theorem (Emerson & Halpern, 1986)**

\[ EG(P_1 \lor X P_2) \text{ cannot be expressed with ECTL⁺.} \]

\[ \Rightarrow \text{CTL⁺ is strictly more expressive than ECTL⁺.} \]

**Theorem (Wolper, 1983)**

"P holds for every even state" cannot be expressed in CTL⁺.

\[ \Rightarrow \text{Propositional } \mu\text{-calculus is strictly more expressive than CTL⁺.} \]

\[ (\nu Y.P \land E X E X Y) \]

\[ (\nu Y.P \land E X E X Y) \]
\textbf{ECTL}^+, \textit{CTL}^* and beyond (Emerson, 1990)

\textbf{Theorem (Emerson & Halpern, 1986)}

\[ \text{E}(\infty \mathbf{P}_1 \land \infty \mathbf{P}_2) \text{ cannot be expressed with ECTL.} \]
\[ \Rightarrow \text{ECTL}^+ \text{ is strictly more expressive than ECTL.} \]

\textbf{Theorem (Emerson & Halpern, 1986)}

\[ \text{E G}(P_1 \lor X P_2) \text{ cannot be expressed with ECTL}^+. \]
\[ \Rightarrow \text{CTL}^* \text{ is strictly more expressive than ECTL}^+. \]

\textbf{Theorem (Wolper, 1983)}

\[ \text{“P holds for every even state” cannot be expressed in CTL}. \]
\[ \Rightarrow \text{Propositional } \mu \text{-calculus is strictly more expressive than CTL}^*. \]
\[ (\nu Y. P \land EX EX Y) \]
ECTL$^+$, CTL$^*$ and beyond (Emerson, 1990)

Theorem (Emerson & Halpern, 1986)
\[ E(\exists F P_1 \land \exists F P_2) \text{ cannot be expressed with ECTL.} \]
\[ \Rightarrow ECTL^+ \text{ is strictly more expressive than ECTL.} \]

Theorem (Emerson & Halpern, 1986)
\[ EG(P_1 \lor X P_2) \text{ cannot be expressed with ECTL}^+. \]
\[ \Rightarrow CTL^* \text{ is strictly more expressive than ECTL}^+. \]

Theorem (Wolper, 1983)
“P holds for every even state” cannot be expressed in CTL$^*$.
\[ \Rightarrow \text{Propositional } \mu\text{-calculus is strictly more expressive than CTL}^*. \]
\[ (\nu Y.P \land EX EX Y) \]
Expressiveness of Temporal Logics

François Laroussinie and Nicolas Markey

Lab. Specification et Verification
ENS Cachan & CNRS, France

August 3, 2006
Outline

12 BT-temporal logics with Past

13 $CTL^*$ vs Monadic second order logic

14 Automata theory and BT-temporal logics

15 Alternating-time temporal logic
**CTL** + Past

**Definition**

\[
PCTL^* \ni \varphi, \psi ::= P_1 | \ldots | \neg \varphi | \varphi \land \psi | E\varphi_p | A\varphi_p \\
| \varphi S \psi | X^{-1} \varphi \\
PCTL^*_p \ni \varphi_p, \psi_p ::= \varphi | \neg \varphi_p | \varphi_p \land \psi_p | X \varphi_p | \varphi_p U \psi_p
\]

with \( P \in AP \)

**CTL + S, X^{-1}, CTL + F^{-1}, ECTL + S, \ldots**

**PCTL** formulae are interpreted over states with an history.
Structure of the past

In the linear-time case, past and future are symmetric.

In the branching-time case, several choices are possible. Here we consider a past which is:

- **determined:** an history contains the events which already took place. **Ockhamist past.**
  Thus past and future have a different structure.

- **finite:** the studied behavior has a starting point.
- **cumulative:** whenever the system performs some steps, its history becomes richer and longer.

\[ \text{PCTL}^* \text{ formulas are interpreted over finite prefixes: the last state is the current state, the other ones define the history.} \]
Structure of the past

In the linear-time case, past and future are symmetric.

In the branching-time case, several choices are possible. Here we consider a past which is:

- **determined**: an history contains the events which already took place. **Ockhamist past**.
  Thus past and future have a different structure.

- **finite**: the studied behavior has a starting point.

- **cumulative**: whenever the system performs some steps, its history becomes richer and longer.
Structure of the past

In the linear-time case, past and future are symmetric.

In the branching-time case, several choices are possible. Here we consider a past which is:

- **determined**: an history contains the events which already took place. Ockhamist past.
  Thus past and future have a different structure.

- **finite**: the studied behavior has a starting point.

- **cumulative**: whenever the system performs some steps, its history becomes richer and longer.
Structure of the past

In the linear-time case, past and future are symmetric.

In the branching-time case, several choices are possible. Here we consider a past which is:

- **determined**: an history contains the events which already took place. Ockhamist past. Thus past and future have a different structure.
- **finite**: the studied behavior has a starting point.
- **cumulative**: whenever the system performs some steps, its history becomes richer and longer.

*PCTL* formulas are interpreted over finite prefixes:

- the last state is the current state,
- the other ones define the history.
**Semantics**

\[
\sigma \overset{\text{def}}{=} q_1 \cdots q_n
\]

### Definition

<table>
<thead>
<tr>
<th>Expression</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\sigma \models_S P)</td>
<td>iff (P \in l(q_n))</td>
</tr>
<tr>
<td>(\sigma \models_S \varphi \land \psi)</td>
<td>iff (\sigma \models_S \varphi) and (\sigma \models_S \psi)</td>
</tr>
<tr>
<td>(\sigma \models_S \lnot \varphi)</td>
<td>iff (\sigma \not\models_S \varphi)</td>
</tr>
<tr>
<td>(\sigma \models_S \text{E}\varphi_p)</td>
<td>iff (\exists \rho \in \text{Exec}(q_n) \text{ s.t. } \sigma \cdot \rho, n \models_S \varphi_p)</td>
</tr>
<tr>
<td>(\sigma \models_S \text{A}\varphi_p)</td>
<td>iff (\forall \rho \in \text{Exec}(q_n) \text{ we have } \sigma \cdot \rho, n \models_S \varphi_p)</td>
</tr>
<tr>
<td>(\rho, n \models_S \varphi_p \textbf{ U } \psi_p)</td>
<td>iff (\exists i \geq n, \rho, i \models_S \psi_p) and (\forall n \leq j &lt; i, \text{ we have } \rho, j \models_S \varphi_p)</td>
</tr>
<tr>
<td>(\rho, n \models_S \textbf{X} \varphi_p)</td>
<td>iff (\rho, n + 1 \models_S \varphi_p)</td>
</tr>
<tr>
<td>(\sigma \models_S \varphi \textbf{ S } \psi)</td>
<td>iff (\exists 1 \leq i \leq n, \text{ s.t. } \sigma</td>
</tr>
<tr>
<td>(\sigma \models_S \textbf{X}^{-1} \varphi)</td>
<td>iff (n &gt; 1) and (\sigma</td>
</tr>
</tbody>
</table>
Adding $S$ or $X^{-1}$ (Laroussinie & Schnoebelen, 1994)

$L'$ is initially as expressive as $L : \forall \varphi \in L, \exists \varphi' \in L'$, such that for any state $q$ in any KS, we have $q \models \varphi$ iff $q \models \varphi'$.

- $ECTL^+$ is not as expressive as $UB + S$.
  $E(a \lor b U c) U d$ can be expressed in $UB + S$.

- $ECTL^+$ is not as expressive as $UB + X^{-1}$.
  $EG(a \lor Xa) \equiv_i EG(a \lor X^{-1}a \rightarrow X^{-1}\tt)$

PAST may add expressivity!
**Adding S or X⁻¹ (Laroussinie & Schnoebelen, 1994)**

$L'$ is **initially** as expressive as $L$: $\forall \varphi \in L, \exists \varphi' \in L'$, such that for any state $q$ in any KS, we have $q \models \varphi$ iff $q \models \varphi'$.

- $ECTL^+$ is not as expressive as $UB + S$.
  \[ E(a \lor b U c) U d \text{ can be expressed in } UB + S \]

- $ECTL^+$ is not as expressive as $UB + X^{-1}$.
  \[ E G(a \lor Xa) \equiv_i E G(a \lor X^{-1}a \rightarrow X^{-1}t) \]

PAST may add expressivity!
Adding $S$ or $X^{-1}$ (Laroussinie & Schnoebelen, 1994)

$L'$ is initially as expressive as $L$: $\forall \varphi \in L, \exists \varphi' \in L'$, such that for any state $q$ in any KS, we have $q \models \varphi$ iff $q \models \varphi'$.

- $ECTL^+$ is not as expressive as $UB + S$.
  $E(a \lor b U c) U d$ can be expressed in $UB + S$

- $ECTL^+$ is not as expressive as $UB + X^{-1}$.
  $EG(a \lor X a) \equiv; EG(a \lor X^{-1} a \rightarrow X^{-1} tt)$

PAST may add expressivity!
Adding $S$ or $X^{-1}$ (Laroussinie & Schnoebelen, 1994)

$L'$ is initially as expressive as $L$ : $\forall \varphi \in L, \exists \varphi' \in L'$, such that for any state $q$ in any KS, we have $q \models \varphi$ iff $q \models \varphi'$.

- $ECTL^+$ is not as expressive as $UB + S$.
  $E(a \lor b \mathbin{U} c) \mathbin{U} d$ can be expressed in $UB + S$

- $ECTL^+$ is not as expressive as $UB + X^{-1}$.
  $EG(a \lor X a) \equiv_i EG(a \lor X^{-1} a \rightarrow X^{-1} tt)$

PAST may add expressivity!
Adding $F^{-1}$

$CTL + F^{-1}$ can be **weakly** separated.

$F^{-1} E F \ P$ cannot be (fully) separated.

**Definition**

A formula is **weakly separated** when no past-modalities occur in the scope of a future-modality.

**Theorem (Laroussinie & Schnoebelen, 1994)**

Any $CTL + F^{-1}$ formula can be separated.
Any $B(X, X^{-1}, S)$ formula can be separated.

(based on separation rules of (Gabbay, 1987))
Adding $F^{-1}$

$CTL + F^{-1}$ can be weakly separated.

$F^{-1} E F P$ cannot be (fully) separated.

**Definition**

A formula is *weakly separated* when no past-modalities occur in the scope of a future-modality.

**Theorem (Laroussinie & Schnoebelen, 1994)**

*Any $CTL + F^{-1}$ formula can be separated.*  
*Any $B(X, X^{-1}, S)$ formula can be separated.*

(based on separation rules of (Gabbay, 1987))
Example of separation

\[ E(P_1 \land F^{-1}P_2) \cup (P_3 \land F^{-1}P_4) \equiv \\
(P_3 \land F^{-1}P_4) \lor \\
(F^{-1}P_2 \land \ldots) \\
F^{-1}P_4 \land EP_1 \cup P_3 \lor \\
EP_1 \cup (P_1 \land P_4 \land EP_1 \cup P_3) \]
Example of separation

\[ E(P_1 \land F^{-1}P_2) \cup (P_3 \land F^{-1}P_4) \equiv \]
\[ (P_3 \land F^{-1}P_4) \lor \]
\[ (F^{-1}P_2 \land \ldots) \]
\[ F^{-1}P_4 \land EP_1 \cup P_3 \lor \]
\[ EP_1 \cup (P_1 \land P_4 \land EP_1 \cup P_3) \]
Example of separation

\[
\begin{align*}
E(P_1 \land F^{-1}P_2) \cup (P_3 \land F^{-1}P_4) & \equiv \\
(P_3 \land F^{-1}P_4) \lor \\
(F^{-1}P_2 \land \ldots) \\
& \quad \land F^{-1}P_4 \land EP_1 \cup P_3 \lor \\
& \quad EP_1 \cup (P_1 \land P_4 \land EP_1 \cup P_3)
\end{align*}
\]
Example of separation

\[ E(P_1 \land F^{-1} P_2) \cup (P_3 \land F^{-1} P_4) \equiv (P_3 \land F^{-1} P_4) \lor \left( F^{-1} P_2 \land \ldots \right) \]
\[ F^{-1} P_4 \land EP_1 \cup P_3 \lor EP_1 \cup (P_1 \land P_4 \land EP_1 \cup P_3) \]
Separation and initial equivalence

If a logic can be weakly separated, it is initially equivalent to its pure-future fragment.

Let $\Phi$ be a weakly separated formula: every past-modality in $\Phi$ occurs at the root of $\Phi$ (possibly in the scope of boolean connectives) or in the scope of another past-modality.

We have:

1. $\varphi \mathbf{S} \psi \equiv_i \psi$  
2. $\mathbf{X}^{-1} \psi \equiv_i \bot$

By applying rules (1) and (2), we can easily deduce that $\Phi$ is initially equivalent to some pure-future formula.

Theorem (Hafer & Thomas, 1987)

$PCTL^*$ is initially equivalent to $CTL^*$.

(based on Kamp’s theorem)
BTL with $F^{-1}$ (Laroussinie & Schnoebelen, 1994)

The following results hold for initial equivalence.

- $B(X)$ is as expressive as $B(X, X^{-1}, S)$.

- $CTL$ is as expressive as $CTL^+ + F^{-1}$.
  (but $CTL^+ + F^{-1}$ is exponentially more succinct)

- $ECTL^+$ is as expressive as $ECTL^+ + F^{-1}$.

- $ECTL + F^{-1}$ is strictly more expressive than $ECTL$.
  ($\mathsf{EF}(a \land G^{-1} b)$ cannot be expressed in $ECTL$)

- $ECTL + F^{-1}$ is strictly less expressive than $ECTL^+$.
  ($\mathsf{E}(\mathsf{F} a \land \mathsf{F} b)$ cannot be expressed in $ECTL + F^{-1}$)
BTL with $F^{-1}$ (Laroussinie & Schnoebelen, 1994)

The following results hold for initial equivalence.

- $B(X)$ is as expressive as $B(X, X^{-1}, S)$.

- $CTL$ is as expressive as $CTL^+ + F^{-1}$.
  (but $CTL^+ + F^{-1}$ is exponentially more succinct)

- $ECTL^+$ is as expressive as $ECTL^+ + F^{-1}$.

- $ECTL + F^{-1}$ is strictly more expressive than $ECTL$.
  ($EF (a \land G^{-1} b)$ cannot be expressed in $ECTL$)

- $ECTL + F^{-1}$ is strictly less expressive than $ECTL^+$.
  ($E(\infty F a \land \infty F b)$ cannot be expressed in $ECTL + F^{-1}$)
BTL with $F^{-1}$ (Laroussinie & Schnoebelen, 1994)

The following results hold for initial equivalence.

- $B(X)$ is as expressive as $B(X, X^{-1}, S)$.

- $CTL$ is as expressive as $CTL^+ + F^{-1}$.
  (but $CTL^+ + F^{-1}$ is exponentially more succinct)

- $ECTL^+$ is as expressive as $ECTL^+ + F^{-1}$.

- $ECTL + F^{-1}$ is strictly more expressive than $ECTL$.
  ($E F (a \land G^{-1} b)$ cannot be expressed in $ECTL$)

- $ECTL + F^{-1}$ is strictly less expressive than $ECTL^+$.
  ($E(\infty F a \land \infty F b)$ cannot be expressed in $ECTL + F^{-1}$)
**BTL with $\mathbf{F}^{-1}$ (Laroussinie & Schnoebelen, 1994)**

The following results hold for **initial** equivalence.

- $\mathcal{B}(X)$ is as expressive as $\mathcal{B}(X, X^{-1}, S)$.
- $CTL$ is as expressive as $CTL^+ + \mathbf{F}^{-1}$.
  (but $CTL^+ + \mathbf{F}^{-1}$ is exponentially more succinct)
- $ECTL^+$ is as expressive as $ECTL^+ + \mathbf{F}^{-1}$.
- $ECTL + \mathbf{F}^{-1}$ is strictly more expressive than $ECTL$.
  ($\mathbf{E}\mathbf{F} (a \land G^{-1} b)$ cannot be expressed in $ECTL$)
- $ECTL + \mathbf{F}^{-1}$ is strictly less expressive than $ECTL^+$.
  ($\mathbf{E}(\mathbf{F} a \land \mathbf{F} b)$ cannot be expressed in $ECTL + \mathbf{F}^{-1}$)
BTL with $F^{-1}$ (Laroussinie & Schnoebelen, 1994)

The following results hold for initial equivalence.

- $\mathcal{B}(X)$ is as expressive as $\mathcal{B}(X, X^{-1}, S)$.

- $CTL$ is as expressive as $CTL^+ + F^{-1}$.
  (but $CTL^+ + F^{-1}$ is exponentially more succinct)

- $ECTL^+$ is as expressive as $ECTL^+ + F^{-1}$.

- $ECTL + F^{-1}$ is strictly more expressive than $ECTL$.
  ($E F (a \land G^{-1} b)$ cannot be expressed in $ECTL$)

- $ECTL + F^{-1}$ is strictly less expressive than $ECTL^+$.
  ($E(\infty F a \land F b)$ cannot be expressed in $ECTL + F^{-1}$)
Outline

12 BT-temporal logics with Past

13 $CTL^*$ vs Monadic second order logic

14 Automata theory and BT-temporal logics

15 Alternating-time temporal logic
Relation with other formalisms

Relationship between linear-time temporal logics and first-order logic or with automata theory is well known.

What about branching-time temporal logics?

Need a formalism able to quantify over paths and not only on positions along a path.
Monadic Second Order Logic

Consider the monadic second order logic MSOL \((<, \Sigma)\) to express properties of \(\Sigma\)-labeled trees. It contains:

- individual variables \(x, y, z, \ldots\) (for the nodes)
- set variables \(X, Y, Z, \ldots\) (for set of nodes)
- predicate constants \(P_a\) for \(a \in \Sigma\)
- \(x = y, x < y, x \in X, x \in P_a\)
- \(\land, \lor, \neg, \exists, \forall\)

\((\text{FOL} (<, \Sigma)\) is the restriction without set variables.\)

The monadic path logic \(\text{MPL}\) is the restriction of MSOL where the interpretation of set variables \(X\) ranges only over paths.
Monadic Second Order Logic

Consider the monadic second order logic MSOL \((<, \Sigma)\) to express properties of \(\Sigma\)-labeled trees. It contains:

- individual variables \(x, y, z, \ldots\) (for the nodes)
- set variables \(X, Y, Z, \ldots\) (for set of nodes)
- predicate constants \(P_a\) for \(a \in \Sigma\)
- \(x = y, x < y, x \in X, x \in P_a\)
- \(\land, \lor, \neg, \exists, \forall\)

(FOL \((<, \Sigma)\) is the restriction without set variables.)

The Monadic path logic MPL is the restriction of MSOL where the interpretation of set variables \(X\) ranges only over paths.
Monadic Second Order Logic

Consider the monadic second order logic MSOL ($<, \Sigma$) to express properties of $\Sigma$-labeled trees. It contains:

- individual variables $x, y, z, \ldots$ (for the nodes)
- set variables $X, Y, Z, \ldots$ (for set of nodes)
- predicate constants $P_a$ for $a \in \Sigma$

And $x = y, x < y, x \in X, x \in P_a$

And $\land, \lor, \neg, \exists, \forall$

(FOL ($<, \Sigma$) is the restriction without set variables.)

The Monadic path logic MPL is the restriction of MSOL where the interpretation of set variables $X$ ranges only over paths.
Monadic Second Order Logic

Consider the monadic second order logic MSOL ($<, \Sigma$) to express properties of $\Sigma$-labeled trees. It contains:

- individual variables $x, y, z, \ldots$ (for the nodes)
- set variables $X, Y, Z, \ldots$ (for set of nodes)
- predicate constants $P_a$ for $a \in \Sigma$
- And $x = y$, $x < y$, $x \in X$, $x \in P_a$
- And $\land$, $\lor$, $\neg$, $\exists$, $\forall$

(FOL ($<, \Sigma$) is the restriction without set variables.)

The Monadic path logic MPL is the restriction of MSOL where the interpretation of set variables $X$ ranges only over paths.
Monadic Second Order Logic

Consider the monadic second order logic MSOL \((<, \Sigma)\) to express properties of \(\Sigma\)-labeled trees. It contains:

- individual variables \(x, y, z, \ldots\) (for the nodes)
- set variables \(X, Y, Z, \ldots\) (for set of nodes)
- predicate constants \(P_a\) for \(a \in \Sigma\)
- And \(x = y, x < y, x \in X, x \in P_a\)
- And \(\land, \lor, \neg, \exists, \forall\)

(FOL \((<, \Sigma)\) is the restriction without set variables.)

The monadic path logic MPL is the restriction of MSOL where the interpretation of set variables \(X\) ranges only over paths.
Monadic Second Order Logic

Consider the monadic second order logic MSOL \(<, \Sigma\) to express properties of \(\Sigma\)-labeled trees. It contains:

- individual variables \(x, y, z, \ldots\) (for the nodes)
- set variables \(X, Y, Z, \ldots\) (for set of nodes)
- predicate constants \(P_a\) for \(a \in \Sigma\)
- And \(x = y, x < y, x \in X, x \in P_a\)
- And \(\land, \lor, \neg, \exists, \forall\)

(FOL \(<, \Sigma\) is the restriction without set variables.)

The Monadic path logic MPL is the restriction of MSOL where the interpretation of set variables \(X\) ranges only over paths.
Monadic Second Order Logic

Consider the monadic second order logic MSOL \((<, \Sigma)\) to express properties of \(\Sigma\)-labeled trees. It contains:

- Individual variables \(x, y, z, \ldots\) (for the nodes)
- Set variables \(X, Y, Z, \ldots\) (for set of nodes)
- Predicate constants \(P_a\) for \(a \in \Sigma\)
- And \(x = y, x < y, x \in X, x \in P_a\)
- And \(\land, \lor, \neg, \exists, \forall\)

(FOL \((<, \Sigma)\) is the restriction without set variables.)

The Monadic path logic \(MPL\) is the restriction of MSOL where the interpretation of set variables \(X\) ranges only over paths.
Example of MSOL formulas

\( P_1 \): characterizing the set of even states in \( T \)

We have to specify that:

- the root belongs to \( X \) (\( \exists x \in X. \forall y. x < y \lor x = y \) )
- If \( y \) is a succ. of \( x \in X \) (\( x < y \land \forall u : \neg(x < u < y) \) ):
  - \( y \) is not in \( X \)
  - any successor of \( y \) is in \( X \)

Thus \( P_1 \) can be written:

\[
\exists X. \left( \exists x_0 \in X. \forall y. x_0 < y \lor x_0 = y \right) \land \left( \forall x \in X \forall y \left( (x < y \land \forall u : \neg(x < u < y)) \Rightarrow (y \not\in X \land \forall z (y < z \land \forall u : \neg(y < u < z)) \Rightarrow z \in X) \right) \right)
\]
Example of MSOL formulas

\( P_2 \): characterizing an infinite path in \( T \)

We have to specify that:

- the root is in the set \( X \).
- any two nodes in \( X \) are \( <, > \) or \( = \).
- any node in \( X \) has a successor in \( X \).

Thus:

\[
\exists X. \left( \exists x_0 \in X. \forall y : x_0 < y \lor x_0 = y \right) \land \left( \forall x, y \in X : (x < y \lor y < x \lor x = y) \right) \land \left( \forall x \in X : \exists y \in X. (x < y \land \forall u : \neg(x < u < y)) \right)
\]
From *CTL* to *MPL*

**Theorem**

For any \( \varphi \in \text{CTL}^* \), there exists \( F_\varphi \in \text{MPL} \) s.t. \( \varphi \equiv F_\varphi \)

\( F_\varphi \) is defined by induction.

A formula \( F_\varphi(x) \) is associated with every state formula \( \varphi \).

A formula \( F_{\varphi_p}(X) \) is associated with every path formula \( \varphi_p \).

For any tree \( T \) and any node \( s \in T \) and any path \( \rho \) in \( T \), we have:

\[
\begin{align*}
    s \models_T \varphi & \iff (T, s) \models F_\varphi(x) \\
    \rho \models_T \varphi_p & \iff (T, \rho) \models F_{\varphi_p}(X)
\end{align*}
\]

i.e. \( T \models F_\varphi(x \leftarrow s) \) and \( T \models F_\varphi(X \leftarrow \rho) \).
From $CTL^*$ to $MPL$

**Theorem**

For any $\varphi \in CTL^*$, there exists $F_{\varphi} \in MPL$ s.t. $\varphi \equiv F_{\varphi}$

$F_{\varphi}$ is defined by induction.
A formula $F_{\varphi}(x)$ is associated with every state formula $\varphi$.
A formula $F_{\varphi_p}(X)$ is associated with every path formula $\varphi_p$.

For any tree $T$ and any node $s \in T$ and any path $\rho$ in $T$, we have:

$s \models_T \varphi \iff (T, s) \models F_{\varphi}(x)$

$\rho \models_T \varphi_p \iff (T, \rho) \models F_{\varphi_p}(X)$

i.e. $T \models F_{\varphi}(x \leftarrow s)$ and $T \models F_{\varphi}(X \leftarrow \rho)$. 
From *CTL* to *MPL*

Definition of $F_\varphi$ (Hafer & Thomas, 1987):

- $F_a(x) \overset{\text{def}}{=} x \in P_a$
- $F_{E\varphi_p}(x) \overset{\text{def}}{=} \exists Y. ("Y starts at x" \land F_{\varphi_p}(Y))$
  
  with " Y starts at x" $\overset{\text{def}}{=} x \in Y \land \forall y \in Y(x < y \lor x = y)$
- $F_{X\varphi_p}(Z) \overset{\text{def}}{=} \exists z \exists y \exists Y. ("Y \subset Z" \land$
  "Y starts at y" \land "y is a successor of z" \land $F_{\varphi_p}(Y))$
  
  with: " Y \subset Z" $\overset{\text{def}}{=} \forall u \in Y.y \in Z$
  
  and: "y is a successor of z" $\overset{\text{def}}{=} z < y \land \forall u. \neg(z < u < y)$
- $F_{\varphi_p} \cup_{\psi_p}(Z) \overset{\text{def}}{=} \exists Y. ("Y \subset Z" \land F_{\psi_p}(Y) \land (\forall Y'. Y \subset Y' \land Y' \subset Z \Rightarrow F_{\varphi_p}(Y')))
From $CTL^*$ to $MPL$

Definition of $F_\varphi$ (Hafer & Thomas, 1987):

- $F_a(x) \overset{\text{def}}{=} x \in P_a$
- $F_{E\varphi_p}(x) \overset{\text{def}}{=} \exists Y. \left( "Y \text{ starts at } x" \land F_{\varphi_p}(Y) \right)$
  
  with "$Y \text{ starts at } x" \overset{\text{def}}{=} x \in Y \land \forall y \in Y (x < y \lor x = y)$

- $F_{X\varphi_p}(Z) \overset{\text{def}}{=} \exists z \exists y \exists Y. \left( "Y \subset Z" \land "Y \text{ starts at } y" \land "y \text{ is a successor of } z" \land F_{\varphi_p}(Y) \right)$
  
  with: "$Y \subset Z" \overset{\text{def}}{=} \forall u \in Y. y \in Z
  
  and: "y \text{ is a successor of } z" \overset{\text{def}}{=} z < y \land \forall u. \neg (z < u < y)$

- $F_{\varphi_p \cup \psi_p}(Z) \overset{\text{def}}{=} \exists Y. \left( "Y \subset Z" \land F_{\psi_p}(Y) \land (\forall Y' Y \subset Y' \land Y' \subset Z \Rightarrow F_{\varphi_p}(Y')) \right)$
From $\text{CTL}^*$ to $\text{MPL}$

Definition of $F_{\varphi}$ (Hafer & Thomas, 1987):

- $F_a(x) \overset{\text{def}}{=} x \in P_a$
- $F_{E\varphi_p}(x) \overset{\text{def}}{=} \exists Y. \left( \text{"Y starts at x”} \land F_{\varphi_p}(Y) \right)$
  with $\text{"Y starts at x”} \overset{\text{def}}{=} x \in Y \land \forall y \in Y (x < y \lor x = y)$
- $F_{X\varphi_p}(Z) \overset{\text{def}}{=} \exists z \exists y \exists Y. \left( \text{"Y ⊂ Z”} \land \text{"Y starts at y”} \land \text{"y is a successor of z”} \land F_{\varphi_p}(Y) \right)$
  with: $\text{"Y ⊂ Z”} \overset{\text{def}}{=} \forall u \in Y. y \in Z$
  and: $\text{"y is a successor of z”} \overset{\text{def}}{=} z < y \land \forall u. \lnot (z < u < y)$
- $F_{\varphi_p \cup \psi_p}(Z) \overset{\text{def}}{=} \exists Y. \left( \text{"Y ⊂ Z”} \land F_{\psi_p}(Y) \land (\forall Y' Y \subset Y' \land Y' \subset Z \Rightarrow F_{\varphi_p}(Y')) \right)$
From $CTL^*$ to $MPL$

Definition of $F\varphi$ (Hafer & Thomas, 1987):

- $F_a(x) \overset{\text{def}}{=} x \in P_a$
- $F_E\varphi_p(x) \overset{\text{def}}{=} \exists Y.\left( "Y starts at x" \land F\varphi_p(Y) \right)$
  
  with $"Y starts at x" \overset{\text{def}}{=} x \in Y \land \forall y \in Y (x < y \lor x = y)$
- $F_X\varphi_p(Z) \overset{\text{def}}{=} \exists z \exists y \exists Y.\left( "Y \subset Z" \land "Y starts at y" \land "y is a successor of z" \land F\varphi_p(Y) \right)$
  
  with: $"Y \subset Z" \overset{\text{def}}{=} \forall u \in Y. y \in Z$
  and: $"y is a successor of z" \overset{\text{def}}{=} z < y \land \forall u. \neg (z < u < y)$
- $F\varphi_p \cup \psi_p(Z) \overset{\text{def}}{=} \exists Y.\left( "Y \subset Z" \land F\psi_p(Y) \land (\forall Y' Y \subset Y' \land Y' \subset Z \Rightarrow F\varphi_p(Y')) \right)$
From *MPL* to *CTL* *

**Theorem (Hafer & Thomas, 1987)**

*CTL* ∗ is expressively equivalent to the *MPL* sentences over full binary trees.

But this is not true in general!

*CTL* ∗ respects the bisimulation equivalence but *MPL* sentences do not:

\[ Φ \overset{\text{def}}{=} \exists x \exists y \left( \neg (x < y \lor x = y \lor y < x) \land x \in P_a \land y \in P_a \right) \]

Φ expresses that there are two incomparable states satisfying a.
From *MPL* to *CTL* *

**Theorem (Hafer & Thomas, 1987)**

*CTL* * is expressively equivalent to the *MPL* sentences over full binary trees.

But this is not true in general!

*CTL* * respects the bisimulation equivalence but *MPL* sentences do not:

\[
\Phi \overset{\text{def}}{=} \exists x \exists y \left( \neg (x < y \lor x = y \lor y < x) \land x \in P_a \land y \in P_a \right)
\]

*Φ* expresses that there are two incomparable states satisfying *a*.
From $MPL$ to $CTL^*$

Theorem (Moller & Rabinovich, 1999)

$CTL^*$ is expressively equivalent to the $MPL$ sentences which respect bisimulation equivalence.

Main ideas of the proof...

- Let $MPL_n$ and $FOL_n$ be the restrictions to formulas with a quantifier depth less than $n$.

- $T \equiv_n T'$ iff $T$ and $T'$ satisfy the same $MPL_n$ formulas.

- $\equiv_n$ defines finitely many equivalence classes: $C_1, \ldots, C_m$.

- Given a (in)finite path $\rho$ in $T$, $\nu_n(\rho)$ is a word over an extended alphabet $\Sigma'$ that describes precisely $\rho$ w.r.t. $MPL_n$.

Idea: for every state along $\rho$, we store its letter (in $\Sigma$) and the equivalence classes of all its subtrees. ($\Sigma' = \Sigma \times \mathcal{P}([1, \ldots, m])$).
From \textit{MPL} to \textit{CTL}*

\begin{itemize}
  \item Let $MPL_n$ and $FOL_n$ be the restrictions to formulas with a quantifier depth less than $n$.
  \item $T \equiv_n T'$ iff $T$ and $T'$ satisfy the same $MPL_n$ formulas.
  \item $\equiv_n$ defines finitely many equivalence classes: $C_1, \ldots, C_m$.
  \item Given a (in)finite path $\rho$ in $T$, $\nu_n(\rho)$ is a word over an extended alphabet $\Sigma'$ that describes precisely $\rho$ w.r.t. $MPL_n$.
\end{itemize}

Idea: for every state along $\rho$, we store its letter (in $\Sigma$) and the equivalence classes of all its subtrees.

( $\Sigma' = \Sigma \times P(\{1,\ldots, m\})$ ).
From MPL to CTL*

**Theorem (Moller & Rabinovich, 1999)**

*CTL* is expressively equivalent to the MPL sentences which respect bisimulation equivalence.

Main ideas of the proof...

- Let $MPL_n$ and $FOL_n$ be the restrictions to formulas with a quantifier depth less than $n$.

- $T \equiv_n T'$ iff $T$ and $T'$ satisfy the same $MPL_n$ formulas.

- $\equiv_n$ defines finitely many equivalence classes: $C_1, \ldots, C_m$.

- Given a (in)finite path $\rho$ in $T$, $\nu_n(\rho)$ is a word over an extended alphabet $\Sigma'$ that describes precisely $\rho$ w.r.t. $MPL_n$.

**Idea:** for every state along $\rho$, we store its letter (in $\Sigma$) and the equivalence classes of all its subtrees. ($\Sigma' = \Sigma \times \mathcal{P}([1, \ldots, m])$).
From *MPL* to *CTL*

**Theorem (Moller & Rabinovich, 1999)**

*CTL* is expressively equivalent to the *MPL* sentences which respect bisimulation equivalence.

Main ideas of the proof...

- Let *MPL*$_n$ and *FOL*$_n$ be the restrictions to formulas with a quantifier depth less than $n$.
- $T \equiv_n T'$ iff $T$ and $T'$ satisfy the same *MPL*$_n$ formulas.
- $\equiv_n$ defines finitely many equivalence classes: $C_1, \ldots, C_m$.
- Given a (in)finite path $\rho$ in $T$, $\nu_n(\rho)$ is a word over an extended alphabet $\Sigma'$ that describes precisely $\rho$ w.r.t. *MPL*$_n$.

**Idea:** for every state along $\rho$, we store its letter (in $\Sigma$) and the equivalence classes of all its subtrees. ($\Sigma' = \Sigma \times P(\{1, \ldots, m\})$).
From \textit{MPL} to \textit{CTL}*

**Theorem (Moller & Rabinovich, 1999)**

\textit{CTL}* is expressively equivalent to the \textit{MPL} sentences which respect bisimulation equivalence.

Main ideas of the proof...

- Let \textit{MPL}_n and \textit{FOL}_n be the restrictions to formulas with a quantifier depth less than \(n\).
- \(T \equiv_n T'\) iff \(T\) and \(T'\) satisfy the same \textit{MPL}_n formulas.
- \(\equiv_n\) defines finitely many equivalence classes: \(C_1, \ldots, C_m\).
- Given a (in)finite path \(\rho\) in \(T\), \(\nu_n(\rho)\) is a word over an extended alphabet \(\Sigma'\) that describes precisely \(\rho\) w.r.t. \textit{MPL}_n.

\textbf{Idea:} for every state along \(\rho\), we store its letter (in \(\Sigma\)) and the equivalence classes of all its subtrees. 
\( (\Sigma' = \Sigma \times \mathcal{P}(\{1, \ldots, m\}) \).

\(\equiv_n\) defines finitely many equivalence classes: \(C_1, \ldots, C_m\).
From *MPL* to *CTL* *(Moller & Rabinovich, 1999)*

A tree is **wide** if every subtree is reproduced an infinite number of times.  
(Any tree *T* can be transformed into a wide tree *T̅* and *T ≈ T̅*.)

The proof is based on a composition Theorem:

**Theorem (Moller & Rabinovich, 1999)**

For every *MPL* sub formula *F(x)* there is a *FOL* sub formula *α*, s.t. for any wide tree *T*, and any node *s* ∈ *T*, we have:

\[(T, s) \models F(x) \iff \nu_n(\varepsilon_T \rightarrow s) \models \alpha\]

(+ similar result for *F(X)* and *ρ* ∈ *T* . . . )

And use Kamp’s theorem to go from *FOL* to *LTL*: we can translate *F(x)* into Φ*F* ∈ *CTL*°.
From *MPL* to *CTL* *(Moller & Rabinovich, 1999)*

A tree is **wide** if every subtree is reproduced an infinite number of times.
(Any tree \( T \) can be transformed into a wide tree \( \overline{T} \) and \( T \approx \overline{T} \).)

The proof is based on a composition Theorem:

**Theorem (Moller & Rabinovich, 1999)**

For every MPL\(_n\) formula \( F(x) \) there is a FOL\(_n\)(\(<, \Sigma'\)) formula \( \alpha \), s.t. for any wide tree \( T \), and any node \( s \in T \), we have:

\[
(T, s) \models F(x) \iff \nu_n(\varepsilon_T \rightarrow s) \models \alpha
\]

(\(+ \text{ similar result for } F(X) \text{ and } \rho \in T \ldots\) )

And use Kamp’s theorem to go from FOL to \( LTL \): we can translate \( F(x) \) into \( \Phi_F \in CTL^* \).
From *MPL* to *CTL* *(Moller & Rabinovich, 1999)*

Given a *MPL* formula $F$ invariant under bisimulation, then:

$$ T \models F $$

$\iff$ $\overline{T} \models F$

$\iff$ $\overline{T} \models \Phi_F$

$\iff$ $T \models \Phi_F$

$F$ inv. bisim.

$\Phi_F \in CTL^*$, cf previous slide

$T \approx \overline{T}$ and $CTL^*$ resp. $\approx$.

Another result exists for the propositional $\mu$-calculus:

**Theorem (Janin & Walukiewicz, 1996)**

Propositional $\mu$-calculus is expressively equivalent to the MSOL sentences which respect bisimulation equivalence.
Given a *MPL* formula $F$ invariant under bisimulation, then:

\[ T \models F \]
\[ \iff \bar{T} \models F \]
\[ \iff \bar{T} \models \Phi_F \]
\[ \iff T \models \Phi_F \]

$F$ inv. bisim. \[ \Phi_F \in \text{CTL}^*, \text{cf previous slide} \]

Another result exists for the propositional $\mu$-calculus:

**Theorem (Janin & Walukiewicz, 1996)**

Propositional $\mu$-calculus is expressively equivalent to the MSOL sentences which respect bisimulation equivalence.
From *MPL* to *CTL* *(Moller & Rabinovich, 1999)*

Given a *MPL* formula *F* invariant under bisimulation, then:

\[
T \models F \\
\iff \overline{T} \models F \\
\iff \overline{T} \models \Phi_F \\
\iff T \models \Phi_F \\
\]

\(F \text{ inv. bisim.}\)
\(\Phi_F \in CTL^*, \text{ cf previous slide}\)
\(T \approx \overline{T} \text{ and } CTL^* \text{ resp. } \approx.\)

Another result exists for the propositional \(\mu\)-calculus:

**Theorem (Janin & Walukiewicz, 1996)**

 Propositional \(\mu\)-calculus is expressively equivalent to the MSOL sentences which respect bisimulation equivalence.
From *MPL* to *CTL* *(Moller & Rabinovich, 1999)*

Given a *MPL* formula $F$ invariant under bisimulation, then:

$$
T \models F \\
\Leftrightarrow \overline{T} \models F \\
\Leftrightarrow \overline{T} \models \Phi_F \\
\Leftrightarrow T \models \Phi_F
$$

$F$ inv. bisim. 

$\Phi_F \in \text{CTL}^*$, cf previous slide 

$T \approx \overline{T}$ and *CTL* resp. $\approx$.

Another result exists for the propositional $\mu$-calculus:

**Theorem (Janin & Walukiewicz, 1996)**

Propositional $\mu$-calculus is expressively equivalent to the MSOL sentences which respect bisimulation equivalence.
From **MPL** to **CTL** *(Moller & Rabinovich, 1999)*

Given a **MPL** formula $F$ invariant under bisimulation, then:

$$T \models F$$

$\iff$  

$$\overline{T} \models F$$

$\iff$  

$$\overline{T} \models \Phi_F$$

$\iff$  

$$T \models \Phi_F$$

$F$ inv. bisim.  

$\Phi_F \in \text{CTL}^*$, cf previous slide  

$T \approx \overline{T}$ and $\text{CTL}^*$ resp. $\approx$.

Another result exists for the propositional $\mu$-calculus:

**Theorem (Janin & Walukiewicz, 1996)**

Propositional $\mu$-calculus is expressively equivalent to the MSOL sentences which respect bisimulation equivalence.
Additional results

Theorem (Moller & Rabinovich, 2003)

*Counting-CTL* is expressively equivalent to MPL.

New modalities $D^n$

$s \models D^n \varphi$ iff “for at least $n$ different $s \rightarrow s'$, we have $s' \models \varphi$”.

Let $BTL_k$ be the temporal logic defined with the modalities $E \varphi$ with $\varphi$ a first-order future formula with $qd(\varphi) \leq k$.

Theorem (Rabinovich & Schnoebelen, 2000)

$ECTL^+$ and $BTL_2$ have the same expressive power.
Additional results

**Theorem (Moller & Rabinovich, 2003)**

*Counting-CTL*\(^*\) is expressively equivalent to MPL.

New modalities \(D^n\)
\[ s \models D^n \varphi \text{ iff “for at least } n \text{ different } s \rightarrow s', \text{ we have } s' \models \varphi”. \]

Let \(BTL_k\) be the temporal logic defined with the modalities \(E \varphi\) with \(\varphi\) a first-order future formula with \(qd(\varphi) \leq k\).

**Theorem (Rabinovich & Schnoebelen, 2000)**

\(ECTL^+\) and \(BTL_2\) have the same expressive power.
Outline

12 BT-temporal logics with Past

13 $CTL^*$ vs Monadic second order logic

14 Automata theory and BT-temporal logics

15 Alternating-time temporal logic
Automata theory and branching-time logics

For any $\varphi \in LTL$, there exists a Büchi automaton $A_\varphi$ that recognizes the models of $\varphi$.
And $|A_\varphi|$ is in $2^{O(|\varphi|)}$

For any $\varphi \in LTL$, there exists an alternating Büchi automaton $A^a_\varphi$ that recognizes $\mathcal{M}(\varphi)$.
And $|A^a_\varphi|$ is in $O(|\varphi|)$

And for $\varphi \in CTL$?

One can build an Alternating Tree Automaton that recognizes $\mathcal{M}(\varphi)$.

References: (Vardi, 1995), (Kupferman, Vardi, Wolper, 2000), (Wilke, 1999).
Automata theory and branching-time logics

For any $\varphi \in LTL$, there exists a Büchi automaton $A_\varphi$ that recognizes the models of $\varphi$. And $|A_\varphi|$ is in $2^O(|\varphi|)$.

For any $\varphi \in LTL$, there exists an alternating Büchi automaton $A^a_\varphi$ that recognizes $M(\varphi)$. And $|A^a_\varphi|$ is in $O(|\varphi|)$.

And for $\varphi \in CTL$?

One can build an Alternating Tree Automaton that recognizes $M(\varphi)$.

References: (Vardi, 1995), (Kupferman, Vardi, Wolper, 2000), (Wilke, 1999).
Automata theory and branching-time logics

For any $\varphi \in LTL$, there exists a Büchi automaton $A_\varphi$ that recognizes the models of $\varphi$.
And $|A_\varphi|$ is in $2^{O(|\varphi|)}$

For any $\varphi \in LTL$, there exists an alternating Büchi automaton $A^a_\varphi$ that recognizes $M(\varphi)$.
And $|A^a_\varphi|$ is in $O(|\varphi|)$

And for $\varphi \in CTL$?

One can build an Alternating Tree Automaton that recognizes $M(\varphi)$.

References: (Vardi, 1995), (Kupferman, Vardi, Wolper, 2000), (Wilke, 1999).
Automata theory and branching-time logics

For any $\varphi \in LTL$, there exists a Büchi automaton $A_\varphi$ that recognizes the models of $\varphi$. 
And $|A_\varphi|$ is in $2^{O(|\varphi|)}$

For any $\varphi \in LTL$, there exists an alternating Büchi automaton $A^a_\varphi$ that recognizes $M(\varphi)$. 
And $|A^a_\varphi|$ is in $O(|\varphi|)$

And for $\varphi \in CTL$?

One can build an Alternating Tree Automaton that recognizes $M(\varphi)$.

References: (Vardi, 1995), (Kupferman, Vardi, Wolper, 2000), (Wilke, 1999).
Non-deterministic tree automata

Let $D \subseteq \mathbb{N}$ be a finite set of arities. We consider automata on $\Sigma$-labeled leafless $D$-trees. $A = (\Sigma, D, S, s_0, \rho, F)$

- $S$ : a finite set of states, and $s_0 \in S$.
- $F \subseteq S$ : a Büchi acceptance condition.
- $\rho : S \times \Sigma \times D \rightarrow 2^{S^*}$ : a transition function s.t. $\rho(s, a, k)$ is a set of $k$-tuples $(s_1, \ldots, s_k)$.

Let $T = (T, l)$ be a $\Sigma$-labeled $D$-tree. A run $r : T \rightarrow S$ of $A$ on $T$ is an $S$-labeled $D$-tree s.t. $r(\varepsilon) = s_0$

- For any $x$ s.t. $\text{arity}(x) = k$, we have $(r(x \cdot 1), \ldots, r(x \cdot k)) \in \rho(r(x), l(x), k)$
- for any branch $x_1x_2\ldots$, there are infinitely many $i$ s.t. $r(x_i) \in F$

$T(A)$ : set of trees accepted by $A$. 

Non-deterministic tree automata

Let $\mathcal{D} \subset \mathbb{N}$ be a finite set of arities.
We consider automata on $\Sigma$-labeled leafless $\mathcal{D}$-trees.

$A = (\Sigma, \mathcal{D}, S, s_0, \rho, F)$

- $S$ : a finite set of states, and $s_0 \in S$.
- $F \subseteq S$ : a Büchi acceptance condition.
- $\rho : S \times \Sigma \times \mathcal{D} \rightarrow 2^{S^*}$ : a transition function s.t. $\rho(s, a, k)$ is a set of $k$-tuples $(s_1, \ldots, s_k)$.

Let $T = (T, l)$ be a $\Sigma$-labeled $\mathcal{D}$-tree.
A run $r : T \rightarrow S$ of $A$ on $T$ is an $S$-labeled $\mathcal{D}$-tree s.t.

- $r(\varepsilon) = s_0$
- For any $x$ s.t. $arity(x) = k$, we have $\langle r(x \cdot 1), \ldots, r(x \cdot k) \rangle \in \rho(r(x), l(x), k)$
- for any branch $x_1 x_2 \ldots$, there are infinitely many $i$ s.t. $r(x_i) \in F$

$\mathcal{T}(A) :$ set of trees accepted by $A$. 
Example of NDTA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\}) \] with
\[ \rho(s_0, a, 2) = (s_1, s_1), \quad \rho(s_0, b, 2) = (s_0, s_0), \]
\[ \rho(s_1, a, 2) = (s_1, s_1), \quad \rho(s_1, b, 2) = (s_0, s_0). \]
Example of NDTA

\[ A = \{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\} \] with
\[ \rho(s_0, a, 2) = (s_1, s_1), \quad \rho(s_0, b, 2) = (s_0, s_0), \]
\[ \rho(s_1, a, 2) = (s_1, s_1), \quad \rho(s_1, b, 2) = (s_0, s_0). \]
Example of NDTA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\}) \]  
with

\[ \rho(s_0, a, 2) = (s_1, s_1), \quad \rho(s_0, b, 2) = (s_0, s_0), \]
\[ \rho(s_1, a, 2) = (s_1, s_1), \quad \rho(s_1, b, 2) = (s_0, s_0). \]
Example of NDTA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\}) \] with
\[ \rho(s_0, a, 2) = (s_1, s_1), \quad \rho(s_0, b, 2) = (s_0, s_0), \]
\[ \rho(s_1, a, 2) = (s_1, s_1), \quad \rho(s_1, b, 2) = (s_0, s_0). \]
Example of NDTA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\}) \] with
\[ \rho(s_0, a, 2) = (s_1, s_1), \quad \rho(s_0, b, 2) = (s_0, s_0), \]
\[ \rho(s_1, a, 2) = (s_1, s_1), \quad \rho(s_1, b, 2) = (s_0, s_0). \]
Example of NDTA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\}) \]

with

\[ \rho(s_0, a, 2) = (s_1, s_1), \quad \rho(s_0, b, 2) = (s_0, s_0), \]

\[ \rho(s_1, a, 2) = (s_1, s_1), \quad \rho(s_1, b, 2) = (s_0, s_0). \]
Example of NDTA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_1\}) \] with
\[ \rho(s_0, a, 2) = (s_1, s_1), \quad \rho(s_0, b, 2) = (s_0, s_0), \]
\[ \rho(s_1, a, 2) = (s_1, s_1), \quad \rho(s_1, b, 2) = (s_0, s_0). \]

\( A \) recognizes infinite binary trees where any branch contains infinitely many \( a \).
Example of NDTA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0, s_1\}) \] with

\[ \rho(s_0, a, 2) = \rho(s_0, b, 2) = \rho(s_1, a, 2) = \{(s_1, s_0), (s_0, s_1)\} \]

\[ \rho(s_1, b, 2) = \emptyset \]
Example of NDTA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0, s_1\}) \] with
\[ \rho(s_0, a, 2) = \rho(s_0, b, 2) = \rho(s_1, a, 2) = \{(s_1, s_0), (s_0, s_1)\} \]
\[ \rho(s_1, b, 2) = \emptyset \]
Example of NDTA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0, s_1\}) \] with

\[ \rho(s_0, a, 2) = \rho(s_0, b, 2) = \rho(s_1, a, 2) = \{(s_1, s_0), (s_0, s_1)\} \]

\[ \rho(s_1, b, 2) = \emptyset \]
Example of NDTA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0, s_1\}) \] with

\[ \rho(s_0, a, 2) = \rho(s_0, b, 2) = \rho(s_1, a, 2) = \{(s_1, s_0), (s_0, s_1)\} \]

\[ \rho(s_1, b, 2) = \emptyset \]
Example of NDTA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0, s_1\}) \text{ with} \]
\[ \rho(s_0, a, 2) = \rho(s_0, b, 2) = \rho(s_1, a, 2) = \{(s_1, s_0), (s_0, s_1)\} \]
\[ \rho(s_1, b, 2) = \emptyset \]
Example of NDTA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0, s_1\}) \] with
\[ \rho(s_0, a, 2) = \rho(s_0, b, 2) = \rho(s_1, a, 2) = \{(s_1, s_0), (s_0, s_1)\} \]
\[ \rho(s_1, b, 2) = \emptyset \]
Example of NDTA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0, s_1\}) \] with
\[ \rho(s_0, a, 2) = \rho(s_0, b, 2) = \rho(s_1, a, 2) = \{(s_1, s_0), (s_0, s_1)\} \]
\[ \rho(s_1, b, 2) = \emptyset \]

A recognizes infinite binary trees where every node has an immediate successor labeled by \( a \).
Alternating Tree Automata

\( \rho : S \times \Sigma \times \mathcal{D} \rightarrow \mathcal{B}^+(\mathbb{N} \times S). \)

with \( \rho(s, a, k) \in \mathcal{B}^+(\{1, \ldots, k\} \times S). \)

For ex.: \( \rho(s, a, 3) = \left( (1, s_1) \lor (2, s_1) \right) \land (3, s_2) \)

NDTA: \( \rho'(s, a, k) = \bigvee_{(s_1, \ldots, s_k) \in \rho(s, a, k)} (1, s_1) \land (2, s_2) \land \ldots (k, s_k) \)
Alternating Tree Automata

\[ \rho : S \times \Sigma \times \mathcal{D} \rightarrow \mathcal{B}^+(\mathbb{N} \times S). \]
with \( \rho(s, a, k) \in \mathcal{B}^+(\{1, \ldots, k\} \times S). \)

A run on \( \Sigma \)-labeled leafless \( \mathcal{D} \)-tree \((T, l)\) is a \((\mathbb{N}^* \times S)\)-labeled tree \((T_r, l_r)\).
Each node of \( T_r \) corresponds to a node of \( T \).
Label \((x, s)\): a copy of \( A \) reading the node \( x \) of \( T \) in state \( s \).

- \( l_r(\varepsilon) = (\varepsilon, s_0) \)
- If \( y \in T_r \), \( l_r(y) = (x, s) \), \( \text{arity}(x) = k \) and \( \rho(s, l(x), k) = \theta \),

then:
There exists \( Q = \{(c_1, s_1), \ldots, (c_n, s_n)\} \subseteq \{1, \ldots, k\} \times S \) s.t.
\( Q \models \theta \) and \( \forall 1 \leq i \leq n \), we have:
\( y \cdot i \in T_r \) and \( l_r(y \cdot i) = (x \cdot c_i, s_i) \)
Example of ATA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0\}) \quad \text{with} \]
\[ \rho(s_0, a, 2) = ((1, s_1) \lor (2, s_1)) \land (1, s_0) \land (2, s_0) \]
\[ \rho(s_0, b, 2) = (1, s_0) \land (2, s_0) \]
\[ \rho(s_1, a, 2) = (1, s_1) \lor (2, s_1) \quad \rho(s_1, b, 2) = \top \]
Example of ATA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0\}) \] with

\[ \rho(s_0, a, 2) = ((1, s_1) \lor (2, s_1)) \land (1, s_0) \land (2, s_0) \]
\[ \rho(s_0, b, 2) = (1, s_0) \land (2, s_0) \]
\[ \rho(s_1, a, 2) = (1, s_1) \lor (2, s_1) \quad \rho(s_1, b, 2) = \top \]
Example of ATA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0\}) \] with
\[ \rho(s_0, a, 2) = ((1, s_1) \lor (2, s_1)) \land (1, s_0) \land (2, s_0) \]
\[ \rho(s_0, b, 2) = (1, s_0) \land (2, s_0) \]
\[ \rho(s_1, a, 2) = (1, s_1) \lor (2, s_1) \quad \rho(s_1, b, 2) = \top \]
Example of ATA

\[
A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0\}) \text{ with }
\]
\[
\rho(s_0, a, 2) = ((1, s_1) \lor (2, s_1)) \land (1, s_0) \land (2, s_0)
\]
\[
\rho(s_0, b, 2) = (1, s_0) \land (2, s_0)
\]
\[
\rho(s_1, a, 2) = (1, s_1) \lor (2, s_1) \quad \rho(s_1, b, 2) = \top
\]
Example of ATA

\[ A = (\{ a, b \}, \{ 2 \}, \{ s_0, s_1 \}, s_0, \rho, \{ s_0 \}) \text{ with} \]

\[ \rho(s_0, a, 2) = ((1, s_1) \lor (2, s_1)) \land (1, s_0) \land (2, s_0) \]

\[ \rho(s_0, b, 2) = (1, s_0) \land (2, s_0) \]

\[ \rho(s_1, a, 2) = (1, s_1) \lor (2, s_1) \quad \rho(s_1, b, 2) = \top \]
Example of ATA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0\}) \text{ with} \]
\[ \rho(s_0, a, 2) = ((1, s_1) \lor (2, s_1)) \land (1, s_0) \land (2, s_0) \]
\[ \rho(s_0, b, 2) = (1, s_0) \land (2, s_0) \]
\[ \rho(s_1, a, 2) = (1, s_1) \lor (2, s_1) \quad \rho(s_1, b, 2) = \top \]
Example of ATA

\[ A = (\{a, b\}, \{2\}, \{s_0, s_1\}, s_0, \rho, \{s_0\}) \text{ with} \]
\[ \rho(s_0, a, 2) = ((1, s_1) \lor (2, s_1)) \land (1, s_0) \land (2, s_0) \]
\[ \rho(s_0, b, 2) = (1, s_0) \land (2, s_0) \]
\[ \rho(s_1, a, 2) = (1, s_1) \lor (2, s_1) \]
\[ \rho(s_1, b, 2) = \top \]
Alternating Trees Automata for CTL (Vardi, 1995)

Let $\varphi$ be a CTL formula in positive normal form, and $D \subset \mathbb{N}$.

$A_{D,\varphi} = (2^{\text{AP}}, D, \text{SubF}(\varphi), \varphi, \rho, F)$:

- $\rho(P, a, k) = \text{tt}$ if $P \in a$, $\rho(P, a, k) = \text{ff}$ if $P \notin a$, 
- $\rho(\varphi_1 \land \varphi_2, a, k) = \rho(\varphi_1, a, k) \land \rho(\varphi_2, a, k)$
- $\rho(\mathsf{E X} \varphi_1, a, k) = \bigvee_{c=1,\ldots,k}(c, \varphi_1)$
- $\rho(\mathsf{A X} \varphi_1, a, k) = \bigwedge_{c=1,\ldots,k}(c, \varphi_1)$
- $\rho(\mathsf{E} \varphi_1 \mathsf{ U} \varphi_2, a, k) = \rho(\varphi_2, a, k) \lor \left( \rho(\varphi_1, a, k) \land \bigvee_{c=1,\ldots,k}(c, \mathsf{E} \varphi_1 \mathsf{ U} \varphi_2) \right)$
- $\mathsf{A} \varphi_1 \mathsf{ U} \varphi_2, \mathsf{E} \varphi_1 \mathsf{ W} \varphi_2, \mathsf{A} \varphi_1 \mathsf{ W} \varphi_2 \ldots$

And $F$ is the set of $\mathsf{W}$-formulae in $\varphi$.

Theorem (Kupferman, Vardi & Wolper, 2000)

$T(A_{D,\varphi})$ is the set of $D$-trees satisf. $\varphi$. 
Alternating Trees Automata for CTL (Vardi, 1995)

Let $\varphi$ be a CTL formula in positive normal form, and $D \subseteq \mathbb{N}$.

$$A_{D, \varphi} = (2^{2^{|\mathcal{P}|}}, D, \text{SubF}(\varphi), \varphi, \rho, F):$$

- $\rho(P, a, k) = \text{tt}$ if $P \in a$, $\rho(P, a, k) = \text{ff}$ if $P \notin a$, ...  
- $\rho(\varphi_1 \land \varphi_2, a, k) = \rho(\varphi_1, a, k) \land \rho(\varphi_2, a, k)$
- $\rho(\exists X \varphi_1, a, k) = \bigvee_{c=1, \ldots, k}(c, \varphi_1)$
- $\rho(\forall X \varphi_1, a, k) = \bigwedge_{c=1, \ldots, k}(c, \varphi_1)$
- $\rho(\exists \varphi_1 \cup \varphi_2, a, k) = \rho(\varphi_2, a, k) \lor \left(\rho(\varphi_1, a, k) \land \bigvee_{c=1, \ldots, k}(c, \exists \varphi_1 \cup \varphi_2)\right)$
- $A \varphi_1 \cup \varphi_2, E \varphi_1 \setminus \varphi_2, A \varphi_1 \setminus \varphi_2$ ...

And $F$ is the set of $W$-formulae in $\varphi$.

Theorem (Kupferman, Vardi & Wolper, 2000)

$T(A_{D, \varphi})$ is the set of $D$-trees satisf. $\varphi$. 
Example

$$\varphi = \mathbf{A F AG P}$$
$$\varphi \equiv \mathbf{A F (AP W ff)}$$

$$\mathcal{A}_D,\varphi = (\{\{P\}, \emptyset\}, \mathcal{D}, \{\varphi, \mathbf{AG P}, P\}, \varphi, \rho, \{\mathbf{AG P}\})$$

<table>
<thead>
<tr>
<th>$S$</th>
<th>$\rho(s, {P}, k)$</th>
<th>$\rho(s, \emptyset, k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>$\mathsf{tt}$</td>
<td>$\mathsf{ff}$</td>
</tr>
<tr>
<td>$\mathbf{AG P}$</td>
<td>$\bigwedge_{c=1 \ldots k} (c, \mathbf{AG P})$</td>
<td>$\mathsf{ff}$</td>
</tr>
<tr>
<td>$\varphi$</td>
<td>$\bigwedge_{c=1 \ldots k} (c, \mathbf{AG P}) \lor \bigwedge_{i=1 \ldots k} (c, \varphi)$</td>
<td>$\bigwedge_{i=1 \ldots k} (c, \varphi)$</td>
</tr>
</tbody>
</table>
Decision procedures for \(CTL\)

**Theorem (Emerson & Sistla, 1984)**

A \(CTL\) formula \(\varphi\) is satisfiable iff it is satisfied in an \(\{n\}\)-tree where \(n\) is the number of \(E\) in \(\varphi\).

Satisfiability checking \(\rightarrow\) non-emptiness checking of \(A_{\{n\},\varphi}\). (in exponential time)

Model checking: \(S \models \varphi\)?

- construct \(A_{D_S,\varphi}\)
  (weak alternating tree automaton)
- construct \(A_{S,\varphi} = A_{D_S,\varphi} \times S\)
  (one-letter weak alternating word automaton)
- emptiness checking of \(A_{S,\varphi}\) (linear time !)

These algorithms are optimal.

Other constructions are possible for \(CTL^*\) and the \(\mu\)-calculus.
Decision procedures for $CTL$

**Theorem (Emerson & Sistla, 1984)**

A $CTL$ formula $\varphi$ is satisfiable iff it is satisfied in an $\{n\}$-tree where $n$ is the number of $E$ in $\varphi$.

Satisfiability checking $\rightarrow$ non-emptiness checking of $A_{\{n\},\varphi}$.
(in exponential time)

Model checking: $S \models \varphi$?

- construct $A_{D_S,\varphi}$
  (weak alternating tree automaton)
- construct $A_{S,\varphi} = A_{D_S,\varphi} \times S$
  (one-letter weak alternating word automaton)
- emptiness checking of $A_{S,\varphi}$ (linear time !)

These algorithms are optimal.

Other constructions are possible for $CTL^*$ and the $\mu$-calculus.
Outline

12 BT-temporal logics with Past

13 $CTL^*$ vs Monadic second order logic

14 Automata theory and BT-temporal logics

15 Alternating-time temporal logic
Other results

There are many branching-time temporal logics.

ATL (Alternating-time Temporal Logic) extends CTL by considering strategies of agents.

Instead of quantifying over paths, we can quantify over the ability of some agents to ensure a given property.

(...whatever the choices of the other players.)

The same techniques can be applied.

The results may be quite different: it is important to consider carefully expressivity of TL.
Other results

There are many branching-time temporal logics...

ATL (Alternating-time Temporal Logic) extends CTL by considering strategies of agents.

Instead of quantifying over paths, we can quantify over the ability of some agents to ensure a given property. (. . . whatever the choices of the other players.)

The same techniques can be applied.

The results may be quite different: it is important to consider carefully expressivity of TL.
Other results

There are many branching-time temporal logics. . .

ATL (Alternating-time Temporal Logic) extends $CTL$ by considering strategies of agents.

Instead of quantifying over paths, we can quantify over the ability of some agents to ensure a given property. ( . . . whatever the choices of the other players.)

The same techniques can be applied.

The results may be quite different: it is important to consider carefully expressivity of TL.
A CGS $\mathcal{C}$ is a 6-tuple $(Q, AP, I, Agt, Mov, \rightarrow)$ s.t:

- $Q$: a finite set of *locations*;
- $AP$: *atomic propositions*;
- $I$: $Q \rightarrow 2^{AP}$: a labeling function;
- $Agt = \{A_1, ..., A_k\}$: a set of *agents* (or *players*);
- $Mov$: $Q \times Agt \rightarrow \mathbb{N}_{\geq 1}$ the choice function. $Mov(\ell, A_i) =$ number of possible moves for $A_i$ from $\ell$.
- $\rightarrow$: $Q \times \mathbb{N}^k \rightarrow Q$: the transition table.
CGS Example

Example

\[ \langle p|p \rangle, \langle r|r \rangle, \langle s|s \rangle \]

\[ \text{Paper, rock and scissors} \]
From a location $\ell$, each $A_i$ chooses some $m_{A_i}$ with $m_{A_i} < \text{Mov}(\ell, A_i)$.

$\rightarrow(\ell, m_{A_1}, \cdots, m_{A_k})$ gives the new location.

Notations:

- $\text{Next}(\ell) = \{ \rightarrow(\ell, \cdots m_{A_i} \cdots) \mid \forall m_{A_i} \cdot 1 \leq i \leq k \}$
- $\text{Next}(\ell, A_j, m) = \{ \rightarrow(\ell, \cdots, m_{A_{j-1}}, m, m_{A_{j+1}} \cdots) \}$
CGS example

Example

Paper, rock and scissors
CGS example

Example

Paper, rock and scissors
CGS example

Example

\[\langle p|p\rangle, \langle r|r\rangle, \langle s|s\rangle\]

\[\text{Start } q_0\]

\[q_0 \xrightarrow{\langle p|p\rangle} q_0, q_2 \xrightarrow{\langle s|r\rangle} q_1, q_1 \xrightarrow{\langle r|s\rangle} q_0\]

\[q_2 \xrightarrow{2\text{-Win}} q_2, q_1 \xrightarrow{1\text{-Win}} q_1, q_0 \xrightarrow{1\text{-Win}} q_0\]

<table>
<thead>
<tr>
<th>State</th>
<th>p</th>
<th>q0</th>
<th>q1</th>
<th>q2</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td></td>
<td>p</td>
<td>r</td>
<td>s</td>
</tr>
<tr>
<td>p</td>
<td>q0</td>
<td>q1</td>
<td>q2</td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>q2</td>
<td>q0</td>
<td>q1</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>q1</td>
<td>q2</td>
<td>q0</td>
<td></td>
</tr>
</tbody>
</table>

Paper, rock and scissors
Strategy and outcomes

**Definition**

- A **computation** is an infinite sequence $\rho = \ell_0 \ell_1 \cdots$ such that $\forall i, \ell_{i+1} \in \text{Next}(\ell_i)$.

- A **strategy** is a function $f_{A_i}$ s.t.
  \[
  f_{A_i}(\ell_0, \cdots, \ell_m) = \text{a possible move for } A_i \text{ from } \ell_m.
  \]

- The **outcomes** $\text{Out}(\ell, f_{A_i})$ are the set of computations from $\ell$ induced by the strategy $f_{A_i}$ for $A_i$.

- Given $A \subseteq \text{Agt}$ we note:
  - $F_A = \{ f_{A_i} | A_i \in A \}$
  - $\text{Out}(\ell, F_A)$
Syntax of ATL

Definition (Alur, Henzinger & Kupferman, 1997)

The syntax of $\textit{ATL}$ is defined by the following grammar:

$$\textit{ATL} \ni \varphi_s, \psi_s ::= P \mid \neg \varphi_s \mid \varphi_s \lor \psi_s \mid \langle \langle A \rangle \rangle \varphi_p$$

$$\varphi_p ::= X \varphi_s \mid G \varphi_s \mid \varphi_s U \psi_s.$$ 

with $P \in \text{AP}$ and $A \subseteq \text{Agt}$. 

- $E = \langle \langle \text{Agt} \rangle \rangle$
- $A = \langle \langle \emptyset \rangle \rangle$
### Definition

<table>
<thead>
<tr>
<th>Formula</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell \models \langle A \rangle \varphi_p$</td>
<td>if $\exists F_A \in \text{Strat}(A)$. $\forall \rho \in \text{Out}(\ell, F_A)$. $\rho \models \varphi_p$</td>
</tr>
<tr>
<td>$\rho \models \varphi_s \mathbf{U} \psi_s$</td>
<td>if $\exists i. \rho[i] \models \psi_s$ and $\forall 0 \leq j &lt; i. \rho[j] \models \varphi_s$</td>
</tr>
</tbody>
</table>

- Abbreviation: $\square[A]\varphi$ for $\neg \langle A \rangle \neg \varphi$
- $\neg \langle A \rangle \varphi \not\iff \langle \text{Agt} \setminus A \rangle \neg \varphi$
Until vs. Weak Until

**Definition**

- \( \varphi \ W \psi \equiv \varphi \ U \psi \lor G \varphi \)
- \( \neg (\varphi \ U \psi) \equiv (\neg \psi) \ W (\neg \varphi \land \neg \psi) \)

**Theorem**

\( \mathcal{E} \varphi \ W \psi \equiv \mathcal{E} G \varphi \lor \mathcal{E} \varphi \ U \psi \)

**Theorem (Laroussinie, Markey & Oreiby, 2006)**

\( \langle A \rangle (a \ W \ b) \) cannot be expressed in ATL.

\( \langle A \rangle (a \ W \ b) \iff \langle A \rangle G a \lor \langle A \rangle a \ U \ b \)
Until vs. Weak Until

**Definition**

- $\varphi W \psi \equiv \varphi U \psi \lor G \varphi$
- $\neg (\varphi U \psi) \equiv (\neg \psi) W (\neg \varphi \land \neg \psi)$

**Theorem**

$E \varphi W \psi \equiv E G \varphi \lor E \varphi U \psi$

**Theorem (Laroussinie, Markey & Oreiby, 2006)**

$\langle A \rangle (a W b)$ cannot be expressed in ATL.

$\langle A \rangle (a W b) \iff \langle A \rangle G a \lor \langle A \rangle a U b$
Until vs. Weak Until

**Definition**
- $\phi \mathbf{W} \psi \equiv \phi \mathbf{U} \psi \lor \mathbf{G} \phi$
- $\neg (\phi \mathbf{U} \psi) \equiv (\neg \psi) \mathbf{W} (\neg \phi \land \neg \psi)$

**Theorem**
$\mathbf{E} \phi \mathbf{W} \psi \equiv \mathbf{E} \mathbf{G} \phi \lor \mathbf{E} \phi \mathbf{U} \psi$

**Theorem (Laroussinie, Markey & Oreiby, 2006)**
$\langle A \rangle (a \mathbf{W} b)$ cannot be expressed in ATL.

$\langle A \rangle (a \mathbf{W} b) \iff \langle A \rangle \mathbf{G} a \lor \langle A \rangle a \mathbf{U} b$
Proof

\[ s'_i \models \langle A \rangle a \text{ W } b \quad \text{but} \quad s_i \not\models \langle A \rangle a \text{ W } b \]

Lemma

\[ \forall i > 0, \forall \psi \in ATL \text{ with } |\psi| \leq i \text{ we have: } s_i \models \psi \text{ iff } s'_i \models \psi. \]
Expressiveness of Temporal Logics

François Laroussinie and Nicolas Markey

Lab. Specification et Verification
ENS Cachan & CNRS, France

August 4, 2006
Outline

16 Timed temporal logics
   - Definitions
   - Expressiveness and complexity

17 TPTL vs MTL

18 Timed logics and timed automata
Outline of today’s lecture

16 Timed temporal logics
   • Definitions
   • Expressiveness and complexity

17 TPTL vs MTL

18 Timed logics and timed automata
Outline

16 Timed temporal logics
   • Definitions
     • Expressiveness and complexity

17 TPTL vs MTL

18 Timed logics and timed automata
Outline of today’s lecture

16 Timed temporal logics
   • Definitions
      • Expressiveness and complexity

17 TPTL vs MTL

18 Timed logics and timed automata
Timed temporal logics

Temporal logics = qualitative requirements

Timed temporal logics adds quantitative requirements.
Timed temporal logics

Temporal logics = qualitative requirements

Timed temporal logics adds quantitative requirements.

Example

Any request is granted in at most 1 minute.

An alarm rings if the doors are open for more than 30 seconds.
Timed temporal logics

Temporal logics = qualitative requirements

Timed temporal logics adds quantitative requirements.

Example

Any request is granted in at most 1 minute.

An alarm rings if the doors are open for more than 30 seconds.

Requires explicit timing constraints in the model.
Adding “time” in Kripke structures

- basic idea: counting the number of transitions:
Adding “time” in Kripke structures

- basic idea: counting the number of transitions:

Examples

1. first floor
   - open_1
   - go 3rd floor

2. first floor
   - closed_1
   - go 3rd floor

3. second floor
   - go 3rd floor

4. third floor
   - closed_3
   - go 3rd floor

5. third floor
   - open_3
Adding “time” in Kripke structures

- basic idea: counting the number of transitions:

Examples

\[ G(\text{go 3rd floor} \Rightarrow F_{\leq 4} \text{open}_3) \]
Adding “time” in Kripke structures

- basic idea: counting the number of transitions:

Examples

\[ AG(\text{EF}_{\leq 10} \text{open}_1) \]
Adding “time” in Kripke structures

- basic idea: counting the number of transitions:
- slightly more involved: adding timing informations in Kripke structures:
Adding “time” in Kripke structures

- basic idea: counting the number of transitions:

- slightly more involved: adding timing informations in Kripke structures:

Examples

```
first floor
  open_1
  go 3rd floor
```

```
second floor
  closed_1
  go 3rd floor
```

```
third floor
  closed_3
  go 3rd floor
```

2

5

5

2
Adding “time” in Kripke structures

- basic idea: counting the number of transitions:

- slightly more involved: adding timing informations in Kripke structures:

Examples

$$G(\text{go 3rd floor} \Rightarrow F_{\leq 14} \text{open}_3)$$
Adding “time” in Kripke structures

- basic idea: counting the number of transitions:
- slightly more involved: adding timing informations in Kripke structures:

Examples

\[ A \neg G(E F_{\leq 25} \neg o_{\text{en}_1}) \]
Adding “time” in Kripke structures

- basic idea: counting the number of transitions:
- slightly more involved: adding timing informations in Kripke structures:

→ those models are not very expressive (only more succinct);
Adding “time” in Kripke structures

- basic idea: counting the number of transitions:

- slightly more involved: adding timing informations in Kripke structures:

~~ those models are not very expressive (only more succinct);
~~ in this settings, the logics also are not more expressive:

$$\forall G(\exists F_{\leq 25} \text{open}_1) \equiv \forall G(\exists X(\text{open}_1 \lor \exists X(\text{open}_1 \lor \exists X(\text{open}_1 \lor \exists X(\text{open}_1 \ldots))))))$$
Adding “time” in Kripke structures

- basic idea: counting the number of transitions:
- slightly more involved: adding timing informations in Kripke structures:

\[ \neg \neg \text{those models are not very expressive (only more succinct);} \]
\[ \neg \neg \text{in this settings, the logics also are not more expressive:} \]

\[
AG(EF_{\leq 25} \text{open}_1) \equiv AG(EX(open_1 \lor EX(open_1 \lor EX(open_1 \lor EX(open_1 \ldots))))
\]

\[ \neg \neg \text{we need a } \text{real-time} \text{ extension of those models.} \]
Timed automata

Definition

A *timed automaton* is a tuple $\mathcal{A} = \langle Q, Q_0, C, \rightarrow, \Sigma, \ell \rangle$ s.t.:

- $Q$ is the set of locations, of which $Q_0$ are initial;
- $C$ is a (finite) set of *clock variables*;
- $\rightarrow$ is the set of transitions
- $\Sigma$ is the alphabet;
- $\ell$ labels either the states or the transitions.
Timed automata

Definition

A *timed automaton* is a tuple $\mathcal{A} = \langle Q, Q_0, C, \rightarrow, \Sigma, \ell \rangle$ s.t.:

- $Q$ is the set of locations, of which $Q_0$ are initial;
- $C$ is a (finite) set of *clock variables*;
- $\rightarrow$ is the set of transitions
- $\Sigma$ is the alphabet;
- $\ell$ labels either the states or the transitions.

Clocks are used on transitions: a transition is labeled with a *guard*, i.e., a list of constraints $x \sim n$ where $x \in C$, $n \in \mathbb{Z}^+$ and $\sim \in \{<, \leq, =, \geq, >\}$. 
Timed automata

Example

- \(q_0\) (initial state)
  - \(z \leq 1, y := 0\) transitions to \(q_1\) (green)
  - \(y = 1, y := 0\) transitions to \(q_3\) (green)

- \(q_1\)
  - \(y < 1, x := 0\) transitions to \(q_2\) (red)
  - \(y = 1, z := 0\) transitions to \(q_3\) (blue)

- \(q_2\)
  - \(y = 1, z := 0\) transitions to \(q_3\) (blue)

- \(q_3\) (final state)
Timed automata

Example

\begin{center}
\begin{tikzpicture}[node distance=2cm,thick]
  \node[state, initial] (q0) {$q_0$};
  \node[state, right of=q0] (q1) {$q_1$};
  \node[state, right of=q1] (q2) {$q_2$};
  \node[state, below of=q1] (q3) {$q_3$};

  \path[->, green] (q0) edge node {$y:=0$} (q1);
  \path[->, red] (q2) edge [loop above] node {$y<1, x:=0$} (q2);
  \path[->, blue] (q2) edge node {$y=1, z:=0$} (q3);
  \path[->, blue] (q0) edge node {$z\leq 1$} (q1);
  \path[->, green] (q1) edge node {$y=1, y:=0$} (q3);
  \path[->, green] (q3) edge node {$y=1, z:=0$} (q2);

\end{tikzpicture}
\end{center}

\begin{itemize}
  \item $x =$
  \item $y =$
  \item $z =$
\end{itemize}
Timed automata

Example

- From $q_0$ to $q_1$: $z \leq 1, y := 0$
  - Transition color: green
- From $q_1$ to $q_2$: $y = 1, y := 0$
  - Transition color: blue
- From $q_1$ to $q_3$: $y = 1, y := 0$
  - Transition color: green
- From $q_2$: $y < 1, x := 0$
  - Transition color: red
- From $q_3$: $y = 1, z := 0$
  - Transition color: blue

Initial state: $q_0$

Transition times:
- $x = 0$
- $y = 0$
- $z = 0$
Timed automata

Example

\begin{align*}
  &x = 0.6 \\
  &y = 0.6 \\
  &z = 0.6
\end{align*}
Timed automata

Example

\[
\begin{align*}
q_0 & \xrightarrow{z \leq 1, y := 0} q_1 & \xrightarrow{y < 1, x := 0} q_2 \\
q_1 & \xrightarrow{y = 1, y := 0} q_3 & \xrightarrow{y = 1, z := 0} q_2 \\
q_3 & \xrightarrow{\text{green}} q_1 & \xrightarrow{\text{blue}} q_2
\end{align*}
\]
Timed automata

Example

```
x = 0 0.6 0.8
y = 0 0 0.2
z = 0 0.6 0.8
```

\( q_0 \) 往 \( q_1 \)
\( z \leq 1 \), \( y := 0 \)
\( \text{green} \)

\( q_1 \) 往 \( q_2 \)
\( y = 1, y := 0 \)
\( \text{blue} \)

\( q_1 \) 往 \( q_3 \)
\( y = 1, z := 0 \)
\( \text{green} \)

\( q_2 \) 往 \( q_1 \)
\( y < 1, x := 0 \)
\( \text{red} \)

\( q_3 \) 往 \( q_2 \)
\( y = 1, z := 0 \)
\( \text{blue} \)
Timed automata

**Example**

Timed automata with the following states and transitions:

- **q₀**: Transition on input `z ≤ 1` and `y := 0` to `q₁` labeled with "green".
- **q₁**: Transition on input `y = 1` and `y := 0` to `q₂` labeled with "blue".
- **q₂**: Transition on input `y < 1` and `x := 0` to `q₀` labeled with "red".
- **q₀**: Transition on input `y = 1` and `y := 0` to `q₃` labeled with "green".
- **q₃**: Transition on input `y = 1` and `z := 0` to `q₂` labeled with "blue".

The diagram also shows the timeline with intervals labeled (0, 0.6), (0.6, 0.8), and (0.8, 1.0) for `x`, `y`, and `z` respectively.
Timed automata

Example

\[
\begin{align*}
q_0 &\xrightarrow{z \leq 1, y := 0} q_1 \\
q_1 &\xrightarrow{y = 1, y := 0} q_2 \\
q_3 &\xrightarrow{y = 1, z := 0} q_2
\end{align*}
\]

\[
\begin{array}{c}
x = 0 & 0.6 & 0.8 & 1.1 \\
y = 0 & 0 & 0.2 & 0.5 \\
z = 0 & 0.6 & 0.8 & 1.1
\end{array}
\]
Timed automata

Example

x = 0 0.6 0.8 0
y = 0 0 0.2 0.5
z = 0 0.6 0.8 1.1
Timed automata

Example

\begin{itemize}
\item \( q_0 \)\( \xrightarrow{z \leq 1, y := 0} \) \( q_1 \)
\item \( q_1 \)\( \xrightarrow{y = 1, y := 0} \) \( q_3 \)
\item \( q_2 \)\( \xrightarrow{y \leq 1, x := 0} \) \( q_0 \)
\end{itemize}

\begin{align*}
&x = \begin{cases} 0 & 0.6 & 0.8 & 0 & 0.5 \\ y = \begin{cases} 0 & 0 & 0.2 & 0.5 & 1 \\ z = \begin{cases} 0 & 0.6 & 0.8 & 1.1 & 1.6 \\
\end{cases}
\end{cases}
\end{align*}

\text{(g,0.6)(b,0.8)} \quad \text{(r,1.1)}
Timed automata

Example

- \( q_0 \): \( z \leq 1, y := 0 \) (green)
  - \( x = y = z = 0 \)
  - \( (g,0.6)(b,0.8) \)
- \( q_1 \): \( y = 1, y := 0 \) (blue)
  - \( x = y = z = 0 \)
  - \( (r,1.1) \)
- \( q_2 \): \( y < 1, x := 0 \) (red)
  - \( x = y = z = 0 \)
  - \( (b,1.6) \)
- \( q_3 \): \( y = 1, z := 0 \) (blue)
  - \( x = y = z = 0 \)

Transition Table:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0.6</th>
<th>0.8</th>
<th>0</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y=</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>z=</td>
<td>0</td>
<td>0.6</td>
<td>0.8</td>
<td>1.1</td>
<td>0</td>
</tr>
</tbody>
</table>
Timed automata

Example

The diagram represents a timed automaton with states $q_0, q_1, q_2,$ and $q_3$. The transitions and conditions are as follows:

- From $q_0$ to $q_1$: $z \leq 1$ and $y := 0$.
- From $q_0$ to $q_3$: $y = 1$ and $y := 0$.
- From $q_1$ to $q_2$: $y < 1$ and $x := 0$.
- From $q_1$ to $q_3$: $y = 1$ and $z := 0$.
- From $q_2$ to $q_3$: $y = 1$ and $z := 0$.

The transition labels include time delays and actions:

- $0 \rightarrow 1$: $(g,0.6)$ and $(b,0.8)$.
- $1 \rightarrow 2$: $(r,1.1)$.
- $(b,1.6)$ and $(g,1.6)$.

The table below shows the values of $x$, $y$, and $z$ at different times:

<table>
<thead>
<tr>
<th>Time</th>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
</tr>
<tr>
<td>0.6</td>
<td>0.6</td>
<td>0.2</td>
<td>0.8</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Timed automata

Definition

A *timed word* is a function \( w : \mathbb{Z}^+ \rightarrow 2^{AP} \times \mathbb{R}^+ \) s.t. \( w_2 \) is nondecreasing and diverges.
Timed automata

Definition
A **timed word** is a function $w : \mathbb{Z}^+ \to 2^\text{AP} \times \mathbb{R}^+$ s.t. $w_2$ is nondecreasing and diverges.

Example

$$w = (\text{green, 0.6})(\text{blue, 0.8})(\text{red, 1.1})$$

$$\quad (\text{blue, 1.6})(\text{green, 1.6})...$$
Timed automata

Definition

A *timed word* is a function \( w : \mathbb{Z}^+ \to 2^{AP} \times \mathbb{R}^+ \) s.t. \( w_2 \) is nondecreasing and diverges.

Example

\[
w = (\varepsilon, 0)(\text{green}, 0.6)(\text{blue}, 0.8)(\text{red}, 1.1)
\]

\[
(\text{blue}, 1.6)(\text{green}, 1.6) \ldots
\]
Timed automata

Example

\[
\begin{align*}
q_0 & \xrightarrow{z \leq 1, y := 0} q_1 \\
q_0 & \xrightarrow{y = 1, y := 0} q_3 \\
q_3 & \xrightarrow{y = 1, z := 0} q_2 \\
q_2 & \xrightarrow{y < 1, x := 0} q_2
\end{align*}
\]
Timed automata

Example

- Transition: $z \leq 1$, $y := 0$
- Transition: $y = 1$, $y := 0$
- Transition: $y = 1$, $z := 0$
- Transition: $y < 1$, $x := 0$

States:
- $q_0$
- $q_1$
- $q_2$
- $q_3$

Initial state: $q_0$

States:
- $x =$
- $y =$
- $z =$
Timed automata

Example

\[\begin{align*}
q_0 & \xrightarrow{z \leq 1, y := 0} q_1 \\
q_0 & \xrightarrow{y = 1, y := 0} q_3 \\
q_1 & \xrightarrow{y < 1, x := 0} q_2 \\
q_3 & \xrightarrow{y = 1, z := 0} q_2 \\
\end{align*}\]
Timed automata

Example

\begin{align*}
q_0 &\xrightarrow{z \leq 1, y := 0} q_1 \\
q_1 &\xrightarrow{y = 1, y := 0} q_2 \\
q_2 &\xrightarrow{y < 1, x := 0} q_2 \\
q_2 &\xrightarrow{y = 1, z := 0} q_3 \\
q_3 &\xrightarrow{y = 1, y := 0} q_2 \\
\end{align*}

\begin{tabular}{lll}
\text{Time} & x & y & z \\
0 & 0 & 0 & 0.6 \\
1 & 0 & 0 & 0.6 \\
2 & 0 & 0 & 0.6 \\
\end{tabular}
Timed automata

Example

\[
\begin{align*}
q_0 & \xrightarrow{z \leq 1, y := 0} q_1 \\
q_1 & \xrightarrow{y = 1, y := 0} q_2 \\
q_2 & \xrightarrow{y < 1, x := 0} q_3 \\
q_3 & \xrightarrow{y = 1, z := 0} q_2
\end{align*}
\]

\begin{align*}
x &= 0 & 0.6 \\
y &= 0 & 0 \\
z &= 0 & 0.6
\end{align*}
Timed automata

Example

\begin{align*}
q_0 & \xrightarrow{z \leq 1, y := 0} q_1 \\
q_1 & \xrightarrow{y = 1, y := 0} q_2 \\
q_2 & \xrightarrow{y < 1, x := 0} q_1 \\
q_3 & \xrightarrow{y = 1, z := 0} q_2 \\
\end{align*}

\begin{tabular}{lcccc}
\text{time} & 0 & 0.6 & 0.8 & 2 \\
\text{x=} & 0 & 0.6 & 0.8 & 2 \\
\text{y=} & 0 & 0 & 0.2 & 2 \\
\text{z=} & 0 & 0.6 & 0.8 & 2 \\
\end{tabular}
Timed automata

Example

The figure shows a timed automaton with states $q_0$, $q_1$, $q_2$, and $q_3$. The transitions and conditions are as follows:

- $q_0$ to $q_1$: $z \leq 1$ and $y := 0$
- $q_1$ to $q_2$: $y = 1$, $y := 0$
- $q_2$: $y < 1$, $x := 0$
- $q_2$ to $q_3$: $y = 1$, $z := 0$
- $q_3$: $y = 1$, $z := 0$

The table below shows the timing information:

<table>
<thead>
<tr>
<th>State</th>
<th>0</th>
<th>0.6</th>
<th>0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y$</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>$z$</td>
<td>0</td>
<td>0.6</td>
<td>0.8</td>
</tr>
</tbody>
</table>
Timed automata

Example

- From $q_0$, transition on $z \leq 1$ with $y := 0$ to $q_1$.
- From $q_1$, transition on $y = 1, y := 0$ to $q_2$.
- From $q_2$, transition on $y < 1, x := 0$ back to $q_1$.
- From $q_1$, transition on $y = 1, z := 0$ to $q_3$.
- From $q_3$, transition on $x = y = z = 0$ to $q_0$.

Transition times:

- $x$: 0, 0.6, 0.8, 1.1
- $y$: 0, 0, 0.2, 0.5
- $z$: 0, 0.6, 0.8, 1.1
Timed automata

Example

\[
\begin{align*}
q_0 &\xrightarrow{z \leq 1, y := 0} q_1 \\
q_1 &\xrightarrow{y = 1, y := 0} q_2 \\
q_2 &\xrightarrow{y < 1, x := 0} q_0 \\
q_3 &\xrightarrow{y = 1, z := 0} q_1
\end{align*}
\]
Timed automata

Example

- **q0**: $y := 0$
- **q1**: $z \leq 1$
- **q2**: $y < 1, x := 0$
- **q3**: $y = 1, y := 0$

<table>
<thead>
<tr>
<th>$x$</th>
<th>0</th>
<th>0.6</th>
<th>0.8</th>
<th>0</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>$z$</td>
<td>0</td>
<td>0.6</td>
<td>0.8</td>
<td>1.1</td>
<td>1.6</td>
</tr>
</tbody>
</table>
Timed automata

Example

```
x = 0  0.6  0.8  0  0.5
y = 0  0    0.2  0.5  1
z = 0  0.6  0.8  1.1  0
```

\(q_0\) to \(q_1\) with \(z \leq 1, y := 0\)
\(q_1\) to \(q_2\) with \(y < 1, x := 0\)
\(q_2\) to \(q_3\) with \(y = 1, y := 0\)
\(q_3\) to \(q_2\) with \(y = 1, z := 0\)
Timed automata

Example

- Transition from $q_0$: $z \leq 1, y:=0$
- Transition from $q_1$: $y=1, y:=0$
- Transition from $q_2$: $y<1, x:=0$
- Transition from $q_3$: $y=1, z:=0$

Transition probabilities:

- $x$: 0, 0.6, 0.8, 0, 0.5, 0.5
- $y$: 0, 0, 0.2, 0.5, 1, 0
- $z$: 0, 0.6, 0.8, 1.1, 0, 0
Timed automata

Example

\[
\begin{align*}
q_0 & \xrightarrow{z \leq 1} q_1 \\
q_1 & \xrightarrow{y = 1, y := 0} q_2 \\
& \xrightarrow{y = 1, z := 0} q_3 \\
q_3 & \xrightarrow{y < 1, x := 0} q_2
\end{align*}
\]

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>0.6</th>
<th>0.8</th>
<th>0</th>
<th>0.5</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>z</td>
<td>0</td>
<td>0.6</td>
<td>0.8</td>
<td>1.1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Timed automata

Definition

A *timed state sequence* is a function $\pi : \mathbb{R}^+ \rightarrow 2^{AP}$. 
Timed automata

Definition
A *timed state sequence* is a function $\pi : \mathbb{R}^+ \rightarrow 2^{AP}$.

Example
Timed automata

Timed automata can also be used for defining languages: it suffices to add an acceptance (e.g., Büchi) condition.
Timed automata

Timed automata can also be used for defining languages: it suffices to add an acceptance (e.g., Büchi) condition.

Example

This automaton accepts timed words in which there exists an occurrence of \textit{green} that is not followed, one time unit later, by another occurrence of \textit{green}:

\[ F(\text{green} \land \neg F(\text{green})). \]
Timed automata

Timed automata can also be used for defining languages: it suffices to add an acceptance (e.g., Büchi) condition.

Example

This automaton accepts timed words in which there exists an occurrence of green that is not followed, one time unit later, by another occurrence of green:

$$F(\text{green} \land \neg F_{=1}\text{green}).$$
Timed automata

The negation of

\[ F(\text{green} \land \neg F_{=1}\text{green}) \]

is

\[ G(\text{green} \Rightarrow F_{=1}\text{green}) \]
The negation of

\[ F(\text{green} \land \neg F_{=1} \text{green}) \]

is

\[ G(\text{green} \Rightarrow F_{=1} \text{green}) \]

**Lemma**

No Büchi timed automaton accepts the language defined by

\[ G(\text{green} \Rightarrow F_{=1} \text{green}) \]
Timed automata

The negation of

\[ F(\text{green} \land \neg F_{=1} \text{green}) \]

is

\[ G(\text{green} \Rightarrow F_{=1} \text{green}) \]

Lemma

No Büchi timed automaton accepts the language defined by

\[ G(\text{green} \Rightarrow F_{=1} \text{green}) \]

Theorem

Timed automata are closed under conjunction and disjunction, but not under negation.
Extending temporal logics with time

Two different ways of extending temporal logics:

- **by associating intervals with modalities:** those intervals (having rational bounds) indicate e.g. the moment at which an eventuality is to be fulfilled.
Extending temporal logics with time

Two different ways of extending temporal logics:

- by associating intervals with modalities: those intervals (having rational bounds) indicate e.g. the moment at which an eventuality is to be fulfilled.

Examples

\[ \mathbf{G}(\text{call}_3 \Rightarrow \mathbf{F}_{[0,1]} \text{open}_3) \]
Extending temporal logics with time

Two different ways of extending temporal logics:

- **by associating intervals with modalities:** those intervals (having rational bounds) indicate e.g. the moment at which an eventuality is to be fulfilled.

Examples

\[ G(\text{call}_3 \Rightarrow F_{[0,1]} \text{ open}_3) \]

\[ A \ G(\text{E} F_{[0,3]} \text{ open}_1) \]
Extending temporal logics with time

Two different ways of extending temporal logics:

- **by associating intervals with modalities**: those intervals (having rational bounds) indicate e.g. the moment at which an eventuality is to be fulfilled.

- **by using real clocks in the formula**: the clocks can be reset at some point during the evaluation of the formula, and then compared to rationals.
Extending temporal logics with time

Two different ways of extending temporal logics:

- **by associating intervals with modalities**: those intervals (having rational bounds) indicate e.g. the moment at which an eventuality is to be fulfilled.

- **by using real clocks in the formula**: the clocks can be reset at some point during the evaluation of the formula, and then compared to rationals.

### Examples

\[ G(\text{call}_3 \Rightarrow x. F(\text{open}_3 \land x \leq 1)) \]
Extending temporal logics with time

Two different ways of extending temporal logics:

- by associating intervals with modalities: those intervals (having rational bounds) indicate e.g. the moment at which an eventuality is to be fulfilled.

- by using real clocks in the formula: the clocks can be reset at some point during the evaluation of the formula, and then compared to rationals.

Examples

\[ G(\textit{call}_3 \Rightarrow x. F(\textit{open}_3 \land x \leq 1)) \]

\[ A G(x.E F(\textit{open}_1 \land x \leq 3)) \]
Timed logics in the pointwise framework

- Syntax of MTL:

\[
\text{MTL} \ni \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \mathbf{U} I \varphi
\]

where \( p \) ranges over \( \text{AP} \) and \( I \) is an interval with bounds in \( \mathbb{Q}^+ \cup \{+\infty\} \).
Timed logics in the pointwise framework

- **Syntax of MTL:**

  \[
  \text{MTL } \ni \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \mathbf{U}_I \varphi
  \]

  where \( p \) ranges over \( \text{AP} \) and \( I \) is an interval with bounds in \( \mathbb{Q}^+ \cup \{+\infty\} \).

- **Pointwise semantics of MTL:** over \( \pi = ((w_i)_i,(t_i)_i) \):

  - \( \pi, i \models \varphi \mathbf{U}_I \psi \) iff there exists some \( j > 0 \) s.t.
    - \( \pi, i + j \models \psi \),
    - \( \pi, i + k \models \varphi \) for all \( 0 < k < j \),
    - \( t_{i+j} - t_i \in I \).
Timed logics in the pointwise framework

- **Syntax of MTL:**

  \[
  \text{MTL} \ni \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \mathbf{U}_I \varphi
  \]

  where \( p \) ranges over \( \text{AP} \) and \( I \) is an interval with bounds in \( \mathbb{Q}^+ \cup \{+\infty\} \).

- **Pointwise semantics of MTL:** over \( \pi = ((w_i)_i, (t_i)_i) \):

  - \( \pi, i \models \varphi \mathbf{U}_I \psi \) iff there exists some \( j > 0 \) s.t.
    - \( \pi, i + j \models \psi \),
    - \( \pi, i + k \models \varphi \) for all \( 0 < k < j \),
    - \( t_{i+j} - t_i \in I \).

- **Examples:**

  - \( 0 \ 1 \ 2 \)
  - \( (\text{red},0.3) \quad (\text{red},1.2) \quad (\text{blue},2.1) \)
  - \( \text{red} \mathbf{U}_{[2,3]} \text{ blue} \)
Timed logics in the pointwise framework

- **Syntax of MTL:**

\[
\text{MTL} \ni \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \quad \exists I \varphi
\]

where \( p \) ranges over \( AP \) and \( I \) is an interval with bounds in \( Q^+ \cup \{+\infty\} \).

- **Pointwise semantics of MTL:** over \( \pi = ((w_i)_i, (t_i)_i) \):
  - \( \pi, i \models \varphi \quad \exists I \varphi \psi \) iff there exists some \( j > 0 \) s.t.
    - \( \pi, i + j \models \psi \),
    - \( \pi, i + k \models \varphi \) for all \( 0 < k < j \),
    - \( t_{i+j} - t_i \in I \).

- **Examples:**

\[
\begin{array}{c}
0 & 1 & 2 \\
\text{(green,0.2)} & \text{(green,1.3)} & \text{(red,2.3)}
\end{array}
\]

\[ F(\text{green} \land \bot \quad \exists [1,1] \text{ red}) \]
Timed logics in the pointwise framework

- Syntax of MTL:

\[ \text{MTL} \ni \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \cup I \varphi \]

where \( p \) ranges over \( \text{AP} \) and \( I \) is an interval with bounds in \( \mathbb{Q}^+ \cup \{+\infty\} \).

- Pointwise semantics of MTL: over \( \pi = ((w_i)_i, (t_i)_i) \):

  - \( \pi, i \models \varphi \cup I \psi \) iff there exists some \( j > 0 \) s.t.
    - \( \pi, i + j \models \psi \),
    - \( \pi, i + k \models \varphi \) for all \( 0 < k < j \),
    - \( t_{i+j} - t_i \in I \).

- Examples:

  \[
  \begin{array}{c}
  0 \quad 1 \quad 2 \\
  \hline
  \text{(red,0.2)} \quad \text{(green,0.9)} \quad \text{(blue,2)} \\
  \end{array}
  \Rightarrow \text{F}_{[2,2]} \text{ blue}
  \]
Timed logics in the pointwise framework

- **Syntax of MTL:**

\[
\text{MTL} \ni \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \mathcal{U}_I \varphi
\]

where \( p \) ranges over \( AP \) and \( I \) is an interval with bounds in \( \mathbb{Q}^+ \cup \{+\infty\} \).

- **Pointwise semantics of MTL:** over \( \pi = ((w_i)_i, (t_i)_i) \):
  
  \[ \pi, i \models \varphi \mathcal{U}_I \psi \text{ iff there exists some } j > 0 \text{ s.t.} \]

  - \( \pi, i + j \models \psi \),
  - \( \pi, i + k \models \varphi \) for all \( 0 < k < j \),
  - \( t_{i+j} - t_i \in I \).

- **Examples:**

\[
\begin{array}{c}
\text{red,0.2} \quad \text{green,0.9} \quad \text{blue,2} \\
0 \quad 1 \quad 2
\end{array}
\]

\[ F_{[2,2]} \text{ blue} \overset{\text{def}}{=} F_{\leq 2} \text{ blue} \]
Timed logics in the pointwise framework

- Syntax of MTL:

\[
\text{MTL} \ni \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \mathbf{U}_I \varphi
\]

where \( p \) ranges over \( \text{AP} \) and \( I \) is an interval with bounds in \( \mathbb{Q}^+ \cup \{+\infty\} \).

- Pointwise semantics of MTL: over \( \pi = ((w_i)_i, (t_i)_i) \):

  - \( \pi, i \models \varphi \mathbf{U}_I \psi \) iff there exists some \( j > 0 \) s.t.
    - \( \pi, i + j \models \psi \),
    - \( \pi, i + k \models \varphi \) for all \( 0 < k < j \),
    - \( t_{i+j} - t_i \in I \).

- Examples:

\[
\text{F}_{[2,2]} \text{blue} \not\equiv \text{F}_{=1} \text{F}_{=1} \text{blue}
\]

\[
0 \quad 1 \quad 2
\]

\[
\text{(red,0.2)} \quad \text{(green,0.9)} \quad \text{(blue,2)}
\]
Timed logics in the pointwise framework

- **Syntax of MTL:**

  \[
  \text{MTL } \ni \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \mathbf{U} I \varphi
  \]

  where \( p \) ranges over \( AP \) and \( I \) is an interval with bounds in \( \mathbb{Q}^+ \cup \{+\infty\} \).

- **Pointwise semantics of MTL:** over \( \pi = ((w_i)_i, (t_i)_i) : \)

  \[ \pi, i \models \varphi \mathbf{U} I \psi \text{ iff there exists some } j > 0 \text{ s.t.} \]
  \[ \begin{align*}
  &\pi, i + j \models \psi, \\
  &\pi, i + k \models \varphi \text{ for all } 0 < k < j, \\
  &t_{i+j} - t_i \in I.
  \end{align*} \]

- **Examples:**

  \[
  \text{F(blue } \land \text{ G}_{[-1,0]}^{-1} \bot) \]

  \[ \begin{array}{ccc}
  0 & 1 & 2 \\
  \text{(red,0.2)} & \text{(green,0.9)} & \text{(blue,2.2)}
  \end{array} \]
Timed logics in the pointwise framework

• Syntax of TPTL:

\[ \text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid x. \varphi \]

where \( p \) ranges over \( \text{AP} \), \( x \) ranges over a set of formula clocks, \( c \in \mathbb{Q}^+ \) and \( \sim \in \{<, \leq, =, \geq, >\} \).
Timed logics in the pointwise framework

- Syntax of TPTL:

\[
\text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \mathbf{U} \varphi \mid x. \varphi
\]

where \( p \) ranges over \( \text{AP} \), \( x \) ranges over a set of formula clocks, \( c \in \mathbb{Q}^+ \) and \( \sim \in \{<, \leq, =, \geq, >\} \).

- Pointwise semantics of TPTL: over \( \pi = ((w_i)_i, (t_i)_i) \):

  \[\pi, i, \tau \models x \sim c \iff \tau(x) \sim c\]
Timed logics in the pointwise framework

- Syntax of TPTL:

  \[
  \text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \mathbf{U} \varphi \mid x. \varphi
  \]

  where \( p \) ranges over \( \text{AP} \), \( x \) ranges over a set of formula clocks, \( c \in \mathbb{Q}^+ \) and \( \sim \in \{<, \leq, =, \geq, >\} \).

- Pointwise semantics of TPTL: over \( \pi = ((w_i)_i, (t_i)_i) \):
  - \( \pi, i, \tau \models x \sim c \) iff \( \tau(x) \sim c \)
  - \( \pi, i, \tau \models x. \varphi \) iff \( \pi, i, \tau[x\leftarrow0] \models \varphi \)
Timed logics in the pointwise framework

- Syntax of TPTL:

\[
\text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \cup \varphi \mid x. \varphi
\]

where \( p \) ranges over \( \text{AP} \), \( x \) ranges over a set of formula clocks, \( c \in \mathbb{Q}^+ \) and \( \sim \in \{<, \leq, =, \geq, >\} \).

- Pointwise semantics of TPTL: over \( \pi = ((w_i)_i, (t_i)_i) \):

  - \( \pi, i, \tau \models x \sim c \) iff \( \tau(x) \sim c \)
  - \( \pi, i, \tau \models x. \varphi \) iff \( \pi, i, \tau[x \leftarrow 0] \models \varphi \)
  - \( \pi, i, \tau \models \varphi \cup \psi \) iff there exists some \( j > 0 \) s.t.
    - \( \pi, i + j, \tau + t_{i+j} - t_i \models \psi \),
    - \( \pi, i + k, \tau + t_{i+k} - t_i \models \varphi \) for all \( 0 < k < j \).
Timed logics in the pointwise framework

- Syntax of TPTL:

  \[ \text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \lor \varphi \mid x. \varphi \]

  where \( p \) ranges over \( \text{AP} \), \( x \) ranges over a set of formula clocks, \( c \in \mathbb{Q}^+ \) and \( \sim \in \{<, \leq, =, \geq, >\} \).

- Pointwise semantics of TPTL: over \( \pi = ((w_i)_i, (t_i)_i) \):

  - \( \pi, i, \tau \models x \sim c \) iff \( \tau(x) \sim c \)
  - \( \pi, i, \tau \models x. \varphi \) iff \( \pi, i, \tau[x\leftarrow 0] \models \varphi \)
  - \( \pi, i, \tau \models \varphi \lor \psi \) iff there exists some \( j > 0 \) s.t.
    - \( \pi, i + j, \tau + t_{i+j} - t_i \models \psi \),
    - \( \pi, i + k, \tau + t_{i+k} - t_i \models \varphi \) for all \( 0 < k < j \).

- Examples:

  0 1 2
  \[ (\text{red,0.3}) \quad (\text{red,1.2}) \quad (\text{blue,2.1}) \]

  \[ x.(\text{red} \lor (\text{blue} \land x \in [2, 3])) \]
Timed logics in the pointwise framework

- Syntax of TPTL:

\[ \text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \U \varphi \mid x. \varphi \]

where \( p \) ranges over \( \text{AP} \), \( x \) ranges over a set of formula clocks, \( c \in \mathbb{Q}^+ \) and \( \sim \in \{<,\leq,=,\geq,>\} \).

- Pointwise semantics of TPTL: over \( \pi = ((w_i)_i, (t_i)_i) \):

  - \( \pi, i, \tau \models x \sim c \) iff \( \tau(x) \sim c \)
  - \( \pi, i, \tau \models x. \varphi \) iff \( \pi, i, \tau[x \leftarrow 0] \models \varphi \)
  - \( \pi, i, \tau \models \varphi \U \psi \) iff there exists some \( j > 0 \) s.t.
    - \( \pi, i + j, \tau + t_{i+j} - t_i \models \psi \),
    - \( \pi, i + k, \tau + t_{i+k} - t_i \models \varphi \) for all \( 0 < k < j \).

- Examples:

\( F(green \land x.(\bot \U (red \land x = 1))) \)

0 1 2
(red,0.2) (green,1.1) (red,2.1)
Timed logics in the pointwise framework

- **Syntax of TPTL:**

  \[ \text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \cup \varphi \mid x. \varphi \]

  where \( p \) ranges over \( \text{AP} \), \( x \) ranges over a set of formula clocks, \( c \in \mathbb{Q}^+ \) and \( \sim \in \{<,\leq,=,\geq,>\} \).

- **Pointwise semantics of TPTL:** over \( \pi = ((w_i)_i,(t_i)_i) \):
  
  - \( \pi, i, \tau \models x \sim c \iff \tau(x) \sim c \)
  - \( \pi, i, \tau \models x. \varphi \iff \pi, i, \tau[x\leftarrow 0] \models \varphi \)
  - \( \pi, i, \tau \models \varphi \cup \psi \iff \) there exists some \( j > 0 \) s.t.
    - \( \pi, i + j, \tau + t_{i+j} - t_i \models \psi \),
    - \( \pi, i + k, \tau + t_{i+k} - t_i \models \varphi \) for all \( 0 < k < j \).

- **Examples:**

  \[
  \begin{array}{c|c|c}
  0 & 1 & 2 \\
  \hline
  (\text{red},0.2) & (\text{green},0.9) & (\text{blue},2) \\
  \hline
  \end{array}
  \]

  \( x. \text{F(red} \land \text{F(green} \land x \leq 1) \)
Timed logics in the continuous framework

- Syntax of MTL:

\[ \text{MTL } \exists \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \mathcal{U}_t \varphi \]
Timed logics in the continuous framework

- Syntax of MTL:

  \[ \text{MTL } \ni \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \mathbf{U}_I \varphi \]

- Continuous semantics of MTL: over \( \pi : \mathbb{R}^+ \rightarrow 2^{\text{AP}} \):
  \begin{itemize}
  \item \( \pi, t \models \varphi \mathbf{U}_I \psi \) iff there exists some \( u > 0 \) s.t.
    \begin{itemize}
    \item \( \pi, t + u \models \psi \),
    \item \( \pi, t + \nu \models \varphi \) for all \( 0 < \nu < u \),
    \item \( u \in \mathcal{I} \).
    \end{itemize}
  \end{itemize}
Timed logics in the continuous framework

- Syntax of MTL:

\[
\text{MTL} \ni \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \ U \ I \ \varphi
\]

- Continuous semantics of MTL: over \( \pi : \mathbb{R}^+ \to 2^{\text{AP}} \):

  - \( \pi, t \models \varphi \ U \ I \ \psi \) iff there exists some \( u > 0 \) s.t.
    
    \begin{itemize}
      \item \( \pi, t + u \models \psi \),
      \item \( \pi, t + v \models \varphi \) for all \( 0 < v < u \),
      \item \( u \in I \).
    \end{itemize}

  - \( \pi, t \models p \) iff \( p \in \pi(t) \)
Timed logics in the continuous framework

- Syntax of MTL:

  $$\text{MTL } \ni \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \mathrel{U} I \varphi$$

- Continuous semantics of MTL: over $\pi : \mathbb{R}^+ \to 2^{\text{AP}}$:
  - $\pi, t \models \varphi \mathrel{U} I \psi$ iff there exists some $u > 0$ s.t.
    - $\pi, t + u \models \psi$,
    - $\pi, t + v \models \varphi$ for all $0 < v < u$,
    - $u \in I$.
  - $\pi, t \models p$ iff $p \in \pi(t)$

- Examples:

  $0$ $1$ $2$

  $(\text{red} \lor \text{blue}) \mathrel{U}_{\leq 2} \text{green}$
Timed logics in the continuous framework

- **Syntax of MTL:**

  \[
  \text{MTL} \ni \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \mathbf{U}_t \psi
  \]

- **Continuous semantics of MTL:** over \( \pi : \mathbb{R}^+ \rightarrow 2^{\text{AP}} \):

  - \( \pi, t \models \varphi \mathbf{U}_t \psi \iff \) there exists some \( u > 0 \) s.t.
    - \( \pi, t + u \models \psi \),
    - \( \pi, t + \nu \models \varphi \) for all \( 0 < \nu < u \),
    - \( u \in l \).
  - \( \pi, t \models p \iff p \in \pi(t) \)

- **Examples:**

  \[
  \begin{array}{c|c|c|c|c}
  0 & 1 & 2 & F =_2 \text{green} \\
  \hline
  \text{red} & \text{red} & \text{green} &
  \end{array}
  \]
Timed logics in the continuous framework

- **Syntax of MTL:**

  \[ \text{MTL} \ni \varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \mathbf{U}_I \varphi \]

- **Continuous semantics of MTL:** over \( \pi : \mathbb{R}^+ \rightarrow 2^{\text{AP}} \):

  - \( \pi, t \models \varphi \mathbf{U}_I \psi \) iff there exists some \( u > 0 \) s.t.
    - \( \pi, t + u \models \psi \),
    - \( \pi, t + v \models \varphi \) for all \( 0 < v < u \),
    - \( u \in I \).
  - \( \pi, t \models p \) iff \( p \in \pi(t) \)

- **Examples:**

  \[
  \begin{array}{c|c|c}
  0 & 1 & 2 \\
  \text{red} & \text{green} & \text{red} \text{green} \\
  \end{array}
  \]

  \( \mathbf{F}_{=2} \text{green} \equiv \mathbf{F}_{=1}(\mathbf{F}_{=1} \text{green}) \)
Timed logics in the continuous framework

Syntax of TPTL:

\[ \text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg \varphi \mid \varphi \vee \varphi \mid \varphi \cup \varphi \mid x. \varphi \]
Timed logics in the continuous framework

- Syntax of TPTL:

\[
\text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \text{ U } \varphi \mid x. \varphi
\]

- Continuous semantics of TPTL: over \( \pi: \mathbb{R}^+ \rightarrow 2^{\text{AP}} \):
  - \( \pi, t, \tau \models x \sim c \) iff \( \tau(x) \sim c \)
Timed logics in the continuous framework

- **Syntax of TPTL:**

\[ \text{TPTL \models \varphi ::= p \mid x \sim c \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi \mid x. \varphi} \]

- **Continuous semantics of TPTL:** over \( \pi : \mathbb{R}^+ \rightarrow 2^{AP} \):
  
  \( \pi, t, \tau \models x \sim c \) iff \( \tau(x) \sim c \)

  \( \pi, t, \tau \models x. \varphi \) iff \( \pi, i, \tau[x \leftarrow 0] \models \varphi \)
Timed logics in the continuous framework

- **Syntax of TPTL:**

  \[
  \text{TPTL} \ni \phi ::= p \mid x \sim c \mid \neg \phi \mid \phi \lor \psi \mid \phi \cup \psi \mid x. \phi
  \]

- **Continuous semantics of TPTL:** over \( \pi: \mathbb{R}^+ \rightarrow 2^\text{AP} \):
  - \( \pi, t, \tau \models x \sim c \) iff \( \tau(x) \sim c \)
  - \( \pi, t, \tau \models x. \phi \) iff \( \pi, i, \tau[x\leftarrow 0] \models \phi \)
  - \( \pi, t, \tau \models \phi \cup \psi \) iff there exists some \( u > 0 \) s.t.
    - \( \pi, t + u, \tau + u - t \models \psi \),
    - \( \pi, i + k, \tau + v - t \models \phi \) for all \( 0 < v < u \).
Timed logics in the continuous framework

- **Syntax of TPTL:**

\[
\text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \mathbf{U} \varphi \mid x. \varphi
\]

- **Continuous semantics of TPTL:** over \( \pi : \mathbb{R}^+ \rightarrow 2^{\text{AP}} \):

  - \( \pi, t, \tau \models x \sim c \) iff \( \tau(x) \sim c \)
  - \( \pi, t, \tau \models x. \varphi \) iff \( \pi, i, \tau[x \leftarrow 0] \models \varphi \)
  - \( \pi, t, \tau \models \varphi \mathbf{U} \psi \) iff there exists some \( u > 0 \) s.t.
    - \( \pi, t + u, \tau + u - t \models \psi, \)
    - \( \pi, i + k, \tau + v - t \models \varphi \) for all \( 0 < v < u \).

- **Examples:**

\[
\begin{array}{c}
0 \hspace{1cm} 1 \hspace{1cm} 2 \\
\text{red} \hspace{2cm} \text{blue} \hspace{2cm} \text{green}
\end{array}
\]

\( x.((\text{red} \lor \text{blue}) \mathbf{U} (\text{green} \land x \leq 2)) \)
Timed logics in the continuous framework

- Syntax of TPTL:

\[ \text{TPTL} \ni \phi ::= p \mid x \sim c \mid \neg \phi \mid \phi \lor \phi \mid \phi \mathcal{U} \psi \mid x. \phi \]

- Continuous semantics of TPTL: over \( \pi : \mathbb{R}^+ \to 2^{\text{AP}} \):
  - \( \pi, t, \tau \models x \sim c \) iff \( \tau(x) \sim c \)
  - \( \pi, t, \tau \models x. \phi \) iff \( \pi, i, \tau[x \leftarrow 0] \models \phi \)
  - \( \pi, t, \tau \models \phi \mathcal{U} \psi \) iff there exists some \( u > 0 \) s.t.
    - \( \pi, t + u, \tau + u - t \models \psi \),
    - \( \pi, i + k, \tau + v - t \models \phi \) for all \( 0 < v < u \).

- Examples:

\[ x. F(\text{blue} \land F(\text{green} \land x \leq 2)) \]
Outline

16 Timed temporal logics
   - Definitions
   - Expressiveness and complexity

17 TPTL vs MTL

18 Timed logics and timed automata
Outline of today’s lecture

16 Timed temporal logics
   - Definitions
   - Expressiveness and complexity

17 TPTL vs MTL

18 Timed logics and timed automata
MTL and TPTL are very expressive

Lemma

*The halting problem for a Turing machine can be encoded in TPTL and MTL (with past) in both (pointwise and continuous) frameworks.*
MTL and TPTL are very expressive

Lemma

The halting problem for a Turing machine can be encoded in TPTL and MTL (with past) in both (pointwise and continuous) frameworks.

Proof (sketch).

- the successive configurations of the Turing machine are encoded on a one-time-unit-long segment;
MTL and TPTL are very expressive

Lemma

The halting problem for a Turing machine can be encoded in TPTL and MTL (with past) in both (pointwise and continuous) frameworks.

Proof (sketch).

- the successive configurations of the Turing machine are encoded on a one-time-unit-long segment;
- a transition of the Turing machine is applied between one configuration and its successor;
MTL and TPTL are very expressive

Lemma

The halting problem for a Turing machine can be encoded in TPTL and MTL (with past) in both (pointwise and continuous) frameworks.

Proof (sketch).

- the successive configurations of the Turing machine are encoded on a one-time-unit-long segment;
- a transition of the Turing machine is applied between one configuration and its successor;
MTL and TPTL are very expressive

**Lemma**

The halting problem for a Turing machine can be encoded in TPTL and MTL (with past) in both (pointwise and continuous) frameworks.

**Proof (sketch).**

- The successive configurations of the Turing machine are encoded on a one-time-unit-long segment;
- A transition of the Turing machine is applied between one configuration and its successor;
MTL and TPTL are very expressive

Lemma

The halting problem for a Turing machine can be encoded in TPTL and MTL (with past) in both (pointwise and continuous) frameworks.

Proof (sketch).

- the successive configurations of the Turing machine are encoded on a one-time-unit-long segment;
- a transition of the Turing machine is applied between one configuration and its successor;
MTL and TPTL are very expressive

Lemma

The halting problem for a Turing machine can be encoded in TPTL and MTL (with past) in both (pointwise and continuous) frameworks.

Proof (sketch).

- the successive configurations of the Turing machine are encoded on a one-time-unit-long segment;
- a transition of the Turing machine is applied between one configuration and its successor;
- the final state of the Turing machine is eventually reached.

\[
\begin{align*}
\text{tick} & \quad b & b & a & q & b & a & \quad \text{tick} & \quad b & b \ q' \ b \\
\text{n} & & & & & & & & \text{n+1}
\end{align*}
\]
MTL and TPTL are very expressive

**Theorem**

*Satisfiability of an MTL- or TPTL-formula is undecidable.*
MTL and TPTL are very expressive

**Theorem**

*Satisfiability of an MTL- or TPTL-formula is undecidable.*

**Definition**

MITL is a (syntactic) fragment of MTL where *punctuality* is not allowed: intervals cannot be singletons.
MTL and TPTL are very expressive

**Theorem**

*Satisfiability of an MTL- or TPTL-formula is undecidable.*

**Definition**

MITL is a (syntactic) fragment of MTL where *punctuality* is not allowed: intervals cannot be singletons.

**Theorem (Alur, Feder, Henzinger, 1991)**

*In the continuous semantics, with any MITL formula, we can associate a timed automaton that accepts exactly the same set of timed state sequences.*
MTL and TPTL are very expressive

Theorem
*Satisfiability of an MTL- or TPTL-formula is undecidable.*

Definition
MITL is a (syntactic) fragment of MTL where *punctuality* is not allowed: intervals cannot be singletons.

Theorem (Alur, Feder, Henzinger, 1991)
*In the continuous semantics, satisfiability of an MITL formula is EXPSPACE-complete.*
MTL and TPTL are very expressive

Theorem

*Satisfiability of an MTL- or TPTL-formula is undecidable.*

Definition

MITL is a (syntactic) fragment of MTL where *punctuality* is not allowed: intervals cannot be singletons.

Theorem (Alur, Feder, Henzinger, 1991)

*In the continuous semantics, satisfiability of an MITL formula is EXPSPACE-complete.*

Corollary

*In the continuous semantics, there is no translation from MTL into MITL.*
Outline

16 Timed temporal logics
   • Definitions
   • Expressiveness and complexity

17 TPTL vs MTL

18 Timed logics and timed automata
Outline of today’s lecture

16 Timed temporal logics
   - Definitions
   - Expressiveness and complexity

17 TPTL vs MTL

18 Timed logics and timed automata
Relative expressiveness of TPTL and MTL

Clearly, MTL can be translated into TPTL:

\[ \varphi \text{ U}_{I} \psi \equiv x. \varphi \text{ U}(\psi \land x \in I). \]
Relative expressiveness of TPTL and MTL

Conversely, consider the following TPTL formula:

\[ G[\text{green} \Rightarrow x. F(\text{red} \land F(\text{blue} \land x \leq 2))] \].

It characterizes the following pattern:
Relative expressiveness of TPTL and MTL

Conversely, consider the following TPTL formula:

\[ G[\text{green} \Rightarrow x. F(\text{red} \land F(\text{blue} \land x \leq 2))] \].

It characterizes the following pattern:

\[ \begin{array}{c}
0 & 1 & 2 \\
\text{green} & \text{red} & \text{blue}
\end{array} \]

Conjecture (Alur, Henzinger, 1990)

Formula

\[ G[\text{green} \Rightarrow x. F(\text{red} \land F(\text{blue} \land x \leq 2))] \].

cannot be expressed in MTL.
Relative expressiveness of TPTL and MTL

In fact, the formula can be expressed:

\[ G(\text{green} \Rightarrow \begin{cases} \text{red} \land F[0,1] \\ \text{blue} \lor F[0,1](\text{red} \land F[0,1]\text{blue}) \\ \text{blue} \land F[-1,0]\text{red} \end{cases}) \]
Relative expressiveness of TPTL and MTL

In fact, the formula can be expressed:

\[
G(\text{green} \Rightarrow \begin{cases} \end{cases})
\]

\[
F_{[0,1]} \text{ red} \land F_{[1,2]} \text{ blue}
\]
Relative expressiveness of TPTL and MTL

In fact, the formula can be expressed:

$$G(\text{green} \Rightarrow \begin{cases} F[0,1] \text{ red } \land F[1,2] \text{ blue} \\ \lor \\ F[0,1](\text{red } \land F[0,1] \text{ blue}) \end{cases})$$
Relative expressiveness of TPTL and MTL

In fact, the formula can be expressed:

\[
G(\text{green} \Rightarrow \begin{cases} F_{[0,1]} \text{red} \land F_{[1,2]} \text{blue} \\
\land \\
F_{[0,1]}(\text{red} \land F_{[0,1]} \text{blue}) \end{cases})
\]
Relative expressiveness of TPTL and MTL

In fact, the formula can be expressed:

\[ G(\text{green} \Rightarrow \begin{cases} F_{[0,1]} \text{red} \land F_{[1,2]} \text{blue} \\ \lor \\ F_{[0,1]}(\text{red} \land F_{[0,1]} \text{blue}) \\ \lor \\ F_{[1,2]}(\text{blue} \land F_{[-1,0]}^{-1} \text{red}) \end{cases} \]
Relative expressiveness of TPTL and MTL

In fact, the formula can be expressed:

\[ G(green \Rightarrow \begin{cases} F_{[0,1]}(red \land F_{[0,1]}(blue)) \\ \lor \\ \lor F_{[0,1]}(red \land F_{[1,2]}(blue)) \end{cases} = 1 \]
Relative expressiveness of TPTL and MTL

In fact, the formula can be expressed:

$$G(green \Rightarrow \begin{cases} 
F_{[0,1]} \text{red} \land F_{[1,2]} \text{blue} \\
\lor \\
F_{[0,1]}(\text{red} \land F_{[0,1]} \text{blue}) \\
\lor \\
F_{[0,1]}(F(0,1) \text{red} \land F_{=1} \text{blue}) 
\end{cases}$$
Relative expressiveness of TPTL and MTL

In fact, the formula can be expressed:

\[
G(\text{green} \Rightarrow \begin{cases} 
F_{[0,1]} \text{red} \land F_{[1,2]} \text{blue} \\
\lor \\
F_{[0,1]}(\text{red} \land F_{[0,1]} \text{blue}) \\
\lor \\
F_{[0,1]}(F_{(0,1)} \text{red} \land F_{=1} \text{blue})
\end{cases})
\]

Lemma (Bouyer, Chevalier, Markey, 2005)

The formula can be expressed
- in MTL in the continuous framework,
- in MITL \(+\text{Past}\) in the pointwise framework.
The pointwise framework

Theorem (Bouyer, Chevalier, Markey, 2005)

*TPTL is strictly more expressive than MTL in the pointwise semantics.*
The pointwise framework

Theorem (Bouyer, Chevalier, Markey, 2005)

TPTL is strictly more expressive than MTL in the pointwise semantics.

Proof (sketch).

- Let $\varphi = x \cdot \mathbf{F}(\text{red} \land \mathbf{F}(\text{blue} \land x \leq 2))$
- Let $\psi$ be an MTL formula.
  - $n =$ temporal height of $\psi =$ maximum number of nested modalities in $\psi$,
  - $p =$ granularity of $\psi =$ inverse of the least common denominator of the constants appearing in $\psi$.
- $\rightarrow$ we assume that the constants of $\psi$ are multiples of $p$. 
The pointwise framework

Theorem (Bouyer, Chevalier, Markey, 2005)

*TPTL is strictly more expressive than MTL in the pointwise semantics.*

**Proof (sketch).**
The pointwise framework

**Theorem (Bouyer, Chevalier, Markey, 2005)**

*\( \text{TPTL is strictly more expressive than MTL in the pointwise semantics.} \)

**Proof (sketch).**

\[
\begin{align*}
\mathcal{A}_{p,n} & \models \quad 2-p \quad p/n \quad 2 \quad p/4n \\
& \quad b \quad b \quad b \quad br \quad br \quad br \quad br \\
\mathcal{B}_{p,n} & \models \quad 2-p \quad 2 \\
& \quad b \quad b \quad b \quad b \quad br \quad br \quad br
\end{align*}
\]

**Lemma**

*For any formula \( \psi \) of MTL with temporal height \( n \) and granularity \( p \),

\[
\mathcal{A}_{n+3,p}, 0 \models \psi \iff \mathcal{B}_{n+3,p}, 0 \models \psi
\]
The pointwise framework

Theorem (Bouyer, Chevalier, Markey, 2005)

*TPTL is strictly more expressive than MTL in the pointwise semantics.*

Proof (sketch).

\[ A_{p,n} \models \cdots \quad b \quad b \quad b \quad br \quad br \quad br \quad br \]

\[ B_{p,n} \models \cdots \quad b \quad b \quad b \quad b \quad br \quad br \quad br \quad br \]

Lemma

*The formula*

\[ \varphi = x. \ F(\text{red} \land F(\text{blue} \land x \leq 2)) \]

*distinguishes between \( A_{p,n} \) and \( B_{p,n} \).*
The pointwise framework

**Theorem (Bouyer, Chevalier, Markey, 2005)**

*TPTL is strictly more expressive than MTL in the pointwise semantics.*

**Proof (sketch).**

<table>
<thead>
<tr>
<th>(A_{p,n})</th>
<th>(-\ -\ -\ -\ -\ -\ -\ -\ -\ -)</th>
<th>(2-p)</th>
<th>(p/n)</th>
<th>(2)</th>
<th>(p/4n)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(b) (b) (b) (br) (br) (br) (br)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(B_{p,n})</th>
<th>(-\ -\ -\ -\ -\ -\ -\ -\ -\ -)</th>
<th>(2-p)</th>
<th>(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(b) (b) (b) (b) (br) (br) (br)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Corollary**

*In the pointwise framework:*

- *MTL+Past is strictly more expressive than MTL.*
- *MITL+Past is strictly more expressive than MITL.*
The continuous framework

Theorem (Bouyer, Chevalier, Markey, 2005)

*TPTL* is strictly more expressive than *MTL* in the continuous semantics.
The continuous framework

Theorem (Bouyer, Chevalier, Markey, 2005)

*TPTL is strictly more expressive than MTL in the continuous semantics.*

Proof (sketch).

Let $\varphi = x \cdot F(green \land x \leq 1 \land G(x \leq 1 \Rightarrow \neg blue))$
The continuous framework

Theorem (Bouyer, Chevalier, Markey, 2005)

*TPTL is strictly more expressive than MTL in the continuous semantics.*

**Proof (sketch).**

Let \( \varphi = x. \mathbf{F}(\text{green} \land x \leq 1 \land \mathbf{G}(x \leq 1 \Rightarrow \neg \text{blue})) \)

\[ A'_{p,n} \quad B'_{p,n} \]

\[ n \text{ times green} \]
The continuous framework

Theorem (Bouyer, Chevalier, Markey, 2005)

*TPTL is strictly more expressive than MTL in the continuous semantics.*

**Proof (sketch).**
Let $\varphi = x. \mathbf{F}(\text{green} \land x \leq 1 \land \mathbf{G}(x \leq 1 \Rightarrow \neg \text{blue}))$

$\mathcal{A}_{p,n}', \mathcal{B}_{p,n}'$

$n$ times green
The continuous framework

**Theorem (Bouyer, Chevalier, Markey, 2005)**

*TPTL is strictly more expressive than MTL in the continuous semantics.*

**Proof (sketch).**

\[
\begin{align*}
A'_{p,n} & = \begin{array}{ccccccc}
\ldots & 1 & 0 & 1 & 0 & 1 & 0 & \ldots \\
\ldots & p/2 & - & - & - & p/2 & \ldots
\end{array} \\
B'_{p,n} & = \begin{array}{ccccccc}
\ldots & 1 & 0 & 1 & 0 & 1 & 0 & \ldots \\
\ldots & p/2 & - & - & - & p/2 & \ldots
\end{array}
\]

Lemma

*For any formula \( \psi \) of MTL with temporal height \( n \) and granularity \( p \),*

\[
A'_{n+3,p},0 \models \psi \iff B'_{n+3,p},0 \models \psi
\]
The continuous framework

Theorem (Bouyer, Chevalier, Markey, 2005)

TPTL is strictly more expressive than MTL in the continuous semantics.

Proof (sketch).

Lemma

The formula

$$\varphi = x \cdot F(green \land x \leq 1 \land G(x \leq 1 \Rightarrow \neg blue))$$

distinguishes between $A_{p,n}$ and $B_{p,n}$. 
The continuous framework

**Theorem (Bouyer, Chevalier, Markey, 2005)**

*TPTL is strictly more expressive than MTL in the continuous semantics.*

**Proof (sketch).**

\[ A'_{p,n} \]

\[ B'_{p,n} \]

\[ n \text{ times green} \]

**Corollary**

*In the continuous framework:*

- *MTL* + *Past* is strictly more expressive than *MTL*.
- *MITL* + *Past* is strictly more expressive than *MITL*. 
Outline

16 Timed temporal logics
   - Definitions
   - Expressiveness and complexity

17 TPTL vs MTL

18 Timed logics and timed automata
Outline of today’s lecture

16 Timed temporal logics
   • Definitions
   • Expressiveness and complexity

17 TPTL vs MTL

18 Timed logics and timed automata
Two-way timed automata

In the sequel, we assume the pointwise semantics.
Two-way timed automata

In the sequel, we assume the pointwise semantics.

Definition (Alur & Henzinger, 1992)

A \textit{two-way timed automaton} is a tuple $\mathcal{A} = \langle Q, Q_0, C, \rightarrow, \Sigma, \ell \rangle$ where

- $Q$, $Q_0$, $C$, $\Sigma$ and $\ell$ are similar to the case of classical timed automata,
- each transition in $\rightarrow$ is decorated with a direction, telling whether the letter to be read at the next step is the one ahead or the one before.
Two-way timed automata

In the sequel, we assume the pointwise semantics.

**Definition (Alur & Henzinger, 1992)**

A *two-way timed automaton* is a tuple $A = \langle Q, Q_0, C, \rightarrow, \Sigma, \ell \rangle$ where

- $Q$, $Q_0$, $C$, $\Sigma$ and $\ell$ are similar to the case of classical timed automata,
- each transition in $\rightarrow$ is decorated with a direction, telling whether the letter to be read at the next step is the one ahead or the one before.

Two-way timed automata might also have an acceptance condition.
Two-way timed automata

Example

$\begin{align*}
q_0 & \xrightarrow{\text{green, } x:=0} q_1 \\
q_3 & \xrightarrow{\text{yellow, } y>2} q_1 \\
q_3 & \xleftarrow{\text{red, } x<1} q_2 \\
q_2 & \xrightarrow{\text{blue, } y:=0} q_1 \\
\end{align*}$
Two-way timed automata

Example

(q₀ \rightarrow q₁) \text{green, } x := 0
(q₁ \rightarrow q₃) \text{yellow, } y > 2
(q₃ \rightarrow q₂) \text{red, } x < 1
(q₂ \rightarrow q₀) \text{blue, } y := 0

(green, 0.2)(blue, 0.6)(red, 1.1)(yellow, 3.1)
Two-way timed automata

Example

- From state $q_0$: green, $x:=0$
- From state $q_1$: blue
- From state $q_2$: red, $x<1$
- From state $q_3$: yellow, $x>2$
- From state $q'_1$: blue, $x:=0$
- From state $q'_2$: red
Two-way timed automata

Example

\[(\text{green}, 0.2)(\text{blue}, 0.6)(\text{red}, 1.1)(\text{yellow}, 3.1)\ldots\]
Two-way timed automata

Theorem (Alur & Henzinger, 1992)

*Two-way timed automata are more expressive than one-way timed automata.*
Two-way timed automata

Theorem (Alur & Henzinger, 1992)

Two-way timed automata are more expressive than one-way timed automata.

Proof (sketch).
The language accepted by the following MTL formula is known not to be recognizable by a timed automaton:

\[ G(green \Rightarrow F_{\geq 1} green) \]
Two-way timed automata

Theorem (Alur & Henzinger, 1992)

Two-way timed automata are more expressive than one-way timed automata.

Proof (sketch).
The language accepted by the following MTL formula is known not to be recognizable by a timed automaton:

$$G(green \Rightarrow F_{\geq 1} green)$$

It is accepted by the following two-way timed automaton:
Two-way timed automata

Theorem (Alur & Henzinger, 1992)

*Two-way timed automata are more expressive than one-way timed automata.*

Theorem (Alur & Henzinger, 1992)

*Two-way timed automata are closed under all boolean operations.*
Bounded timed automata

Definition
A two-way timed automaton is \textit{k-bounded} if each letter of the input word is read at most $2k + 1$ times (for any input word).
Bounded timed automata

Definition
A two-way timed automaton is \textit{k-bounded} if each letter of the input word is read at most \(2k + 1\) times (for any input word).

Example
This automaton is 1-bounded:
Bounded timed automata

Definition
A two-way timed automaton is \( k \)-bounded if each letter of the input word is read at most \( 2k + 1 \) times (for any input word).

Example
This automaton is not \( k \)-bounded:

\[
(green, 0)(green, \frac{1}{k+1})(green, \frac{2}{k+1})(green, \frac{3}{k+1})... 
\]
Definition

We write $\text{MITL}^{+}\text{Past}_k$ for the fragment of MITL$^{+}$Past where the number of alternations of $U$ and $S$ is at most $k - 1$. 
Definition

We write $MITL+Past_k$ for the fragment of MITL+Past where the number of alternations of $U$ and $S$ is at most $k - 1$.

Example

$\text{green } U_{[1,2]} ((\text{green } S_{[3,\infty)} \text{ red}) S_{[0,1]} (\text{red } U \text{ blue}))$
MITL+Past and $k$-bounded two-way timed automata

**Definition**

We write $\text{MITL+Past}_k$ for the fragment of MITL+Past where the number of alternations of $U$ and $S$ is at most $k - 1$.

**Example**

\[
green U_{[1,2]} ((\text{green } S_{[3,\infty]} \text{ red}) S_{[0,1]} (\text{red } U \text{ blue}))
\]

This expression belongs to $\text{MITL+Past}_3$. The diagram illustrates the structure of the formula with various operators and conditions.
MITL+Past and $k$-bounded two-way timed automata

Definition
We write $\textit{MITL+Past}_k$ for the fragment of MITL+Past where the number of alternations of $\textbf{U}$ and $\textbf{S}$ is at most $k - 1$.

Theorem (Alur & Henzinger, 1992)
With any formula in $\textit{MITL+Past}_k$, we can associate a deterministic $k$-bounded two-way timed automaton that accepts exactly the same set of timed words.
MITL+Past and $k$-bounded two-way timed automata

**Definition**

We write $\text{MITL}^+\text{Past}_k$ for the fragment of MITL+Past where the number of alternations of $U$ and $S$ is at most $k - 1$.

**Theorem (Alur & Henzinger, 1992)**

*With any formula in MITL$^+$Past$_k$, we can associate a deterministic $k$-bounded two-way timed automaton that accepts exactly the same set of timed words.*

**Corollary**

*The satisfiability problem for MITL$^+$Past is PSPACE-complete.*
**MITL+Past and k-bounded two-way timed automata**

**Definition**

We write $MITL^{+\text{Past}}_k$ for the fragment of MITL+Past where the number of alternations of $U$ and $S$ is at most $k - 1$.

**Theorem (Alur & Henzinger, 1992)**

*With any formula in $MITL^{+\text{Past}}_k$, we can associate a deterministic $k$-bounded two-way timed automaton that accepts exactly the same set of timed words.*

Moreover, it can be proved that the hierarchy is strict:

**Theorem (Alur & Henzinger, 1992)**

$MITL^{+\text{Past}}_k \subsetneq MITL^{+\text{Past}}_{k+2}$
Definition

We write $\text{MITL}^+\text{Past}_k$ for the fragment of MITL$^+$Past where the number of alternations of $U$ and $S$ is at most $k - 1$.

Theorem (Alur & Henzinger, 1992)

With any formula in $\text{MITL}^+\text{Past}_k$, we can associate a deterministic $k$-bounded two-way timed automaton that accepts exactly the same set of timed words.

Theorem (Alur & Henzinger, 1992)

With any formula in $\text{MTL}$, we can associate a deterministic two-way timed automaton that accepts exactly the same set of timed words.
Expressiveness of Temporal Logics

Conclusions

François Laroussinie and Nicolas Markey

Lab. Specification et Verification
ENS Cachan & CNRS, France

August 4, 2006
Summary of the course

Three levels for measuring expressiveness:

- **distinguishing power:**
  - very coarse measure: $\mathcal{L}(\mathbf{X})$ and LTL+Past have the same distinguishing power;
  - proofs generally easy.
Summary of the course

Three levels for measuring expressiveness:

- **distinguishing power:**
  - very coarse measure: $\mathcal{L}(X)$ and LTL$\mathbf{+}$Past have the same distinguishing power;
  - proofs generally easy.

- **expressive power:**
  - finer relation;
  - proofs generally hard:
    - ad hoc techniques: families of models;
    - via other formalisms: first-order logic, automata theory, ...
Summary of the course

Three levels for measuring expressiveness:

- **distinguishing power:**
  - very coarse measure: $\mathcal{L}(\mathbf{X})$ and LTL+Past have the same distinguishing power;
  - proofs generally easy.

- **expressive power:**
  - finer relation;
  - proofs generally hard:
    - ad hoc techniques: families of models;
    - via other formalisms: first-order logic, automata theory, ...

- **succinctness:**
  - very precise measure;
  - proofs generally very involved:
    - through automata theory,
    - Ehrenfeucht-Fraïssé games, ...
Summary of the course

Linear time

LTL

Branching time

CTL

CTL*
Summary of the course

Linear time

- LTL
- $\mathcal{L}(X)$
- $\mathcal{L}(U)$
- LTL+Past

Branching time

- CTL*
- ECTL+
- ECTL
- CTL
- CTL+
Summary of the course

Linear time

- LTL
- $\mathcal{L}(U)$
- $\mathcal{L}(X)$

Branching time

- CTL
- ECTL
- ECTL$^+$
- $\text{CTL}^*$

Trace equiv.

Bisimulation
Summary of the course

Linear time

First-order logic

\( LTL \)

\( \mathcal{L}(U) \)

\( \mathcal{L}(X) \)

Trace equiv.

\( LTL + \text{Past} \)

Branching time

Second-order logic

Monadic path logic

\( CTL^* \)

ECTL

ECTL^+

CTL

CTL^+

Bisimulation
Summary of the course

Linear time

First-order logic

LTL

$\mathcal{L}(U)$

$\mathcal{L}(X)$

LTL+Past

Trace equiv.

Büchi automata

1-w.ABA

Branching time

Second-order logic

Monadic path logic

CTL*

ECTL

ECTL$

B\ddot{u}chi$ automata

Bisimulation

Alternating tree automata