Temporal logics for multi-agent systems
Expressiveness and algorithms

François Laroussinie
LIAFA – Univ. Paris-Diderot

Nicolas Markey
LSV – ENS Cachan

ESSLLI 2015
Barcelona, 10-14 August 2015
Verification of computerized systems

The Zune bug

In order to compute the current date, the Zune player has to compute the current year, given the total number of days elapsed since January 1st, 1980. For instance, if 777 days elapsed since January 1st, 1980, then:

- 366 days elapsed in 1980;
- 365 days elapsed in 1981;
- 46 days elapsed since January 1st, 1982.
In order to compute the current date, the Zune player has to compute the current year, given the total number of days elapsed since January 1st, 1980.
In order to compute the current date, the Zune player has to compute the current year, given the total number of days elapsed since January 1st, 1980.

For instance, if 777 days elapsed since January 1st, 1980, then

- 366 days elapsed in 1980;
- 365 days elapsed in 1981;
- 46 days elapsed since January 1st, 1982.
Verification of computerized systems – The Zune bug

function current_year(int days)
year = 1980;
while (days > 365) {
    if (IsLeapYear(year)) {
        if (days > 366) {
            days -= 366;
            year += 1;
        }
    } else {
        days -= 365;
        year += 1;
    }
}
return(year);
function current_year(int days)
year = 1980;

while (days > 365) {
    if (IsLeapYear(year)) {
        if (days > 366) {
            days -= 366;
            year += 1;
        }
    } else {
        days -= 365;
        year += 1;
    }
}
return(year);

Example
on Feb. 15th, 1982:

\[
\begin{align*}
\text{days} &= 777 \\
\text{year} &= 1980
\end{align*}
\]
function current_year(int days)
year = 1980;
while (days > 365) {
    if (IsLeapYear(year)) {
        if (days > 366) {
            days -= 366;
            year += 1;
        }
    } else {
        days -= 365;
        year += 1;
    }
}
return(year);

Example
on Feb. 15th, 1982:
  days = 777
  year = 1980
  days = 411
  year = 1981
  days = 46
  year = 1982
  days = 46
  year = 1982
function current_year(int days)

year = 1980;

while (days > 365) {
    if (IsLeapYear(year)) {
        if (days > 366) {
            days -= 366;
            year += 1;
        }
    } else {
        days -= 365;
        year += 1;
    }
}

return(year);

Example

on Feb. 15th, 1982:

days = 777
year = 1980

days = 411
year = 1981

days = 46
year = 1982
function current_year(int days)
    year = 1980;
    while (days > 365) {
        if (IsLeapYear(year)) {
            if (days > 366) {
                days -= 366;
                year += 1;
            }
        } else {
            days -= 365;
            year += 1;
        }
    }
    return(year);

Example
on Feb. 15th, 1982:
    days = 777
    year = 1980
    days = 411
    year = 1981
    days = 46
    year = 1982
Verification of computerized systems – The Zune bug

function current_year(int days)
year = 1980;

while (days > 365) {
    if (IsLeapYear(year)) {
        if (days > 366) {
            days -= 366;
            year += 1;
        }
    } else {
        days -= 365;
        year += 1;
    }
}
return(year);

Example
on Dec. 31st, 2008:

days = 9132
year = 1980

days = 366
year = 2008

days = 366
year = 2008
function current_year(int days)
year = 1980;

while (days > 365) {
    if (IsLeapYear(year)) {
        if (days > 366) {
            days -= 366;
            year += 1;
        }
    } else {
        days -= 365;
        year += 1;
    }
}

return(year);

Example
on Dec. 31st, 2008:

- days = 9132
  - year = 1980
  - days = 366
  - year = 2008

- days = 366
  - year = 2008
Verification of computerized systems – The Zune bug

```java
function current_year(int days)
year = 1980;

while (days > 365) {
    if (IsLeapYear(year)) {
        if (days > 366) {
            days -= 366;
            year += 1;
        }
    } else {
        days -= 365;
        year += 1;
    }
}
return(year);
```

Example

- on Dec. 31st, 2008:
  - days = 9132
  - year = 1980
  - days = 366
  - year = 2008
  - days = 366
  - year = 2008
function current_year(int days)
year = 1980;

while (days > 365) {
    if (IsLeapYear(year)) {
        if (days > 366) {
            days -= 366;
            year += 1;
        }
    } else {
        days -= 365;
        year += 1;
    }
}
return(year);

Example

on Dec. 31st, 2008:

days = 9132
year = 1980

days = 366
year = 2008

days = 366
year = 2008
Formal verification

- proofs of correctness;
- exhaustive methods;
- automated techniques.
**Formal verification**

- proofs of correctness;
- exhaustive methods;
- automated techniques.

**Theorem (Turing, 1936)**

*The halting problem for Turing machines is undecidable.*
Formal verification

- proofs of correctness;
- exhaustive methods;
- automated techniques.

Theorem (Turing, 1936)

The halting problem for Turing machines is undecidable.

Different techniques

- (model-based) testing, statistical verification
**Formal verification**

- proofs of correctness;
- exhaustive methods;
- automated techniques.

**Theorem (Turing, 1936)**

*The halting problem for Turing machines is undecidable.*

**Different techniques**

- (model-based) testing,
  statistical verification
- theorem proving
**Formal verification**

- proofs of correctness;
- exhaustive methods;
- automated techniques.

**Theorem (Turing, 1936)**

The halting problem for Turing machines is *undecidable.*

**Different techniques**

- (model-based) testing,
  statistical verification
- theorem proving
- model checking
- ...
Model checking and synthesis

system:

[http://www.embedded.com]

property

\[ A \bigwedge (\neg B.\text{overfull} \land \neg B.\text{dried\_up}) \]

model-checking algorithm

yes/no
Model checking and synthesis

system:

property

[http://www.embedded.com]
Outline of the presentation

1. Introduction

2. Basics of temporal logics
   - convenient formalism for expressing properties of reactive systems
   - efficient verification algorithms for finite-state models

3. Temporal logics for multi-agent systems
   - reasoning about interacting components in complex systems
   - temporal logics for expressing controllability properties
Outline of the presentation

1 Introduction

2 Basics of temporal logics
   - convenient formalism for expressing properties of reactive systems
   - efficient verification algorithms for finite-state models

3 Temporal logics for multi-agent systems
   - reasoning about interacting components in complex systems
   - temporal logics for expressing controllability properties
Modelling reactive systems: finite-state transition systems

We consider finite-state machines for modelling reactive systems.
We consider finite-state machines for modelling reactive systems.

Example (An ATM)

- **idle**
- **check**
- **return card**
- **give cash**
- **pin**
- **wrong pin**
- **amount**
- **balance**
- **check**

Transitions:
- Insert card: idle -> check
- Unknown card: check -> return card
- Card ok: pin -> pin ok
- Wrong pin: pin -> wrong pin
- Pin ok: pin -> pin ok
- Stop: return card -> give cash
- Negative: give cash -> balance
- Positive: give cash -> negative
- Negative balance: balance -> negative

States:
- Idle
- Return card
- Check
- Pin
- Wrong pin
- Amount
Finite-state machines can be defined as product of smaller models.
Finite-state machines can be defined as product of smaller models.

Example (An elevator)

- floor\_0
- serv\_0
- floor\_1
- serv\_1
- floor\_2
- serv\_2
- cabin
- buttons
- doors
- op\_0
- o\_0?
- c\_0!
- s\_0!
- idle\_0
- req\_0
- s\_0?
- r\_0!
- cl\_1
- op\_1
- o\_1?
- c\_1!
- s\_1!
- idle\_1
- req\_1
- s\_1?
- r\_1!
- cl\_2
- op\_2
- o\_2?
- c\_2!
- s\_2!
- idle\_2
- req\_2
- s\_2?
Modelling reactive systems: finite-state transition systems

Finite-state machines can be defined as product of smaller models.

Example (An elevator)

Finite-state machines can be defined as product of smaller models.
Finite-state machines can be defined as product of smaller models.

Example (An elevator)
Modeling reactive systems: finite-state transition systems

Formally:

Definition

A finite-state machine (a.k.a. Kripke structure) is a 4-tuple $\mathcal{M} = \langle Q, q_0, R, \ell \rangle$ where

- $Q$ is a finite (or countable) set of states;
- $q_0 \in Q$ is the initial state;
- $R \subseteq Q \times Q$ is a transition relation satisfying
  \[ \forall q \in Q. \exists q' \in Q. (q, q') \in R \]
- $\ell : Q \rightarrow 2^{AP}$ labels states with atomic propositions.
Finite-state machines are used to represent infinite behaviours.
Modelling reactive systems: finite-state transition systems

Finite-state machines are used to represent infinite behaviours.

Example (Execution tree of a finite-state machine)
Finite-state machines are used to represent infinite behaviours.

Example (Execution tree of a finite-state machine)
Modelling reactive systems: finite-state transition systems

Formally:

**Definition**

- A **finite run** of $M$ from $q$ is a pair $(w, \ell)$ where
  - $w = (q_i)_{0 \leq i \leq n}$ is a finite sequence of states with
    - $q_0 = q$
    - $(q_i, q_{i+1}) \in R$ for all $0 \leq i \leq n - 1$;
  - $\ell$ labels states with atomic propositions;
Modelling reactive systems: finite-state transition systems

Formally:

Definition

- A finite run of $\mathcal{M}$ from $q$ is a pair $(w, \ell)$ where
  - $w = (q_i)_{0 \leq i \leq n}$ is a finite sequence of states with
    - $q_0 = q$
    - $(q_i, q_{i+1}) \in R$ for all $0 \leq i \leq n - 1$;
  - $\ell$ labels states with atomic propositions;
Modelling reactive systems: finite-state transition systems

Formally:

**Definition**
A finite run of $M$ from $q$ is a pair $(w, \ell)$ where

- $w = (q_i)_{0 \leq i \leq n}$ is a finite sequence of states with
  - $q_0 = q$
  - $(q_i, q_{i+1}) \in R$ for all $0 \leq i \leq n - 1$;
- $\ell$ labels states with atomic propositions;
Modelling reactive systems: finite-state transition systems

Formally:

**Definition**

- A **finite run** of $M$ from $q$ is a pair $(w, \ell)$ where
  - $w = (q_i)_{0 \leq i \leq n}$ is a finite sequence of states with
    - $q_0 = q$
    - $(q_i, q_{i+1}) \in R$ for all $0 \leq i \leq n - 1$;
  - $\ell$ labels states with atomic propositions;

- The **execution tree** of $M$ from $q$ is a pair $(T, \ell)$ where
  - $T$ is the set of finite runs of $M$ from $q$;
  - $\ell$ labels runs with the label of their last state.
Modelling reactive systems: finite-state transition systems

Formally:

**Definition**

- A **finite run** of $\mathcal{M}$ from $q$ is a pair $(w, \ell)$ where
  - $w = (q_i)_{0 \leq i \leq n}$ is a finite sequence of states with
    - $q_0 = q$
    - $(q_i, q_{i+1}) \in R$ for all $0 \leq i \leq n - 1$;
  - $\ell$ labels states with atomic propositions;

- The **execution tree** of $\mathcal{M}$ from $q$ is a pair $(T, \ell)$ where
  - $T$ is the set of finite runs of $\mathcal{M}$ from $q$;
  - $\ell$ labels runs with the label of their last state.
Modelling reactive systems: finite-state transition systems

Formally:

**Definition**
- A finite run of $\mathcal{M}$ from $q$ is a pair $(w, \ell)$ where
  - $w = (q_i)_{0 \leq i \leq n}$ is a finite sequence of states with
    - $q_0 = q$
    - $(q_i, q_{i+1}) \in R$ for all $0 \leq i \leq n - 1$;
  - $\ell$ labels states with atomic propositions;
- The execution tree of $\mathcal{M}$ from $q$ is a pair $(T, \ell)$ where
  - $T$ is the set of finite runs of $\mathcal{M}$ from $q$;
  - $\ell$ labels runs with the label of their last state.
Linear-time Temporal Logic (LTL)

atomic propositions: \( \rho, \sigma, \ldots \)

boolean combinators: 
\[
\neg \rho, \rho \lor \sigma, \rho \land \sigma, \ldots
\]

temporal modalities: 
\[
\begin{align*}
\quad & X \rho \\
\quad & \rho \text{next } \sigma \\
\quad & \rho \mathrel{U} \sigma \\
\quad & \rho \mathrel{U} \sigma \\
\end{align*}
\]

A Kripke structure satisfies \( \rho \in \text{LTL} \) if all its infinite executions do:
\[
M \models \rho \iff \forall \pi \in T_M. \pi \models \rho.
\]
Linear-time Temporal Logic (LTL)

- atomic propositions: ○, ◯, ...

A Kripke structure satisfies $\phi \in \text{LTL}$ if all its infinite executions do:

$$\mathcal{M} \models \phi \iff \forall \pi \in \text{T}_\mathcal{M}. \ \pi \models \phi.$$
Linear-time Temporal Logic (LTL)

- atomic propositions: \( \bigcirc, \bigcirc, \ldots \)
- boolean combinators: \( \neg \varphi, \varphi \lor \psi, \varphi \land \psi, \ldots \)

A Kripke structure satisfies \( \varphi \in \text{LTL} \) if all its infinite executions do:
\[ M |\ = \varphi \iff \forall \pi \in T_M . \pi |\ = \varphi. \]
Linear-time Temporal Logic (LTL)

- atomic propositions: , , ...
- boolean combinators: ¬ϕ, ϕ ∨ψ, ϕ ∧ψ, ...
- temporal modalities:
  - Xϕ
  - ϕ U ψ

A Kripke structure satisfies ϕ ∈ LTL if all its infinite executions do:

M|= ϕ ⇐⇒ ∀π∈TM. π|= ϕ.
Linear-time Temporal Logic (LTL)

- atomic propositions: $\circ, \circ, \ldots$

- boolean combinators: $\neg \varphi, \varphi \land \psi, \varphi \lor \psi, \ldots$

- temporal modalities:
  - $\text{next } \varphi$
  - $\varphi \text{ until } \psi$
  - $\text{eventually } \varphi$
  - $\text{always } \varphi$

A Kripke structure satisfies $\varphi \in \text{LTL}$ if all its infinite executions do:

$$M \models \varphi \iff \forall \pi \in T_M \pi \models \varphi.$$
Linear-time Temporal Logic (LTL)

- atomic propositions: \( \bigcirc, \bigcirc, \ldots \)
- boolean combinators: \( \neg \varphi, \varphi \lor \psi, \varphi \land \psi, \ldots \)
- temporal modalities:
  - \( X \varphi \) \( \bigcirc \longrightarrow \varphi \longrightarrow \bigcirc \longrightarrow \bigcirc \longrightarrow \varnothing \) “next \( \varphi \)”
  - \( \varphi U \psi \) \( \bigcirc \longrightarrow \bigcirc \longrightarrow \varphi \longrightarrow \bigcirc \longrightarrow \bigcirc \) “\( \varphi \) until \( \psi \)”
  - true U \( \varphi \equiv F \varphi \) \( \bigcirc \longrightarrow \bigcirc \longrightarrow \bigcirc \longrightarrow \varphi \longrightarrow \bigcirc \) “eventually \( \varphi \)”
  - \( \neg F \neg \varphi \equiv G \varphi \) \( \bigcirc \longrightarrow \varphi \longrightarrow \bigcirc \longrightarrow \varphi \longrightarrow \varphi \longrightarrow \varphi \longrightarrow \varphi \) “always \( \varphi \)”

A Kripke structure satisfies \( \varphi \in \text{LTL} \) if all its infinite executions do:

\[ \mathcal{M} \models \varphi \iff \forall \pi \in \mathcal{T}_\mathcal{M}. \pi \models \varphi. \]
Examples of LTL formulas

F $\text{floor}_1$ \hspace{1cm} the cabin eventually reaches the first floor
Examples of LTL formulas

\[ F \ \text{floor}_1 \]  \quad \text{the cabin eventually reaches the first floor}

\[ G (op_1 \Rightarrow \text{serv}_1) \]  \quad \text{if the door is open, the cabin is present}

\[ GF \ \text{idle}_0 \]  \quad \text{the button is off infinitely many times}

\[ GF \ \text{serv}_0 \Rightarrow GF \ \text{req}_0 \]  \quad \text{serve infinitely many times only if there are infinitely many requests}
Examples of LTL formulas

\[ F \text{ floor}_1 \]  
the cabin eventually reaches the first floor

\[ G(\text{ op}_1 \Rightarrow \text{ serv}_1) \]  
if the door is open, the cabin is present

\[ G(\text{ req}_0 \Rightarrow F \text{ serv}_0) \]  
any request is eventually served
Examples of LTL formulas

- $F \text{floor}_1$: the cabin eventually reaches the first floor
- $G(\text{op}_1 \Rightarrow \text{serv}_1)$: if the door is open, the cabin is present
- $G(\text{req}_0 \Rightarrow F \text{serv}_0)$: any request is eventually served
- $GF \text{idle}_0$: the button is off infinitely many times

Serve infinitely many times only if there are infinitely many requests.
Examples of LTL formulas

- $\mathbf{F \ floor_1}$: the cabin eventually reaches the first floor
- $G(\ op_1 \Rightarrow \ serv_1)$: if the door is open, the cabin is present
- $G(\ req_0 \Rightarrow \ F \ serv_0)$: any request is eventually served
- $GF \ idle_0$: the button is off infinitely many times
- $(GF \ serv_0) \Rightarrow (GF \ req_0)$: serve infinitely many times only if there are infinitely many requests
Satisfiability and model checking

Two main algorithmic problems

Satisfiability:
Input: a formula $\phi$ in LTL;
Output: yes if there exists a Kripke structure $M$ s.t. $M|\models \phi$; no otherwise.

Model checking:
Input: a formula $\phi$ in LTL, and a Kripke structure $M$;
Output: yes if $M|\models \phi$; no otherwise.
Satisfiability and model checking

Two main algorithmic problems

- **Satisfiability:**
  - Input: a formula $\varphi$ in LTL;
  - Output:
    - **yes** if there exists a Kripke structure $\mathcal{M}$ s.t. $\mathcal{M} \models \varphi$;
    - **no** otherwise.
Satisfiability and model checking

Two main algorithmic problems

- **Satisfiability:**
  - Input: a formula $\varphi$ in LTL;
  - Output:
    - yes if there exists a Kripke structure $\mathcal{M}$ s.t. $\mathcal{M} \models \varphi$;
    - no otherwise.

- **Model checking:**
  - Input: a formula $\varphi$ in LTL, and a Kripke structure $\mathcal{M}$;
  - Output:
    - yes if $\mathcal{M} \models \varphi$;
    - no otherwise.
Verifying LTL properties

Theorem ([SC85])
LTL satisfiability and model checking are logspace-equivalent.
Proof (Sketch)
Encode behaviour of Kripke structure as an LTL formula:

\[
p, q
\]

Lemma
\[M, \not|= \phi \iff \neg \phi \land \land \Phi \text{ is satisfiable.}\]
Verifying LTL properties

Theorem ([SC85])

LTL satisfiability and model checking are logspace-equivalent.

Proof (Sketch)

Encode behaviour of Kripke structure as an LTL formula:

\[ p, q \]

\[ \langle G \rangle \begin{cases}
\exists \text{ exactly one state-proposition in } \{p, q, \neg p, \neg q\} \\
\wedge (p \iff (\exists \phi) \wedge (q \iff (\exists \phi) \wedge ((\exists \phi) \Rightarrow (\exists \phi) \wedge ((\exists \phi) \Rightarrow (\exists \phi))))}
\end{cases} \]

Lemma

\[ M, \neg \models \phi \iff \neg \phi \wedge \exists \Phi M \text{ is satisfiable.} \]

Verifying LTL properties

Theorem ([SC85])

LTL satisfiability and model checking are logspace-equivalent.

Proof (Sketch)

Encode behaviour of Kripke structure as an LTL formula:

\[ \begin{align*}
\Phi &= \left( p \land \neg q \right) \lor \left( q \land \neg p \right) \\
&\quad \land \left( p \lor q \right) \\
&\quad \land \left( \neg p \lor q \right) \\
&\quad \land \left( \neg q \lor p \right) \\
&\quad \lor \left( p \Rightarrow X (p \lor q) \right) \\
&\quad \land \left( q \Rightarrow X \neg (p \lor q) \right)
\end{align*} \]

Lemma

\[ M, \neg \models \varphi \iff \neg \varphi \land \Phi \]

\[ \text{M is satisfiable.} \]

Verifying LTL properties

**Theorem ([SC85])**

*LTL satisfiability and model checking are logspace-equivalent.*

**Proof (Sketch)**

Encode behaviour of Kripke structure as an LTL formula:

\[
G \left[ \text{exactly one state-proposition in } \{p, q\} \land (p \iff (p \lor q) \land (q \iff (p \lor q) \land (p \lor q) \Rightarrow X (p \lor q) \land (p \lor q) \Rightarrow X (p \lor q)) \right]
\]

**Lemma**

\[ M, \neg \varphi \iff \neg \varphi \land \Phi \]

\[ M \] is satisfiable.

---

Verifying LTL properties

Theorem ([SC85])

*LTL satisfiability and model checking are logspace-equivalent.*

Proof (Sketch)

Encode behaviour of Kripke structure as an LTL formula:

\[ p, q \]

\[ G \text{ exactly one state-proposition in } \{\circ, \bullet, \bigcirc\} \]

Verifying LTL properties

**Theorem ([SC85])**

LTL satisfiability and model checking are logspace-equivalent.

**Proof (Sketch)**

Encode behaviour of Kripke structure as an LTL formula:

\[
\text{G} \left[ \bigwedge \text{exactly one state-proposition in \{} \text{p, q, r} \text{\} } \wedge (p \Leftrightarrow (\text{p, q, r}) \wedge (q \Leftrightarrow (p \vee q)) \right]
\]

---

Verifying LTL properties

Theorem ([SC85])

*LTL satisfiability and model checking are logspace-equivalent.*

Proof (Sketch)

Encode behaviour of Kripke structure as an LTL formula:

\[
G \left[ \begin{array}{c}
\land \\
\land \\
(p \iff (\bigcirc \lor \bigcirc) \land (q \iff (\bigcirc \lor \bigcirc)) \\
((\bigcirc \lor \bigcirc) \Rightarrow X(\bigcirc \lor \bigcirc)[\land [\bigcirc \Rightarrow X \bigcirc]]
\end{array} \right]
\]

Verifying LTL properties

**Theorem ([SC85])**

*LTL satisfiability and model checking are logspace-equivalent.*

**Proof (Sketch)**

Encode behaviour of Kripke structure as an LTL formula:

\[
G \left[ \begin{array}{c}
\wedge (p \iff (O \lor B)) \land (q \iff (O \lor R)) \\
\wedge [(O \lor B) \implies X(O \lor B)] \land [(R \implies X(R)]
\end{array} \right]
\]

**Lemma**

\[
\mathcal{M}, O \not\models \varphi \iff \neg \varphi \land O \land \Phi_{\mathcal{M}} \text{ is satisfiable.}
\]

Verifying LTL properties

Theorem ([SC85])

*LTL satisfiability can be solved in polynomial space.*

Verifying LTL properties

Theorem ([SC85])

LTL satisfiability can be solved in polynomial space.

Proof (Sketch)

Lemma (Small-model property)

If $\varphi \in LTL$ is satisfiable, it holds in an exponential-size lasso-shaped model.

Verifying LTL properties

Theorem ([SC85])

LTL satisfiability can be solved in polynomial space.

Proof (Sketch)

Lemma (Small-model property)

If $\varphi \in LTL$ is satisfiable, it holds in an exponential-size lasso-shaped model.

formula: $\varphi = G(\bigcirc \Rightarrow F \bigcirc)$

Verifying LTL properties

Theorem ([SC85])

LTL satisfiability can be solved in polynomial space.

Proof (Sketch)

Lemma (Small-model property)

If $\varphi \in \text{LTL}$ is satisfiable, it holds in an exponential-size lasso-shaped model.

formula: $\varphi = \mathbf{G}(\bigcirc \Rightarrow \mathbf{F} \bigcirc)$

subformulas: $\bigcirc, \bigcirc, \mathbf{F} \bigcirc, \bigcirc \Rightarrow \mathbf{F} \bigcirc, \mathbf{G}(\bigcirc \Rightarrow \mathbf{F} \bigcirc)$

Verifying LTL properties

**Theorem ([SC85])**

LTL satisfiability can be solved in polynomial space.

**Proof (Sketch)**

**Lemma (Small-model property)**

If $\varphi \in \text{LTL}$ is satisfiable, it holds in an exponential-size lasso-shaped model.

- **Formula:** $\varphi = \mathbf{G}(\bigcirc \Rightarrow \mathbf{F} \bigcirc)$
- **Subformulas:** $\bigcirc, \bigcirc, \mathbf{F} \bigcirc, \bigcirc \Rightarrow \mathbf{F} \bigcirc, \mathbf{G}(\bigcirc \Rightarrow \mathbf{F} \bigcirc)$

Verifying LTL properties

**Theorem ([SC85])**

\[ \text{LTL satisfiability can be solved in polynomial space.} \]

**Proof (Sketch)**

**Lemma (Small-model property)**

*If \( \varphi \in \text{LTL} \) is satisfiable, it holds in an exponential-size lasso-shaped model.*

**Formulas:**
\[ \varphi = G(\circ \Rightarrow F \bullet) \]

**Subformulas:**
\[ \circ, \bullet, F \circ, F \bullet, G(\circ \Rightarrow F \circ) \]

same subformulas hold at both positions; all \( U \)-formulas fulfilled inbetween

Verifying LTL properties

**Theorem ([SC85])**

LTL satisfiability can be solved in polynomial space.

**Proof (Sketch)**

**Lemma (Small-model property)**

If \( \varphi \in \text{LTL} \) is satisfiable, it holds in an exponential-size lasso-shaped model.

- **formula:** \( \varphi = G(\circ \implies F \circ) \)
- **subformulas:** \( \circ, \circ, F \circ, F \circ \implies F \circ, G(\circ \implies F \circ) \)

Verifying LTL properties

Theorem ([SC85])

*LTL satisfiability can be solved in polynomial space.*

Proof (Sketch)

Lemma (Small-model property)

*If \( \varphi \in LTL \) is satisfiable, it holds in an exponential-size lasso-shaped model.*

Theorem ([SC85])

*LTL model checking and satisfiability are PSPACE-complete.*

Verifying LTL properties – automata-theoretic approach

**Theorem ([VW86])**

For any $\varphi \in \text{LTL}$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

For any $\varphi \in LTL$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Example

Verifying LTL properties – automata-theoretic approach

**Theorem ([VW86])**

For any $\varphi \in LTL$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

$$\forall w. \ w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

**Example**

![Automaton Diagram]

Verifying LTL properties – automata-theoretic approach

**Theorem ([VW86])**

For any \( \varphi \in \text{LTL} \), there exists an exponential-size \( \text{Büchi automaton} \ A_\varphi \) such that

\[
\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).
\]

**Example**

[\[VW86\] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.]
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

For any $\varphi \in \text{LTL}$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

\[ \forall w. \ w \models \varphi \iff w \in \text{Lang}(A_\varphi). \]

Example

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

For any $\varphi \in LTL$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Example

Verifying LTL properties – automata-theoretic approach

**Theorem (\[VW86\])**

*For any \( \varphi \in LTL, \) there exists an exponential-size Büchi automaton \( A_\varphi \) such that*

\[
\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).
\]

**Example**

\[\begin{array}{c}
\neg q_0 & q_1 \\
\end{array}\]

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

Verifying LTL properties – automata-theoretic approach

**Theorem ([VW86])**

For any $\varphi \in \text{LTL}$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

$$\forall w. \ w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

**Example**

[Diagram of a Büchi automaton with states $q_0$, $q_1$, and arcs labeled with $\neg$, $\Rightarrow$, and $\Rightarrow$.]

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

For any $\varphi \in \text{LTL}$, there exists an exponential-size Büchi automaton $A_{\varphi}$ such that

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_{\varphi}).$$

Example

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

For any $\varphi \in LTL$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Example

Verifying LTL properties – automata-theoretic approach

**Theorem ([VW86])**

*For any* $\varphi \in LTL$, *there exists an exponential-size Büchi automaton* $A_\varphi$ *such that*

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

**Example**

![Automaton example]

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

For any $\varphi \in LTL$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

$$\forall w. \ w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Example

Verifying LTL properties – automata-theoretic approach

**Theorem ([VW86])**

For any $\varphi \in LTL$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

$$\forall w. \ w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

**Example**

![Automata Diagram]

Verifying LTL properties – automata-theoretic approach

**Theorem ([VW86])**

For any $\varphi \in LTL$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

**Example**

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

For any $\varphi \in LTL$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Example

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

For any $\varphi \in \text{LTL}$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Example

![Automaton Diagram](image)

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

For any $\varphi \in \text{LTL}$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

$$\forall w. \; w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Proof (Sketch)

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

For any $\varphi \in LTL$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

$$\forall w. \ w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Proof (Sketch)

- use a slightly extended notion of subformula:

$$\begin{array}{c}
\begin{array}{c}
\circ \\
U
\end{array}
\end{array} \sim \begin{array}{c}
\begin{array}{c}
\circ
\end{array}
\end{array} \lor \begin{array}{c}
\begin{array}{c}
\circ
\end{array}
\end{array} \land X(\begin{array}{c}
\begin{array}{c}
\circ
\end{array}
\end{array} \lor \begin{array}{c}
\begin{array}{c}
\circ
\end{array}
\end{array} U \begin{array}{c}
\begin{array}{c}
\circ
\end{array}
\end{array}).
\end{array}$$

Verifying LTL properties – automata-theoretic approach

**Theorem ([VW86])**

For any $\varphi \in LTL$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

$$\forall w. \ w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

**Proof (Sketch)**

- use a slightly extended notion of subformula:
  $$\bigcirc U \bigcirc \leadsto \bigcirc \lor (\bigcirc \land X(\bigcirc U \bigcirc)).$$
- states of $A_\varphi = \text{maximal consistent sets of subformulas}$;
- transition $S \to S'$ whenever for all $X \psi \in S$, it holds $\psi \in S'$;

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

For any $\varphi \in \text{LTL}$, there exists an exponential-size Büchi automaton $A_\varphi$ such that

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Proof (Sketch)

- use a slightly extended notion of subformula:
  \[
  \bigcirc \mathbf{U} \bigcirc \sim \bigcirc \lor (\bigcirc \land \bigcirc \mathbf{X}(\bigcirc \mathbf{U} \bigcirc)).
  \]

- states of $A_\varphi =$ maximal consistent sets of subformulas;
- transition $S \rightarrow S'$ whenever for all $\mathbf{X} \psi \in S$, it holds $\psi \in S'$;
- acceptance condition: infinitely often $\neg (\bigcirc \mathbf{U} \bigcirc)$
  (generalized Büchi condition)

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

For any $\varphi \in \text{LTL}$, there exists an exponential-size $\text{Büchi}$ automaton $A_\varphi$ such that

$$\forall w. \ w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Theorem ([VW86])

Model checking LTL can be performed in time $2^{O(|\varphi|)} \cdot O(|M|)$.

Verifying LTL properties – using alternating automata

**Theorem ([MSS88,Var94])**

For any $\varphi \in \text{LTL}$, there exists a linear-size weak alternating Büchi automaton $A_{\varphi}$ such that

$$\forall w. \ w \models \varphi \iff w \in \text{Lang}(A_{\varphi}).$$

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.

Verifying LTL properties – using alternating automata

**Theorem ([MSS88,Var94])**

For any \( \varphi \in \text{LTL} \), there exists a linear-size **weak alternating Büchi automaton** \( A_\varphi \) such that

\[
\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).
\]

**Example**

![Diagram](image)

---

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.

Verifying LTL properties – using alternating automata

**Theorem ([MSS88,Var94])**

For any $\varphi \in \text{LTL}$, there exists a linear-size **weak alternating Büchi automaton** $A_\varphi$ such that

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

**Example**

![Diagram of an alternating Büchi automaton]

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Theorem ([MSS88,Var94])

For any $\varphi \in \text{LTL}$, there exists a linear-size weak alternating Büchi automaton $A_\varphi$ such that

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Example

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Verifying LTL properties – using alternating automata

**Theorem ([MSS88,Var94])**

For any $\varphi \in \text{LTL}$, there exists a linear-size **weak alternating Büchi automaton** $A_{\varphi}$ such that

$$\forall w. \ w \models \varphi \iff w \in \text{Lang}(A_{\varphi}).$$

**Example**

![Diagram](image)

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.

Verifying LTL properties – using alternating automata

**Theorem ([MSS88,Var94])**

For any $\varphi \in LTL$, there exists a linear-size weak alternating Büchi automaton $A_\varphi$ such that

$$\forall w. \ w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

**Example**

[Diagram of an automaton]

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Verifying LTL properties – using alternating automata

**Theorem ([MSS88,Var94])**

For any $\varphi \in LTL$, there exists a linear-size weak alternating Büchi automaton $A_\varphi$ such that

\[
\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).
\]

**Example**

[Diagram of an alternating automaton]

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.

Verifying LTL properties – using alternating automata

**Theorem ([MSS88,Var94])**

For any \( \varphi \in \text{LTL} \), there exists a linear-size weak alternating Büchi automaton \( A_\varphi \) such that

\[
\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).
\]

**Example**

![Diagram](image)

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Verifying LTL properties – using alternating automata

**Theorem ([MSS88,Var94])**

For any $\varphi \in \text{LTL}$, there exists a linear-size weak alternating Büchi automaton $A_\varphi$ such that

$$\forall w. \ w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

**Example**

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

For any $\varphi \in LTL$, there exists a linear-size weak alternating Büchi automaton $A_\varphi$ such that

$$\forall w. \ w \models \varphi \iff w \in Lang(A_\varphi).$$

Example

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Verifying LTL properties – using alternating automata

**Theorem ([MSS88,Var94])**

For any $\varphi \in \text{LTL}$, there exists a linear-size **weak alternating Büchi automaton** $A_\varphi$ such that

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

**Example**

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

For any $\varphi \in \text{LTL}$, there exists a linear-size weak alternating Büchi automaton $A_\varphi$ such that

$$\forall w. \ w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Example

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.

Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

For any $\varphi \in \text{LTL}$, there exists a linear-size weak alternating Büchi automaton $A_\varphi$ such that

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Example

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Verifying LTL properties – using alternating automata

**Theorem ([MSS88,Var94])**

*For any $\varphi \in \text{LTL}$, there exists a linear-size weak alternating Büchi automaton $A_\varphi$ such that*

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

**Example**

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

For any $\varphi \in \text{LTL}$, there exists a linear-size weak alternating Büchi automaton $A_\varphi$ such that

$$\forall w. \ w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Example

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Theorem ([MSS88,Var94])

For any $\varphi \in LTL$, there exists a linear-size weak alternating Büchi automaton $A_\varphi$ such that

$$\forall w. \ w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Example

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Verifying LTL properties – using alternating automata

**Theorem ([MSS88,Var94])**

For any \( \varphi \in LTL \), there exists a linear-size **weak alternating Büchi automaton** \( A_\varphi \) such that

\[
\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).
\]

**Example**

![DFA Diagram]

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

For any $\varphi \in \text{LTL}$, there exists a linear-size weak alternating Büchi automaton $A_\varphi$ such that

$$\forall w. \ w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Example

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Verifying LTL properties – using alternating automata

**Theorem ([MSS88,Var94])**

For any $\varphi \in \text{LTL}$, there exists a linear-size **weak alternating Büchi automaton** $A_\varphi$ such that

\[
\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).
\]

**Proof (Sketch)**

- one state per **temporal modality, atomic proposition, and negations thereof**;

---

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

For any $\varphi \in \text{LTL}$, there exists a linear-size weak alternating Büchi automaton $A_\varphi$ such that

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_\varphi).$$

Proof (Sketch)

- one state per temporal modality, atomic proposition, and negations thereof;
- transitions defined as boolean formulas:

$$\delta(\psi_1 \text{ U } \psi_2, \bigcirc) = \delta(\psi_2, \bigcirc) \lor \left[ \delta(\psi_1, \bigcirc) \land \psi_1 \text{ U } \psi_2 \right].$$

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Verifying LTL properties – using alternating automata

Theorem ([MSS88, Var94])

For any $\varphi \in \text{LTL}$, there exists a linear-size weak alternating Büchi automaton $A_{\varphi}$ such that

$$\forall w. \quad w \models \varphi \iff w \in \text{Lang}(A_{\varphi}).$$

Proof (Sketch)

- one state per temporal modality, atomic proposition, and negations thereof;
- transitions defined as boolean formulas:
  $$\delta(\psi_1 U \psi_2, \bigcirc) = \delta(\psi_2, \bigcirc) \lor [\delta(\psi_1, \bigcirc) \land \psi_1 U \psi_2].$$
- acceptance condition: Büchi condition over non-until states.

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
Behavioural equivalence for LTL

When are two Kripke structures indistinguishable by LTL?
Behavioural equivalence for LTL

When are two Kripke structures indistinguishable by LTL?

Example
Theorem

Two finitely branching Kripke structures satisfy the same LTL formulas if, and only if, they generate the same sets of traces.
**Theorem**

Two *finitely branching* Kripke structures satisfy the same LTL formulas if, and only if, they generate the same sets of traces.

**Proof (Sketch)**

**Lemma**

If any finite trace of $\mathcal{M}$ can be generated by $\mathcal{N}$, then

$$\text{InfTraces}(\mathcal{M}) \subseteq \text{InfTraces}(\mathcal{N})$$

(for *finitely branching* Kripke structures $\mathcal{M}$ and $\mathcal{N}$).
Behavioural equivalence for LTL

Theorem

Two finitely branching Kripke structures satisfy the same LTL formulas if, and only if, they generate the same sets of traces.

Counter-example
Theorem

Two finitely branching Kripke structures satisfy the same LTL formulas if, and only if, they generate the same sets of traces.

Counter-example

\[ w \in \text{Traces}(M) \iff \exists i > 0. \ w(2i) = \]
Computation-Tree Logic (CTL*)

atomic propositions: , , ...

boolean combinators: ¬ϕ, ϕ ∨ψ, ϕ ∧ψ, ...

temporal modalities:
Xϕ ϕ “nextϕ”
ϕ U ψ ϕ ϕ ψ “ϕ until ψ”

path quantifiers:
Eϕ ϕ Aϕ
ϕ
ϕ
ϕ
ϕ
ϕ
ϕ
Computation-Tree Logic (CTL*)

- atomic propositions: $\bigcirc$, $\blacksquare$, ...

- boolean combinators: $\neg \varphi$, $\varphi \lor \psi$, $\varphi \land \psi$, ...

- temporal modalities:
  - $X \varphi \rightarrow \varphi$  "next $\varphi$"
  - $\varphi \mathbf{U} \psi \rightarrow \varphi \rightarrow \psi$  "$\varphi$ until $\psi$"
Computation-Tree Logic (CTL∗)

- atomic propositions: , , ...
- boolean combinators: ¬ϕ, ϕ ∨ψ, ϕ ∧ψ, ...
- temporal modalities:
  - Xϕ ϕ "next ϕ"
  - ϕ U ψ ϕ ψ "ϕ until ψ"
- path quantifiers:
Examples of CTL formulas

In CTL, each temporal modality must be in the immediate scope of a path quantifier.
Examples of CTL formulas

In CTL, each temporal modality must be in the immediate scope of a path quantifier.

$$\text{AF floor}_1$$

the cabin eventually reaches the first floor
In CTL, each temporal modality must be in the immediate scope of a path quantifier.

- $\text{AF} \, \text{Floor}_1$: the cabin eventually reaches the first floor
- $\text{AG}(\text{Op}_1 \Rightarrow \text{Serv}_1)$: if the door is open, the cabin is present
- $\text{AG}\text{AF} \, \text{Idle}_0$: the button is off infinitely often
Examples of CTL formulas

In CTL, each temporal modality must be in the immediate scope of a path quantifier.

- $\mathsf{AF}\;\text{floor}_1$  
  - the cabin eventually reaches the first floor

- $\mathsf{AG}(\mathsf{op}_1 \Rightarrow \mathsf{serv}_1)$  
  - if the door is open, the cabin is present

- $\mathsf{AG}(\mathsf{req}_0 \Rightarrow \mathsf{AF}\;\mathsf{serv}_0)$  
  - any request is eventually served
Examples of CTL formulas

In CTL, each temporal modality must be in the immediate scope of a path quantifier.

$AF_{\text{floor}_1}$  
the cabin eventually reaches the first floor

$AG(\text{op}_1 \Rightarrow \text{serv}_1)$  
if the door is open, the cabin is present

$AG(\text{req}_0 \Rightarrow AF \text{serv}_0)$  
any request is eventually served

$AG(\text{req}_0 \Rightarrow EF \text{serv}_0)$  
any request can eventually be served
Examples of CTL formulas

In CTL, each temporal modality must be in the immediate scope of a path quantifier.

\[ \text{AF} \text{floor}_1 \quad \text{the cabin eventually reaches the first floor} \]

\[ \text{AG} \left( \text{op}_1 \Rightarrow \text{serv}_1 \right) \quad \text{if the door is open, the cabin is present} \]

\[ \text{AG} \left( \text{req}_0 \Rightarrow \text{AF} \text{serv}_0 \right) \quad \text{any request is eventually served} \]

\[ \text{AG} \left( \text{req}_0 \Rightarrow \text{EF} \text{serv}_0 \right) \quad \text{any request can eventually be served} \]

\[ \text{AGAF} \text{idle}_0 \quad \text{the button is off infinitely often} \]
Expressiveness of CTL

Theorem ([EH86])

**CTL cannot express the following property:**

\[\text{there is a path visiting } \bigcirc \text{ infinitely many times.}\]
Expressiveness of CTL

Theorem ([EH86])

CTL cannot express the following property:

there is a path visiting infinitely many times.

Example

AGAF visited infinitely many times along all branches
Expressiveness of CTL

Theorem ([EH86])

CTL cannot express the following property:

there is a path visiting infinitely many times.

Example

AGAF visited infinitely many times along all branches

Expressiveness of CTL

**Theorem ([EH86])**

*CTL cannot express the following property:*

\[
\text{there is a path visiting } \bigcirc \text{ infinitely many times.}
\]

**Example**

\[
\begin{align*}
\text{AGAF} & \quad \text{visited infinitely many times along all branches} \\
\text{EGEF} & \\
\end{align*}
\]

Expressiveness of CTL

Theorem ([EH86])

CTL *cannot* express the following property:

\[ \text{there is a path visiting } \bigcirc \text{ infinitely many times.} \]

Example

\[ \begin{align*}
\text{AGAF} & \quad \bigcirc \text{ visited infinitely many} \\
& \quad \text{times along all branches} \\
\text{EGEFE} & \quad \bigcirc \text{ there is a path along which} \\
& \quad \bigcirc \text{ can always be visited}
\end{align*} \]

Expressiveness of CTL

Theorem ([EH86])

CTL cannot express the following property:

\[
\text{there is a path visiting } \bigcirc \text{ infinitely many times.}
\]

Example

\[
\begin{align*}
\text{AGAF} & \quad \bigcirc \text{ visited infinitely many times along all branches} \\
\text{EGEF} & \quad \text{there is a path along which } \\
& \quad \bigcirc \text{ can always be visited} \\
\text{EGAF} & \quad \bigcirc
\end{align*}
\]

Expressiveness of CTL

Theorem ([EH86])

CTL cannot express the following property:

there is a path visiting infinitely many times.

Example

AGAF visited infinitely many times along all branches

EGEF there is a path along which can always be visited

EGAF there is a path along which cannot be avoided

Expressiveness of CTL

Theorem ([EH86])

CTL cannot express the following property:

there is a path visiting infinitely many times.

Proof

Expressiveness of CTL

**Theorem ([EH86])**

*CTL cannot express the following property:*

\[
\text{there is a path visiting } \bigcirc \text{ infinitely many times.}
\]

**Proof**

\[
\alpha_i \not\models \text{EGF } \bigcirc \\
\alpha'_i \models \text{EGF } \bigcirc
\]

**Lemma**

\(\alpha_i \text{ and } \alpha'_i \text{ satisfy the same CTL formulas of length at most } i.\)

Expressiveness of CTL

Theorem

Modalities $\text{EX}$, $\text{E U}$ and $\text{EG}$ are sufficient to express any CTL formula.

Proof

$AX \equiv \neg \text{EX} \neg A U \equiv \neg \text{E} (\neg) U (\neg \land \neg) \land \neg \text{EG} \neg$

How to falsify $U$:
never;
otherwise:
pick earliest;
there must be a $\neg$ before.
Expressiveness of CTL

Theorem

Modalities $\textbf{EX}$, $\mathbf{E} \mathbf{U}$ and $\mathbf{EG}$ are sufficient to express any CTL formula.

Proof

$\textbf{AX} \equiv \neg \textbf{EX} \neg \mathbf{A} \mathbf{U} \equiv \neg \mathbf{E} (\neg) \mathbf{U} (\neg \land \neg) \land \neg \mathbf{EG} \neg \mathbf{A}$

How to falsify $\mathbf{U}$: never; otherwise: pick earliest; there must be a $\neg$ before.
Expressiveness of CTL

Theorem

Modalities $\mathbf{EX}$, $\mathbf{E} \mathbf{U}$ and $\mathbf{EG}$ are sufficient to express any CTL formula.

Proof

$\mathbf{AX} \equiv \neg \mathbf{EX} \neg \mathbf{A} \mathbf{U} \equiv \neg \mathbf{E} (\neg \mathbf{A}) \mathbf{U} (\neg \mathbf{A} \land \neg \mathbf{A}) \land \neg \mathbf{EG} \neg \mathbf{A}$

How to falsify $\mathbf{U}$:

never;
otherwise:
pick earliest;
there must be a $\neg$ before.
Theorem

Modalities $\textbf{E X}$, $\textbf{E U}$ and $\textbf{E G}$ are sufficient to express any CTL formula.

Proof

$\textbf{A X} \equiv \neg \textbf{E X} \neg$

$\textbf{A} \textbf{U} \equiv$

$\neg$
Expressiveness of CTL

Theorem

Modalities $\text{EX}$, $\text{E U}$ and $\text{EG}$ are sufficient to express any CTL formula.

Proof

$\text{AX} \equiv \neg \text{EX} \neg A$

$\text{A} \text{U} \equiv \neg E( \neg ) U( \neg \land \neg ) \land \neg \text{EG}$

How to falsify $\text{U}$:

never;
otherwise:
pick earliest;
there must be a $\neg$ before.
Expressiveness of CTL

Theorem

Modalities \( \text{EX}, \text{E U} \) and \( \text{EG} \) are sufficient to express any CTL formula.

Proof

\[
\text{AX} \equiv \neg \text{EX} \\
\neg \text{A U} \equiv \neg \text{E( } \neg \text{ ) U( } \neg \text{ } \land \neg \text{ ) } \land \neg \text{EG} \\
\neg \text{How to falsify } \text{U}: \\
\text{never } ; \\
\text{otherwise: } \text{pick earliest } ; \\
\text{there must be a } \neg \text{ before.}
\]
Expressiveness of CTL

Theorem

Modalities $\text{EX}$, $\text{E} \ U$ and $\text{EG}$ are sufficient to express any CTL formula.

Proof

$\text{AX} \equiv \neg \text{EX} \neg \square$

$\text{A} \bigcirc \ U \bigcirc \equiv$

How to falsify $\bigcirc \ U \bigcirc$:
- never $\bigcirc$;
- otherwise:
  - pick earliest $\bigcirc$;
Expressiveness of CTL

Theorem

Modalities $\mathbf{EX}$, $\mathbf{E\ U}$ and $\mathbf{EG}$ are sufficient to express any CTL formula.

Proof

$\mathbf{AX} \equiv \neg \mathbf{EX} \neg \mathbf{A}$

$\mathbf{A} \mathbf{U} \mathbf{O} \equiv$

How to falsify $\mathbf{O} \mathbf{U} \mathbf{O}$:

- never $\mathbf{O}$;
- otherwise:
  - pick earliest $\mathbf{O}$;
  - there must be a $\neg \mathbf{O}$ before.
Expressiveness of CTL

**Theorem**

Modalities $\text{EX}$, $\text{E} \, \text{U}$ and $\text{EG}$ are sufficient to express any CTL formula.

**Proof**

$\text{AX} \equiv \neg \text{EX} \neg \circ$

$\text{A} \circ \text{U} \circ \equiv \neg \text{E} (\neg \circ) \, \text{U} \, (\neg \circ \land \neg \circ) \land \neg \text{EG} \neg \circ$

How to falsify $\circ \, \text{U} \circ$:

- never $\circ$;
- otherwise:
  - pick earliest $\circ$;
  - there must be a $\neg \circ$ before.
Verifying CTL properties

**Theorem ([CE81, QS82])**

CTL model checking is PTIME-complete.

---

Verifying CTL properties

Theorem ([CE81,QS82])

CTL model checking is PTIME-complete.

Proof

label states with the subformulas they satisfy.


Verifying CTL properties

Theorem ([CE81, QS82])

CTL model checking is PTIME-complete.

Proof

label states with the subformulas they satisfy.

Verifying CTL properties

Theorem ([CE81,QS82])

**CTL model checking is PTIME-complete.**

Proof

label states with the subformulas they satisfy.

![Diagram showing state labeling with subformulas](image)

E F is reachable


Verifying CTL properties

**Theorem ([CE81, QS82])**

CTL model checking is PTIME-complete.

**Proof**

label states with the subformulas they satisfy.

\[ \text{EF} \]

is reachable


Verifying CTL properties

**Theorem ([CE81,QS82])**

**CTL model checking is PTIME-complete.**

**Proof**

Label states with the subformulas they satisfy.

\[ EF \text{ is reachable} \]


Verifying CTL properties

Theorem ([CE81,QS82])

CTL model checking is PTIME-complete.

Proof

label states with the subformulas they satisfy.

\[ \text{EG}(\neg \circ \land \text{EF} \circ) \]

there is a path along which \( \circ \) is always reachable, but never reached

Verifying CTL properties

Theorem ([CE81,QS82])

CTL model checking is PTIME-complete.

Proof

label states with the subformulas they satisfy.

\[ \text{EG}(\neg \bigcirc \land \text{EF} p) \]

there is a path along which \( p \) is always reachable, but never reached

Verifying CTL properties

Theorem ([CE81,QS82])

**CTL model checking is PTIME-complete.**

Proof

label states with the subformulas they satisfy.

\[
EG(\neg \Diamond p \land EF \Box p)
\]

there is a path along which \(\Box p\) is always reachable, but never reached


Verifying CTL properties

Theorem ([CE81,QS82])

CTL model checking is PTIME-complete.

Proof

label states with the subformulas they satisfy.

\[ \text{EG}(\neg \square \land \text{EF } p) \]

there is a path along which \( p \) is always reachable, but never reached


Verifying CTL properties

Theorem ([CE81,QS82])

*CTL model checking is PTIME-complete.*

Theorem ([KVW94])

*CTL model checking on product structures is PSPACE-complete.*

Verifying CTL properties – automata-theoretic approach

This automaton corresponds to

\[ \delta(q_0, \star) = (q_1, q_2) \]

Verifying CTL properties – automata-theoretic approach

Tree automata

This automaton corresponds to

\[
\begin{align*}
E \cup \delta(q_0, q_0) &= (q_0, q_1) \\
\delta(q_0, q_1) &= (q_1, q_1) \\
\delta(q_0, q_2) &= (q_2, q_2) \\
\delta(q_1, \star) &= (q_1, q_1) \\
\end{align*}
\]

Verifying CTL properties – automata-theoretic approach

Tree automata

\[
\delta(q_0, \bullet) = (q_0, q_1) \lor (q_1, q_0)
\]

\[
\delta(q_0, \circ) = (q_1, q_1)
\]

\[
\delta(q_0, \circledast) = (q_2, q_2)
\]

\[
\delta(q_1, \star) = (q_1, q_1)
\]

\[
\delta(q_2, \star) = (q_2, q_2)
\]

### Tree automata

\[
\begin{align*}
\delta(q_0, \bullet) &= (q_0, q_1) \lor (q_1, q_0) \\
\delta(q_0, \circ) &= (q_1, q_1) \\
\delta(q_0, \ast) &= (q_2, q_2) \\
\delta(q_1, \bigstar) &= (q_1, q_1) \\
\delta(q_2, \bigstar) &= (q_2, q_2)
\end{align*}
\]

Verifying CTL properties – automata-theoretic approach

Tree automata

\[
\begin{align*}
\delta(q_0, 0) &= (q_0, q_1) \lor (q_1, q_0) \\
\delta(q_0, \text{star}) &= (q_1, q_1) \\
\delta(q_0, \text{circle}) &= (q_2, q_2) \\
\delta(q_1, \star) &= (q_1, q_1) \\
\delta(q_2, \text{circle}) &= (q_2, q_2)
\end{align*}
\]

Verifying CTL properties – automata-theoretic approach

Tree automata

\[
\begin{align*}
\delta(q_0, \bullet) &= (q_0, q_1) \lor (q_1, q_0) \\
\delta(q_0, \circ) &= (q_1, q_1) \\
\delta(q_0, \star) &= (q_2, q_2) \\
\delta(q_1, \star) &= (q_1, q_1) \\
\delta(q_2, \star) &= (q_2, q_2)
\end{align*}
\]

Verifying CTL properties – automata-theoretic approach

Tree automata

\[
\begin{align*}
\delta(q_0, \bigcirc) &= (q_0, q_1) \lor (q_1, q_0) \\
\delta(q_0, \bullet) &= (q_1, q_1) \\
\delta(q_0, \bigotimes) &= (q_2, q_2) \\
\delta(q_1, \bigstar) &= (q_1, q_1) \\
\delta(q_2, \bigstar) &= (q_2, q_2)
\end{align*}
\]

Verifying CTL properties – automata-theoretic approach

Tree automata

\[ \delta(q_0, \bullet) = (q_0, q_1) \lor (q_1, q_0) \]
\[ \delta(q_0, \circ) = (q_1, q_1) \]
\[ \delta(q_0, \circledast) = (q_2, q_2) \]
\[ \delta(q_1, \circledast) = (q_1, q_1) \]
\[ \delta(q_2, \circledast) = (q_2, q_2) \]

Verifying CTL properties – automata-theoretic approach

Tree automata

\[
\delta(q_0, \bullet) = (q_0, q_1) \lor (q_1, q_0)
\]
\[
\delta(q_0, \circ) = (q_1, q_1)
\]
\[
\delta(q_0, \star) = (q_2, q_2)
\]
\[
\delta(q_1, \times) = (q_1, q_1)
\]
\[
\delta(q_2, \star) = (q_2, q_2)
\]

Verifying CTL properties – automata-theoretic approach

Tree automata

\[ \delta(q_0, \bigcirc) = (q_0, q_1) \lor (q_1, q_0) \]
\[ \delta(q_0, \bigotimes) = (q_1, q_1) \]
\[ \delta(q_0, \bigotimes) = (q_2, q_2) \]
\[ \delta(q_1, \bigotimes) = (q_1, q_1) \]
\[ \delta(q_2, \bigotimes) = (q_2, q_2) \]

Verifying CTL properties – automata-theoretic approach

This automaton corresponds to $E \bigcirc U$

\[\begin{align*}
\delta(q_0, \bigcirc) &= (q_0, q_1) \lor (q_1, q_0) \\
\delta(q_0, \bigcirc) &= (q_1, q_1) \\
\delta(q_0, \otimes) &= (q_2, q_2) \\
\delta(q_1, \bigstar) &= (q_1, q_1) \\
\delta(q_2, \bigstar) &= (q_2, q_2)
\end{align*}\]

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

This automaton corresponds to
\[
\begin{align*}
\text{EG} & \land \text{AF} \\
\delta(q_0, q_1, q_2) & \lor \delta(q_2, q_1, q_1) \\
\delta(q_0, \neg) & = \bot \\
\delta(q_1, \neg) & = (q_1, q_2) \\
\delta(q_2, \star) & = (q_2, q_2) \\
\delta(q_3, \neg) & = (q_3, q_3)
\end{align*}
\]

Detailled explanations during François' lectures.

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

\[
\begin{align*}
\delta(q_0, \neg) &= \bot \\
\delta(q_1, \star) &= (q_2, q_2) \\
\delta(q_2, \star) &= (q_2, q_2) \\
\delta(q_3, \neg) &= (q_2, q_2) \\
\delta(q_0, q_1, q_2) &= (q_1, q_2) \\
\delta(q_1, q_2, q_1) &= (q_1, q_2) \\
\delta(q_2, q_1, q_2) &= (q_1, q_2) \\
\delta(q_3, q_3) &= (q_2, q_2) \\
\delta(q_1, q_2, q_1) &= (q_1, q_2) \\
\delta(q_2, q_1, q_2) &= (q_1, q_2) \\
\delta(q_3, q_3) &= (q_2, q_2) \\
\end{align*}
\]

Detailled explanations during François' lectures.

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

\[
\delta(q_0, \circ) = [(q_1, q_2) \lor (q_2, q_1)] \land (q_3, q_3)
\]

\[
\delta(q_0, \neg \circ) = \bot
\]

\[
\delta(q_1, \circ) = (q_1, q_2) \lor (q_2, q_1)
\]

\[
\delta(q_1, \neg \circ) = \bot
\]

\[
\delta(q_2, \ast) = (q_2, q_2)
\]

\[
\delta(q_3, \circ) = (q_2, q_2)
\]

\[
\delta(q_3, \neg \circ) = (q_3, q_3)
\]

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

\[
\delta(q_0, \bigcirc) = [(q_1, q_2) \lor (q_2, q_1)] \land (q_3, q_3)
\]

\[
\delta(q_0, \neg \bigcirc) = \bot
\]

\[
\delta(q_1, \bigcirc) = (q_1, q_2) \lor (q_2, q_1)
\]

\[
\delta(q_1, \neg \bigcirc) = \bot
\]

\[
\delta(q_2, \star) = (q_2, q_2)
\]

\[
\delta(q_3, \bigcirc) = (q_2, q_2)
\]

\[
\delta(q_3, \neg \bigcirc) = (q_3, q_3)
\]

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

\[
\delta(q_0, \bigcirc) = [(q_1, q_2) \lor (q_2, q_1)] \land (q_3, q_3)
\]

\[
\delta(q_0, \neg \bigcirc) = \bot
\]

\[
\delta(q_1, \bigcirc) = (q_1, q_2) \lor (q_2, q_1)
\]

\[
\delta(q_1, \neg \bigcirc) = \bot
\]

\[
\delta(q_2, \star) = (q_2, q_2)
\]

\[
\delta(q_3, \bigcirc) = (q_2, q_2)
\]

\[
\delta(q_3, \neg \bigcirc) = (q_3, q_3)
\]

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

\[ \delta(q_0, \bigcirc) = [(q_1, q_2) \lor (q_2, q_1)] \land (q_3, q_3) \]
\[ \delta(q_0, \neg \bigcirc) = \bot \]
\[ \delta(q_1, \bigcirc) = (q_1, q_2) \lor (q_2, q_1) \]
\[ \delta(q_1, \neg \bigcirc) = \bot \]
\[ \delta(q_2, \star) = (q_2, q_2) \]
\[ \delta(q_3, \bigcirc) = (q_2, q_2) \]
\[ \delta(q_3, \neg \bigcirc) = (q_3, q_3) \]

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

\[ \delta(q_0, \bigcirc) = [(q_1, q_2) \lor (q_2, q_1)] \land (q_3, q_3) \]
\[ \delta(q_0, \neg \bigcirc) = \bot \]
\[ \delta(q_1, \bigcirc) = (q_1, q_2) \lor (q_2, q_1) \]
\[ \delta(q_1, \neg \bigcirc) = \bot \]
\[ \delta(q_2, \star) = (q_2, q_2) \]
\[ \delta(q_3, \bigcirc) = (q_2, q_2) \]
\[ \delta(q_3, \neg \bigcirc) = (q_3, q_3) \]

Alternating tree automata

\[ \delta(q_0, \bigcirc) = [(q_1, q_2) \lor (q_2, q_1)] \land (q_3, q_3) \]
\[ \delta(q_0, \neg \bigcirc) = \bot \]
\[ \delta(q_1, \bigcirc) = (q_1, q_2) \lor (q_2, q_1) \]
\[ \delta(q_1, \neg \bigcirc) = \bot \]
\[ \delta(q_2, \star) = (q_2, q_2) \]
\[ \delta(q_3, \bigcirc) = (q_2, q_2) \]
\[ \delta(q_3, \neg \bigcirc) = (q_3, q_3) \]

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

\[ \delta(q_0, \circ) = \left[ (q_1, q_2) \lor (q_2, q_1) \right] \land (q_3, q_3) \]

\[ \delta(q_0, \neg\circ) = \bot \]

\[ \delta(q_1, \circ) = (q_1, q_2) \lor (q_2, q_1) \]

\[ \delta(q_1, \neg\circ) = \bot \]

\[ \delta(q_2, \star) = (q_2, q_2) \]

\[ \delta(q_3, \circ) = (q_2, q_2) \]

\[ \delta(q_3, \neg\circ) = (q_3, q_3) \]

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

\[ \delta(q_0, \circ) = [(q_1, q_2) \lor (q_2, q_1)] \land (q_3, q_3) \]

\[ \delta(q_0, \neg \circ) = \bot \]

\[ \delta(q_1, \circ) = (q_1, q_2) \lor (q_2, q_1) \]

\[ \delta(q_1, \neg \circ) = \bot \]

\[ \delta(q_2, \star) = (q_2, q_2) \]

\[ \delta(q_3, \circ) = (q_2, q_2) \]

\[ \delta(q_3, \neg \circ) = (q_3, q_3) \]

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

\[ \delta(q_0, \bigcirc) = [(q_1, q_2) \lor (q_2, q_1)] \land (q_3, q_3) \]
\[ \delta(q_0, \neg \bigcirc) = \bot \]
\[ \delta(q_1, \bigcirc) = (q_1, q_2) \lor (q_2, q_1) \]
\[ \delta(q_1, \neg \bigcirc) = \bot \]
\[ \delta(q_2, \bigstar) = (q_2, q_2) \]
\[ \delta(q_3, \bigcirc) = (q_2, q_2) \]
\[ \delta(q_3, \neg \bigcirc) = (q_3, q_3) \]

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

\[ \delta(q_0, \bigcirc) = [(q_1, q_2) \lor (q_2, q_1)] \land (q_3, q_3) \]
\[ \delta(q_0, \neg \bigcirc) = \bot \]
\[ \delta(q_1, \bigcirc) = (q_1, q_2) \lor (q_2, q_1) \]
\[ \delta(q_1, \neg \bigcirc) = \bot \]
\[ \delta(q_2, \ast) = (q_2, q_2) \]
\[ \delta(q_3, \bigcirc) = (q_2, q_2) \]
\[ \delta(q_3, \neg \bigcirc) = (q_3, q_3) \]

This automaton corresponds to \( \textbf{E} \, \textbf{G} \bigcirc \land \, \textbf{A} \, \textbf{F} \bigcirc \)

Detailed explanations during François’ lectures.

Behavioural equivalence for CTL

When are two Kripke structures indistinguishable by CTL?

Example

\[ EX(\neg EX) \]
Behavioural equivalence for CTL

When are two Kripke structures indistinguishable by CTL?

Example

EX (EX ∧ EX)
When are two Kripke structures indistinguishable by CTL?

Example

$$\text{EX}(\text{EX } \text{red } \land \text{EX } \text{green})$$
Behavioural equivalence for CTL

Definition

A relation $\mathcal{B} \subseteq Q \times Q$ is a **bisimulation** if
A relation $\mathcal{B} \subseteq Q \times Q$ is a bisimulation if all states with the same label have successors that are related by $\mathcal{B}$. This means that if two states are related by $\mathcal{B}$, then any transition from one state to another is also possible from the corresponding state in the other side, and vice versa.
Behavioural equivalence for CTL

Definition
A relation $B \subseteq Q \times Q$ is a **bisimulation** if

- same label;

\[ B \]
Definition

A relation $B \subseteq Q \times Q$ is a bisimulation if

- same label;
- successors on one side have corresponding successors on other side.
Definition

A relation $\mathcal{B} \subseteq Q \times Q$ is a **bisimulation** if

- same label;
- successors on one side have **corresponding successors** on the other side.
Behavioural equivalence for CTL

Definition
A relation $\mathcal{B} \subseteq Q \times Q$ is a \textit{bisimulation} if

- \textit{same label};
- \textit{successors} on one side have \textit{corresponding successors} on other side.
Behavioural equivalence for CTL

Definition

A relation $B \subseteq Q \times Q$ is a **bisimulation** if

- **same label**;
- **successors** on one side have **corresponding successors** on other side.
Behavioural equivalence for CTL

Definition
A relation \( B \subseteq Q \times Q \) is a bisimulation if
- same label;
- successors on one side have corresponding successors on other side.
Behavioural equivalence for CTL

**Theorem ([BCG88])**

(The initial states of) two *finitely branching* Kripke structures satisfy the same CTL formulas if, and only if, they are bisimilar.

Behavioural equivalence for CTL

Theorem ([BCG88])
(The initial states of) two finitely branching Kripke structures satisfy the same CTL formulas if, and only if, they are bisimilar.

Proof

Behavourial equivalence for CTL

**Theorem ([BCG88])**

(The initial states of) two finitely branching Kripke structures satisfy the same CTL formulas if, and only if, they are bisimilar.

**Proof**

- bisimilar states have the same possible futures, hence satisfy same CTL formulas (by induction);

Behavourial equivalence for CTL

**Theorem ([BCG88])**

(The initial states of) two finitely branching Kripke structures satisfy the same CTL formulas if, and only if, they are bisimilar.

**Proof**

- bisimilar states have the same possible futures, hence satisfy same CTL formulas (by induction);
- non-bisimilar states with finite branching can be distinguished (with only $\text{EX}$ and $\text{AX}$).

Behavioural equivalence for CTL

Theorem ([BCG88])

(The initial states of) two finitely branching Kripke structures satisfy the same CTL formulas if, and only if, they are bisimilar.

Counter-example

Computation-Tree Logic (CTL*)

atomic propositions: , , ...
boolean combinators: ¬ϕ, ϕ ∨ψ, ϕ ∧ψ, ...
temporal modalities:
Xϕ ϕ “nextϕ”
ϕ U ψ ϕ ϕ ψ “ϕ until ψ”
path quantifiers:
Eϕ ϕ Aϕ
Computation-Tree Logic (CTL*)

- **atomic propositions:** ○, ●, ...

- **boolean combinators:** ¬φ, φ ∨ ψ, φ ∧ ψ, ...

- **temporal modalities:**
  - Xφ "next φ"
  - φ U ψ "φ until ψ"

- **path quantifiers:**
  - Eφ
  - Aφ
Computation-Tree Logic (CTL*)

\[ \text{AG} (\text{req}_2 \Rightarrow \text{AG} (\text{GF} \text{serv}_i \Rightarrow \text{F} \text{serv}_2)) \]

any request is eventually served along any fair execution
Computation-Tree Logic (CTL*)

\[ \mathbf{A} \mathbf{G}(req_2 \Rightarrow A(GF servi \Rightarrow F serv_2)) \]

any request is eventually served along any fair execution

Expressiveness

- CTL* subsumes LTL and CTL;
- two states are bisimilar iff they satisfy the same CTL* formulas.

Theorem ([ES84])
CTL* model checking is PSPACE-complete.
Computation-Tree Logic (CTL*)

\[ \mathbf{A} \mathbf{G}( \text{req}_2 \Rightarrow \mathbf{A}(\mathbf{G} \mathbf{F} \text{serv}_i \Rightarrow \mathbf{F} \text{serv}_2) ) \]

any request is eventually served along any fair execution

Expressiveness

- CTL* subsumes LTL and CTL;
- two states are bisimilar iff they satisfy the same CTL* formulas.

Theorem ([ES84])

CTL* model checking is PSPACE-complete.

Outline of the presentation

1 Introduction

2 Basics of temporal logics
   - convenient formalism for expressing properties of reactive systems
   - efficient verification algorithms for finite-state models

3 Temporal logics for multi-agent systems
   - reasoning about interacting components in complex systems
   - temporal logics for expressing controllability properties
Example (An elevator)

Reasoning about open systems

Non-determinism due to interactions with users; underspecified controller for the lift.
Reasoning about open systems

Example (An elevator)

Non-determinism due to interactions with users; underspecified controller for the lift.
Reasoning about open systems

Example (An elevator)

Non-determinism due to

- interactions with users;
Reasoning about open systems

Example (An elevator)

Non-determinism due to

- interactions with users;
- underspecified controller for the lift.
Reasoning about open systems

Example (An elevator)

Controllability properties:

does there exist a controller under which the system satisfies some property, whatever the users do?
Reasoning about open systems

Games on graphs

A concurrent game structure is made of

- a Kripke structure
- a set of agents (or players);
- a table indicating the transition to be taken given the actions of the players.
Reasoning about open systems

Games on graphs

A concurrent game structure is made of
- a Kripke structure
Reasoning about open systems

Games on graphs

A concurrent game structure is made of
- a Kripke structure
- a set of agents (or players);
Games on graphs

A concurrent game structure is made of
- a Kripke structure
- a set of agents (or players);
- a table indicating the transition to be taken given the actions of the players.

<table>
<thead>
<tr>
<th>player 1</th>
<th>player 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="image" /></td>
<td><img src="image2.png" alt="image" /></td>
</tr>
<tr>
<td><img src="image3.png" alt="image" /></td>
<td><img src="image4.png" alt="image" /></td>
</tr>
<tr>
<td><img src="image5.png" alt="image" /></td>
<td><img src="image6.png" alt="image" /></td>
</tr>
<tr>
<td><img src="image7.png" alt="image" /></td>
<td><img src="image8.png" alt="image" /></td>
</tr>
<tr>
<td><img src="image9.png" alt="image" /></td>
<td><img src="image10.png" alt="image" /></td>
</tr>
<tr>
<td><img src="image11.png" alt="image" /></td>
<td><img src="image12.png" alt="image" /></td>
</tr>
<tr>
<td><img src="image13.png" alt="image" /></td>
<td><img src="image14.png" alt="image" /></td>
</tr>
</tbody>
</table>

q₀, q₁, q₂
Reasoning about open systems

Games on graphs
A concurrent game structure is made of
- a Kripke structure
- a set of agents (or players);
- a table indicating the transition to be taken given the actions of the players.

Turn-based games
A turn-based game structure is a game where each state is controlled by a single player.
Reasoning about open systems

Example (An elevator)

Controllability properties:

does there exist a controller under which the system satisfies some property, whatever the users do?
Reasoning about open systems

Example (An elevator)

Controllability properties:

does there exist a controller under which the system satisfies some property, whatever the users do?
Reasoning about open systems

Example (An elevator)

Controllability properties:

is there a strategy for the circle player to enforce some property against any strategy of the square player?
Reasoning about open systems

**Strategies**

A *strategy* for a given player is a function telling what to play depending on what has happened previously.
Reasoning about open systems

**Strategies**

A strategy for a given player is a function telling what to play depending on what has happened previously.

**Example**

![Diagram of a network or graph](image-url)
Reasoning about open systems

**Strategies**

A strategy for a given player is a function telling what to play depending on what has happened previously.

**Example**

Strategy for player
Reasoning about open systems

Strategies

A strategy for a given player is a function telling what to play depending on what has happened previously.

Example

Strategy for player

alternately go to \( \bullet \) and \( \circ \) (starting with \( \bullet \)).
Reasoning about open systems

**Strategies**

A *strategy* for a given player is a function telling what to play depending on what has happened previously.

**Example**

Strategy for player

alternately go to  and  
(starting with ).
Reasoning about open systems

Strategies
A strategy for a given player is a function telling what to play depending on what has happened previously.

Example
Strategy for player
alternately go to blue and green (starting with blue).
Reasoning about open systems

**Strategies**

A *strategy* for a given player is a function telling what to play depending on what has happened previously.

**Example**

*Strategy for player*  
alternately go to and (starting with ).
Reasoning about open systems

**Strategies**

A *strategy* for a given player is a function telling what to play depending on what has happened previously.

**Example**

Memoryless strategy for player

```
......
......
......
......
```

![Diagram of memoryless strategy](image)
Reasoning about open systems

Strategies
A strategy for a given player is a function telling what to play depending on what has happened previously.

Example
Memoryless strategy for player □
always go to •.
Reasoning about open systems

Strategies
A strategy for a given player is a function telling what to play depending on what has happened previously.

Example
Memoryless strategy for player always go to .
Reasoning about open systems

**Strategies**

A *strategy* for a given player is a function telling what to play depending on what has happened previously.

**Example**

Memoryless strategy for player always go to ♠️.
Strategies

A strategy for a given player is a function telling what to play depending on what has happened previously.

Example

Memoryless strategy for player always go to .
Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

\[ \langle\langle A \rangle\rangle \varphi \] expresses that A has a strategy to enforce \( \varphi \).
Temporal logics for games: ATL [AHK02]

ATL extends CTL with **strategy quantifiers**

\[\langle A \rangle \varphi\] expresses that A has a strategy to enforce \(\varphi\).

\[\langle \Box \rangle F \text{floor}_1\] there is a controller with which the cabin eventually reaches the first floor (whatever Player \(\square\) does).

Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

\(\langle A \rangle \varphi\) expresses that \(A\) has a strategy to enforce \(\varphi\).

\(\langle \circ \rangle F \text{floor}_1\) there is a controller with which the cabin eventually reaches the first floor (whatever Player \(\square\) does).

\(\langle \circ \rangle G (\text{op}_1 \Rightarrow \text{serv}_1)\) there is a controller for which, if the door is open, the cabin is present.
Temporal logics for games: ATL [AHK02]

ATL extends CTL with strategy quantifiers

$\langle A \rangle \varphi$ expresses that $A$ has a strategy to enforce $\varphi$.

$\langle \bigcirc \rangle F \text{floor}_1$

there is a controller with which the cabin eventually reaches the first floor (whatever Player $\Box$ does).

$\langle \bigcirc \rangle G (\text{op}_1 \Rightarrow \text{serv}_1)$

there is a controller for which, if the door is open, the cabin is present.

$AG (\text{req}_0 \Rightarrow \langle \bigcirc \rangle F \text{serv}_0)$

any request can be granted by the controller

Expressiveness of ATL

Semantics of $\langle A \rangle \varphi$

Existential quantification (over strategies) implicitly includes a universal quantification (over outcomes):

$G, \bigcirc \models \langle A \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\bigcirc, \sigma_A). \pi \models \varphi.$

Proposition $E$ and $A$ can be expressed using $\langle A \rangle \varphi$

Proof

$A\varphi \equiv \langle \emptyset \rangle \varphi$

$E\varphi \equiv \neg \langle \emptyset \rangle \neg \varphi$

Remark

Writing $E\varphi \equiv \langle \text{Agt} \rangle \varphi$ would depend on the set of agents...
Expressiveness of ATL

Semantics of $\langle A \rangle \varphi$

Existential quantification (over strategies) implicitly includes a universal quantification (over outcomes):

$$G, \Box \models \langle A \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\Box, \sigma_A). \pi \models \varphi.$$ 

Proposition

$E$ and $A$ can be expressed using $\langle \cdot \rangle$. 
Expressiveness of ATL

Semantics of $\langle A \rangle \varphi$

Existential quantification (over strategies) implicitly includes a universal quantification (over outcomes):

$G, \Diamond \models \langle A \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\Diamond, \sigma_A). \pi \models \varphi.$

Proposition

$E$ and $A$ can be expressed using $\langle \cdot \rangle$.

Proof

$A \varphi \equiv \langle \emptyset \rangle \varphi \quad E \varphi \equiv \neg \langle \emptyset \rangle \neg \varphi$
Expressiveness of ATL

Semantics of $\langle A \rangle \varphi$

Existential quantification (over strategies) implicitly includes a universal quantification (over outcomes):

$$G, \bigcirc \models \langle A \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\bigcirc, \sigma_A). \pi \models \varphi.$$ 

Proposition

$E$ and $A$ can be expressed using $\langle \cdot \rangle$.

Proof

$$A \varphi \equiv \langle \emptyset \rangle \varphi \quad E \varphi \equiv \neg \langle \emptyset \rangle \neg \varphi$$

Remark

Writing $E \varphi \equiv \langle \text{Agt} \rangle \varphi$ would depend on the set of agents...
Expressiveness of ATL

**Definition**

\[ [A] \varphi \equiv \neg \langle A \rangle \neg \varphi \]
Expressiveness of ATL

Definition

\([A] \varphi \equiv \neg \langle A \rangle \neg \varphi\)

Example

\([A] F \bigcirc\)
Expressiveness of ATL

Definition

\[[A] \varphi \equiv \neg \langle A \rangle \neg \varphi\]

Example

\[[A] F \bigcirc \quad \text{for any strategy of } A, \text{ there is an outcome visiting } \bigcirc\]
Expressiveness of ATL

**Theorem**

\[ \langle A \rangle X, \langle A \rangle G \text{ and } \langle A \rangle U \text{ are not sufficient to express ATL.} \]
Expressiveness of ATL

**Theorem**

\[\langle A \rangle X, \langle A \rangle G \text{ and } \langle A \rangle U\] are not sufficient to express ATL.

**Example**

Relaxed version of until in CTL:

\[E \circ W \equiv E[\circ U \lor G]\]
Expressiveness of ATL

Theorem

\langle \langle A \rangle \rangle X, \langle \langle A \rangle \rangle G and \langle \langle A \rangle \rangle U are not sufficient to express ATL.

Example

Relaxed version of until in CTL:

\[ E W \equiv E [U \lor G] \]
\[ \equiv (E U) \lor (E G) \]
Expressiveness of ATL

Theorem

\[\langle A \rangle X, \langle A \rangle G \text{ and } \langle A \rangle U \text{ are not sufficient to express ATL.}\]

Example

Relaxed version of \textit{until} in CTL:

\[E \bigcirc W \equiv E\left[ U \bigcirc \lor G \bigcirc \right]\]
\[\equiv (E \bigcirc U \bigcirc) \lor (E G \bigcirc)\]

In ATL (actually ATL\(^*\)):

\[\langle A \rangle \bigcirc W \equiv \langle A \rangle\left[ U \bigcirc \lor G \bigcirc \right]\]
Expressiveness of ATL

Theorem

\(\langle A \rangle X, \langle A \rangle G \) and \(\langle A \rangle U\) are not sufficient to express ATL.

Example

Relaxed version of until in CTL:

\[
E \bigcirc W \equiv E \left[ \bigcirc U \lor G \right] \\
\equiv (E \bigcirc U) \lor (E G)
\]

In ATL (actually ATL*):

\[
\langle A \rangle \bigcirc W \equiv \langle A \rangle \left[ U \lor G \right] \\
\neq \langle A \rangle \bigcirc U \lor \langle A \rangle G
\]
Expressiveness of ATL

**Theorem**

$\langle A \rangle X, \langle A \rangle G$ and $\langle A \rangle U$ are not sufficient to express ATL.

**Example**

Relaxed version of until in CTL:

$$E W \equiv E[U \lor G]$$

$$\equiv (E U) \lor (EG)$$

In ATL (actually ATL*):

$$\langle A \rangle W \equiv \langle A \rangle[U \lor G]$$

$$\not\equiv \langle A \rangle U \lor \langle A \rangle G$$

player B player A
Expressiveness of ATL

Theorem

\[ \langle A \rangle X, \langle A \rangle G \text{ and } \langle A \rangle U \text{ are not sufficient to express ATL.} \]

Proof

\[ a_i a_{i-1} a_1 s_{i-1} \neq |\langle A \rangle W \text{ but } s_i' |\langle A \rangle W \]
Expressiveness of ATL

Theorem

\langle\langle A \rangle\rangle_X, \langle\langle A \rangle\rangle_G and \langle\langle A \rangle\rangle_U are not sufficient to express ATL.

Proof

\begin{align*}
\text{s}_i \not\models \langle\langle A \rangle\rangle W & \quad \text{but} \quad \text{s}_i' \models \langle\langle A \rangle\rangle W
\end{align*}
Expressiveness of ATL

Theorem

\( \langle A \rangle X, \langle A \rangle G \) and \( \langle A \rangle U \) are not sufficient to express ATL.

Remark

Turn-based games are determined:

\[ \neg (\langle A \rangle \varphi) \equiv \langle \text{Agt} \setminus A \rangle \neg \varphi. \]
Expressiveness of ATL

Theorem

\(\langle A \rangle X, \langle A \rangle G \) and \(\langle A \rangle U\) are not sufficient to express ATL.

Remark

Turn-based games are determined:

\[ \neg (\langle A \rangle \varphi) \equiv \langle \text{Agt} \setminus A \rangle \neg \varphi. \]

Then:

\[ \langle A \rangle \text{W} \equiv \langle A \rangle \neg ((\neg) U (\neg \wedge \neg)). \]
Expressiveness of ATL

Theorem

\[\langle\langle A\rangle\rangle X, \langle\langle A\rangle\rangle G \text{ and } \langle\langle A\rangle\rangle U\] are not sufficient to express ATL.

Remark

Turn-based games are determined:

\[\neg (\langle\langle A\rangle\rangle \varphi) \equiv \langle\langle \text{Agt} \setminus A\rangle\rangle \neg \varphi.\]

Then:

\[\langle\langle A\rangle\rangle \Box W \equiv \langle\langle A\rangle\rangle \neg ((\neg \Box) U (\neg \Box \land \neg \Box))\]

\[\equiv_{\text{t.b.}} \neg \langle\langle \text{Agt} \setminus A\rangle\rangle (\neg \Box) U (\neg \Box \land \neg \Box)\]
Model checking ATL [AHK02]

Model checking ATL [AHK02]

Theorem ([AHK02])
Model checking ATL over turn-based games is \( \text{PTIME} \)-complete.

Theorem ([AHK02])
Model checking ATL over turn-based games is \( \text{PTIME} \)-complete.

Model checking ATL [AHK02]

Theorem ([AHK02])
Model checking ATL over turn-based games is \( \text{PTIME} \)-complete.

Model checking ATL [AHK02]

Theorem ([AHK02])
Model checking ATL over turn-based games is PTIME-complete.

Model checking ATL [AHK02]

Theorem ([AHK02])
Model checking ATL over turn-based games is PTIME-complete.

Model checking ATL [AHK02]

Model checking ATL [AHK02]

Theorem ([AHK02])
Model checking ATL over turn-based games is \text{PTIME}-complete.

Theorem ([AHK02])
Model checking ATL over turn-based games is PTIME-complete.

Model checking ATL [AHK02]

\[ \langle \bigcirc \rangle F \]
\[ \langle \square \rangle F \]
\[ \langle \bigcirc \rangle G(\langle \square \rangle F) \]

Theorem ([AHK02])
Model checking ATL over turn-based games is PTIME-complete.

Model checking ATL [AHK02]

Theorem ([AHK02])

Model checking ATL over turn-based games is PTIME-complete.

Model checking ATL [AHK02]

\[ p \langle\langle\rangle\rangle F \langle\langle\rangle\rangle F \langle\langle\rangle\rangle G(\langle\langle\rangle\rangle F ) \equiv \langle\langle\rangle\rangle G p \]

Theorem ([AHK02])

Model checking ATL over turn-based games is PTIME-complete.

Model checking ATL [AHK02]

Theorem ([AHK02])
Model checking ATL over turn-based games is PTIME-complete.

Theorem ([AHK02])

Model checking ATL over turn-based games is PTIME-complete.

Theorem ([AHK02])

Model checking ATL over turn-based games is PTIME-complete.

Model checking ATL [AHK02]

Theorem ([AHK02])
Model checking ATL over concurrent games is PTIME-complete.

Model checking ATL [AHK02]

Model checking ATL [AHK02]

Theorem ([AHK02]) Model checking ATL over concurrent games is PTIME-complete.

\[ \langle 1.1 \rangle, \langle 2.2 \rangle, \langle 3.3 \rangle \]
\[ \langle 1.2 \rangle, \langle 1.3 \rangle, \langle 3.2 \rangle \]
\[ \langle 2.1 \rangle, \langle 2.3 \rangle, \langle 3.1 \rangle \]

\[ \langle \langle 1 \rangle \rangle \ F \]

Model checking ATL [AHK02]

Theorem ([AHK02])

Model checking ATL over concurrent games is \textit{PTIME}-complete.

Model checking ATL [AHK02]

Model checking ATL [AHK02]


Model checking ATL over concurrent games is PTIME-complete.

Model checking ATL [AHK02]

Theorem ([AHK02])

Model checking ATL over concurrent games is PTIME-complete.

Model checking ATL [AHK02]


Model checking ATL over concurrent games is PTIME-complete.
Model checking ATL [AHK02]

Theorem ([AHK02]) Model checking ATL over concurrent games is PTIME-complete.

Model checking ATL [AHK02]

Model checking ATL over concurrent games is PTIME-complete.

Theorem ([AHK02])

Model checking ATL over concurrent games is PTIME-complete.
Model checking ATL [AHK02]

⟨1.1⟩, ⟨2.2⟩, ⟨3.3⟩
⟨1.2⟩, ⟨1.3⟩, ⟨3.2⟩
⟨2.1⟩, ⟨2.3⟩, ⟨3.1⟩

if \( m_1 = m_2 \) goto ;
otherwise, if \( m_1 = 1 \) or \( m_2 = 2 \) goto ;
otherwise, goto ;

\[\langle \langle 1 \rangle \rangle F \leadsto \text{for each move of Player 1, need to solve several instances of } \text{SAT} \]

Theorem ([LMO07])
Model checking ATL over concurrent games with symbolic transition table is \( \Delta P_3 \)-complete (hence in \( PSPACE \), \( NP \)-hard).

Model checking ATL [AHK02]

\[ \langle 1.1 \rangle, \langle 2.2 \rangle, \langle 3.3 \rangle \]
\[ \langle 1.2 \rangle, \langle 1.3 \rangle, \langle 3.2 \rangle \]
\[ \langle 2.1 \rangle, \langle 2.3 \rangle, \langle 3.1 \rangle \]

- if \( m_1 = m_2 \) goto \( \bullet \);
- otherwise, if \( m_1 = 1 \) or \( m_2 = 2 \) goto \( \bigcirc \);
- otherwise, goto \( \circ \);

Model checking ATL [AHK02]

Model checking ATL [AHK02]

\[ \langle 1.1 \rangle, \langle 2.2 \rangle, \langle 3.3 \rangle \]
\[ \langle 1.2 \rangle, \langle 1.3 \rangle, \langle 3.2 \rangle \]
\[ \langle 2.1 \rangle, \langle 2.3 \rangle, \langle 3.1 \rangle \]

- if \( m_1 = m_2 \) goto \( \bigcirc \);
- otherwise, if \( m_1 = 1 \) or \( m_2 = 2 \) goto \( \bigcirc \);
- otherwise, goto \( \bigcirc \);

\[ \langle \langle 1 \rangle \rangle \] \text{F} \( \bigcirc \)

Model checking ATL \cite{AHK02}

\[
\langle1.1\rangle, \langle2.2\rangle, \langle3.3\rangle
\]
\[
\langle1.2\rangle, \langle1.3\rangle, \langle3.2\rangle
\]
\[
\langle2.1\rangle, \langle2.3\rangle, \langle3.1\rangle
\]

- if \( m_1 = m_2 \) goto \( \bullet \);
- otherwise, if \( m_1 = 1 \) or \( m_2 = 2 \) goto \( \bigcirc \);
- otherwise, goto \( \bigcirc \);

\[\langle1\rangle \models F \bigcirc\]

\( \leadsto \) for each move of Player 1, need to solve several instances of SAT

Model checking ATL [AHK02]

\begin{align*}
\langle 1.1, 2.2, 3.3 \rangle & \\
\langle 1.2, 1.3, 3.2 \rangle & \\
\langle 2.1, 2.3, 3.1 \rangle & \\
\end{align*}

- if $m_1 = m_2$ goto \( \Box \);
- otherwise, if $m_1 = 1$ or $m_2 = 2$ goto \( \Diamond \);
- otherwise, goto \( \Box \);

\( \llbrace 1 \rrbrace F \Box \)

\( \Rightarrow \) for each move of Player 1, need to solve several instances of SAT

\textbf{Theorem ([LMO07])}

Model checking ATL over concurrent games with symbolic transition table is $\Delta_3^P$-complete (hence in PSPACE, NP-hard).


The full logic ATL$^*$

\[ \text{Theorem (AHK02)} \]

Model checking ATL$^*$ is 2EXPTIME-complete.

Proof

Games with LTL objectives can be solved in 2EXPTIME:

- LTL formula to deterministic parity automaton:
  - size doubly-exponential;
  - number of priorities exponential.
- Solve parity game in \( O(|Q||p|) \).
The full logic ATL*

Theorem ([AHK02])
Model checking ATL* is 2EXPTIME-complete.

Proof
Games with LTL objectives can be solved in 2EXPTIME:
LTL formula to deterministic parity automaton:
size doubly-exponential;
number of priorities exponential.
solve parity game in $O(|Q||p|)$.
The full logic ATL$^*$

$\langle A \rangle \left[ GF^{\text{serv}_i} \Rightarrow G(\text{req}_0 \Rightarrow F^{\text{serv}_0}) \right]$

Theorem ([AHK02])

Model checking ATL$^*$ is 2EXPTIME-complete.

The full logic ATL*

\[ \langle A \rangle \left[ GF \text{serv}_i \Rightarrow G(\text{req}_0 \Rightarrow F \text{serv}_0) \right] \]

Theorem ([AHK02])

Model checking ATL* is 2EXPTIME-complete.

Proof

Games with LTL objectives can be solved in 2EXPTIME:

- LTL formula to deterministic parity automaton:
  - size doubly-exponential;
  - number of priorities exponential.
- solve parity game in \( O(|Q||p|) \).

Behavioural equivalence for ATL

Definition

A relation $\mathcal{B} \subseteq Q \times Q$ is an alternating bisimulation if

- same label;
- any action on one side has a corresponding action on other side.


Definition

A relation $\mathcal{B} \subseteq Q \times Q$ is an alternating bisimulation if

- same label;
- any action on one side has a corresponding action on other side.
A relation $\mathcal{B} \subseteq Q \times Q$ is an alternating bisimulation if

- same label;

\[ \mathcal{B} \]
Behavioural equivalence for ATL

Definition

A relation $B \subseteq Q \times Q$ is an alternating bisimulation if

- same label;
- any action on one side has a corresponding action on other side.
Definition

A relation $\mathcal{B} \subseteq Q \times Q$ is an alternating bisimulation if

- same label;
- any action on one side has a corresponding action on other side.
A relation $B \subseteq Q \times Q$ is an alternating bisimulation if
- same label;
- any action on one side has a corresponding action on other side.
Definition

A relation $\mathcal{B} \subseteq Q \times Q$ is an alternating bisimulation if

- same label;
- any action on one side has a corresponding action on the other side.
A relation $\mathcal{B} \subseteq Q \times Q$ is an alternating bisimulation if

- same label;
- any action on one side has a corresponding action on other side.
Behavioural equivalence for ATL

Definition

A relation $\mathcal{B} \subseteq Q \times Q$ is an alternating bisimultion if

- same label;
- any action on one side has a corresponding action on other side.
A relation $B \subseteq Q \times Q$ is an alternating bisimulation if

- same label;
- any action on one side has a corresponding action on other side.
Behavioural equivalence for ATL

**Definition**

A relation $\mathcal{B} \subseteq Q \times Q$ is an alternating bisimulation if

- same label;
- any action on one side has a corresponding action on other side.

**Theorem ([AHKV98])**

(The initial states of) two concurrent game structures (with finitely many moves) satisfy the same ATL formulas if, and only if, they are alternating-bisimilar.

Example

\[(\Box) \text{G} (\Box \Diamond F \Diamond)\]
ATL with strategy contexts [BDLM09,DLM10]

Example

\[ \langle \Box \rangle \ G( \langle \square \rangle \ F \ C ) \]

ATL with strategy contexts [BDLM09,DLM10]

Example

\[ \langle \Box \rangle G( \langle \square \rangle F \Box ) \]

Player \( \bigcirc \) in \( \bigcirc \) always plays to \( \square \).

Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.
Example

Player $\bigcirc$ in $\bigcirc$ always plays to $\square$.
ATL with strategy contexts [BDLM09,DLM10]

Example

\[ \langle \langle \Box \rangle \rangle G(\langle \langle \Box \rangle \rangle F) \]

- Player \( \Box \) in a always plays to \( \Box \).

Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.
ATL with strategy contexts [BDLM09, DLM10]

Example

- Player $\bigcirc$ in $\bigcirc$ always plays to $\square$;
- Player $\square$ in $\square$ then plays to $\bigcirc$.

Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.
ATL with strategy contexts

Definition

\( \text{ATL}_{sc} \) has several \textbf{new strategy quantifiers}:

\[
\langle \cdot A \cdot \rangle \quad \text{is similar to} \quad \langle \langle A \rangle \rangle \quad \text{but assigns the corresponding strategy} \quad A \quad \text{for evaluating} \quad \phi;
\]

\[
\langle \cdot A \cdot \rangle \equiv \langle \cdot A_{\text{agt}} \setminus A \cdot \rangle, \quad \text{which will be useful to get formulas} \quad \text{do not depend on the underlying set of agents};
\]

\[
\langle \cdot A \cdot \rangle^0 \quad \text{is similar to} \quad \langle \cdot A \cdot \rangle \quad \text{but quantifies over memoryless strategies};
\]

\[
[\cdot A \cdot] \quad \text{drops the assigned strategies for} \quad A;
\]

\[
[\cdot A \cdot] \phi \equiv \neg \langle \cdot A \cdot \rangle \neg \phi
\]
ATL with strategy contexts

Definition

ATL_{sc} has several **new strategy quantifiers**:

- \langle \cdot A \cdot \rangle is similar to \langle \langle A \rangle \rangle but **assigns** the corresponding strategy to A for evaluating \( \varphi \);

\[ \langle \cdot A \cdot \rangle \equiv \langle \cdot \text{Agt} \setminus A \cdot \rangle \]

\[ \langle \cdot A \cdot \rangle_0 \]

\[ \llparenthesis A \rrparenthesis \]

\[ \llbracket \cdot A \cdot \rrbracket \]

\[ \llbracket \cdot A \cdot \rrbracket \varphi \equiv \neg \langle \cdot A \cdot \rangle \neg \varphi \]
ATL with strategy contexts

Definition

\(\text{ATL}_{sc}\) has several \textbf{new strategy quantifiers}:

- \(\langle \cdot ; A ; \cdot \rangle\) is similar to \(\langle A \rangle\) but \textit{assigns} the corresponding strategy to \(A\) for evaluating \(\varphi\);

- \(\langle \cdot ; \overline{A} ; \cdot \rangle \equiv \langle \cdot ; \text{Agt} \setminus A ; \cdot \rangle\), which will be useful to get formulas that \textit{do not depend} on the underlying set of agents;
**ATL with strategy contexts**

**Definition**

\[ \text{ATL}_{sc} \text{ has several new strategy quantifiers:} \]

- \( \langle \cdot A \cdot \rangle \) is similar to \( \langle A \rangle \) but **assigns** the corresponding strategy to \( A \) for evaluating \( \varphi \);

- \( \langle \overline{A} \rangle \equiv \langle \text{Agt} \setminus A \rangle \), which will be useful to get formulas that **do not depend** on the underlying set of agents;

- \( \langle A \rangle_0 \) is similar to \( \langle A \rangle \) but quantifies over **memoryless** strategies;
ATL with strategy contexts

Definition

\( \text{ATL}_{sc} \) has several **new strategy quantifiers**: 

- \( \langle \cdot A \cdot \rangle \) is similar to \( \langle \langle A \rangle \rangle \) but **assigns** the corresponding strategy to \( A \) for evaluating \( \varphi \);

- \( \langle \cdot \overline{A} \cdot \rangle \equiv \langle \cdot \text{Ag}t \ \setminus \ A \cdot \rangle \), which will be useful to get formulas that do not depend on the underlying set of agents;

- \( \langle \cdot A \cdot \rangle_0 \) is similar to \( \langle \cdot A \cdot \rangle \) but quantifies over **memoryless** strategies;

- \( \langle A \rangle \) **drops** the assigned strategies for \( A \);
ATL with strategy contexts

Definition

\( \text{ATL}_{sc} \) has several new strategy quantifiers:

\( \langle \cdot A \cdot \rangle \) is similar to \( \langle \langle A \rangle \rangle \) but assigns the corresponding strategy to \( A \) for evaluating \( \varphi \);

\( \langle \cdot \overline{A} \cdot \rangle \equiv \langle \cdot \text{Agt} \setminus A \cdot \rangle \), which will be useful to get formulas that do not depend on the underlying set of agents;

\( \langle \cdot A \cdot \rangle_0 \) is similar to \( \langle \cdot A \cdot \rangle \) but quantifies over memoryless strategies;

\( \langle \cdot \overline{A} \cdot \rangle \) drops the assigned strategies for \( A \);

\( \langle A \rangle \) drops the assigned strategies for \( A \);

\[ A \] is dual to \( \langle \cdot A \cdot \rangle \) :

\[ [A] \varphi \equiv \neg \langle \cdot A \cdot \rangle \neg \varphi \]
Semantics of $⟨·A·⟩\varphi$

Definition

Semantics of ATL strategy quantifier:

$G, \bigcirc \models ⟨⟨A⟩⟩\varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\bigcirc, \sigma_A). \pi \models \varphi$

formulas evaluated in a context (a strategy for some coalition); context initially empty; newly selected strategies added to the context:

$\sigma_A \circ \sigma_B : a \mapsto \begin{cases} \sigma_A(a) & \text{if } a \in A \setminus B \\ \sigma_B(b) & \text{if } b \in B \setminus A \\ \sigma_A(c) & \text{if } c \in B \cap A \end{cases}$
Semantics of \(\langle A \rangle \varphi\)

**Definition**

Semantics of ATL strategy quantifier:

\[ \mathcal{G}, \bigcirc \models \langle A \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\bigcirc, \sigma_A). \pi \models \varphi \]

Semantics of ATL\(_{sc}\) strategy quantifier:

\[ \mathcal{G}, \bigcirc \models_{\sigma_B} \langle A \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\bigcirc, \sigma_A \circ \sigma_B). \pi \models_{\sigma_A \circ \sigma_B} \varphi \]
Semantics of $\langle \cdot A \cdot \rangle \varphi$

**Definition**

Semantics of ATL strategy quantifier:

$G, \emptyset \models \langle A \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\emptyset, \sigma_A). \pi \models \varphi$

Semantics of $\text{ATL}_{sc}$ strategy quantifier:

$G, \emptyset \models_{\sigma_B} \langle \cdot A \cdot \rangle \varphi \iff \exists \sigma_A. \forall \pi \in \text{Out}(\emptyset, \sigma_A \circ \sigma_B). \pi \models_{\sigma_A \circ \sigma_B} \varphi$

- formulas evaluated in a context (a strategy for some coalition);
- context initially empty;
- newly selected strategies added to the context:

$$\sigma_A \circ \sigma_B : a \mapsto \sigma_A(a) \quad \text{if } a \in A \setminus B$$

$$b \mapsto \sigma_B(b) \quad \text{if } b \in B \setminus A$$

$$c \mapsto \sigma_A(c) \quad \text{if } c \in B \cap A$$
What $\text{ATL}_{sc}$ can express

$A$ has a strategy to eventually reach:

$\langle \cdot A \cdot \rangle F$

$\langle \cdot \emptyset \cdot \rangle \llparenthesis \emptyset \rrparenthesis \langle \cdot A \cdot \rangle F$

$\equiv \langle \langle A \rangle \rangle F$

$\langle \cdot A \cdot \rangle \llbracket \cdot A \cdot \rrbracket F$

$\text{CTL}^*$ and $\text{ATL}^*$ properties:

$\langle \cdot \emptyset \cdot \rangle G \langle \cdot \emptyset \cdot \rangle F$

$\equiv E \ G \ F$

$\langle \cdot \emptyset \cdot \rangle (G \langle \cdot \emptyset \cdot \rangle F \land G \langle \cdot \emptyset \cdot \rangle F )$

$\equiv E (G \ F \land G \ F )$

leads to memory needed!
What $\text{ATL}_{sc}$ can express

- $A$ has a strategy to eventually reach $\Diamond$:
What $\text{ATL}_{sc}$ can express

- $A$ has a strategy to eventually reach $\mathbf{G}$:
  - $\langle A \rangle F \equiv \langle A \rangle F$

- $\langle A \rangle F$ CTL$^*$ and ATL$^*$ properties:
  - $\langle A \rangle G F \equiv E G F$
    - $E(G F \land G F)$

/leadsto memory needed!
What $\text{ATL}_{sc}$ can express

- $A$ has a strategy to eventually reach $\bigcirc$:
  - $\langle \cdot A \cdot \rangle F \bigcirc$
  - $\langle \varnothing \rangle \langle \cdot A \cdot \rangle F \bigcirc$

$\text{CTL}^*$ and $\text{ATL}^*$ properties:

- $\langle \cdot \emptyset \cdot \rangle G \langle \cdot \emptyset \cdot \rangle F \equiv E \left( G \langle \cdot \emptyset \cdot \rangle F \land G \langle \cdot \emptyset \cdot \rangle F \right) \equiv E \left( G F \land G F \right)$

/leadsto memory needed!
What $\text{ATL}_{sc}$ can express

- $A$ has a strategy to eventually reach $\bigcirc$:
  - $\langle \cdot A \cdot \rangle \text{ F } \bigcirc$
  - $\langle \overline{\emptyset} \rangle \langle \cdot A \cdot \rangle \text{ F } \bigcirc \equiv \langle A \rangle \text{ F } \bigcirc$

$\text{CTL}^*$ and $\text{ATL}^*$ properties:

- $\langle \cdot \emptyset \cdot \rangle \text{ G } \langle \cdot \emptyset \cdot \rangle \text{ F } \equiv E \text{ G } \text{ F }$
- $\langle \cdot \emptyset \cdot \rangle (\text{ G } \langle \cdot \emptyset \cdot \rangle \text{ F } \land \text{ G } \langle \cdot \emptyset \cdot \rangle \text{ F } ) \equiv E (\text{ G } \text{ F } \land \text{ G } \text{ F } )$

/leadsto memory needed!
What $\text{ATL}_{sc}$ can express

- $A$ has a strategy to eventually reach $\bigcirc$:
  - $\langle \cdot A \cdot \rangle F \bigcirc$
  - $(\emptyset) \langle \cdot A \cdot \rangle F \bigcirc \equiv \langle A \rangle F \bigcirc$
  - $\langle A \rangle \lbrack \overline{A} \rbrack F \bigcirc$

$\text{CTL}^*$ and $\text{ATL}^*$ properties:

- $\langle \cdot \emptyset \cdot \rangle G \langle \cdot \emptyset \cdot \rangle F \equiv E G F$
- $\langle \cdot \emptyset \cdot \rangle (G \langle \cdot \emptyset \cdot \rangle F \land G \langle \cdot \emptyset \cdot \rangle F) \equiv E (G F \land G F)$
What $\text{ATL}_{sc}$ can express

- $A$ has a strategy to eventually reach $\bigcirc$:
  - $\langle \cdot A \cdot \rangle F \bigcirc$
  - $(\emptyset) \langle \cdot A \cdot \rangle F \bigcirc \equiv \langle A \rangle F \bigcirc$
  - $\langle \cdot A \cdot \rangle [\overline{A}] F \bigcirc$

- CTL* and ATL* properties:
What $\text{ATL}_{sc}$ can express

- $A$ has a strategy to eventually reach $\bigcirc$:
  - $\langle \cdot A \cdot \rangle F \bigcirc$
  - $(\overline{\square}) \langle \cdot A \cdot \rangle F \bigcirc \equiv \langle A \rangle F \bigcirc$
  - $\langle \cdot A \cdot \rangle [\overline{A}] F \bigcirc$

- CTL$^*$ and ATL$^*$ properties:
  - $\langle \overline{\square} \rangle G \langle \cdot \varnothing \cdot \rangle F \bigcirc$

\(\text{leadsto}\) memory needed!
What $\text{ATL}_{sc}$ can express

- $A$ has a strategy to eventually reach $\bigcirc$:
  - $\langle \cdot A \cdot \rangle F \bigcirc$
  - $\langle \overline{\emptyset} \rangle \langle \cdot A \cdot \rangle F \bigcirc \equiv \langle A \rangle F \bigcirc$
  - $\langle A \rangle [\overline{A}] F \bigcirc$

- $\text{CTL}^*$ and $\text{ATL}^*$ properties:
  - $\langle \overline{\emptyset} \rangle G \langle \cdot \emptyset \cdot \rangle F \bigcirc \equiv E G F \bigcirc$
What $\text{ATL}_{sc}$ can express

- $A$ has a strategy to eventually reach $\bigcirc$:
  - $\langle \cdot A \cdot \rangle F \bigcirc$
  - $(\overline{\emptyset}) \langle \cdot A \cdot \rangle F \bigcirc \equiv \langle A \rangle F \bigcirc$
  - $\langle A \rangle [\overline{A}] F \bigcirc$

- CTL* and ATL* properties:
  - $\langle \overline{\emptyset} \rangle G \langle \cdot \emptyset \cdot \rangle F \bigcirc \equiv E G F \bigcirc$
  - $\langle \overline{\emptyset} \rangle (G \langle \cdot \emptyset \cdot \rangle F \bigcirc \land G \langle \cdot \emptyset \cdot \rangle F \bigcirc)$
What $\text{ATL}_{sc}$ can express

- $A$ has a strategy to eventually reach $\bigcirc$:
  - $\langle \cdot A \cdot \rangle F \bigcirc$
  - $\langle \overline{\cdot} \rangle \langle \cdot A \cdot \rangle F \bigcirc \equiv \langle A \rangle F \bigcirc$
  - $\langle A \rangle [\overline{A}] F \bigcirc$

- CTL* and ATL* properties:
  - $\langle \overline{\cdot} \rangle G \langle \cdot \emptyset \cdot \rangle F \bigcirc \equiv E G F \bigcirc$
  - $\langle \overline{\cdot} \rangle (G \langle \cdot \emptyset \cdot \rangle F \bigcirc \land G \langle \cdot \emptyset \cdot \rangle F \bigcirc) \equiv E(G F \bigcirc \land G F \bigcirc)$
What $\text{ATL}_{sc}$ can express

- $A$ has a strategy to eventually reach $\bigcirc$:
  - $\langle \cdot A \cdot \rangle F \bigcirc$
  - $(\overline{\emptyset}) \langle \cdot A \cdot \rangle F \bigcirc \equiv \langle \langle A \rangle \rangle F \bigcirc$
  - $\langle A \rangle \overline{[A]} F \bigcirc$

- $\text{CTL}^*$ and $\text{ATL}^*$ properties:
  - $\langle \overline{\emptyset} \rangle G \langle \cdot \emptyset \cdot \rangle F \bigcirc \equiv E G F \bigcirc$
  - $\langle \overline{\emptyset} \rangle (G \langle \cdot \emptyset \cdot \rangle F \bigcirc \land G \langle \cdot \emptyset \cdot \rangle F \bigcirc) \equiv E(G F \bigcirc \land G F \bigcirc)$
What $\text{ATL}_{sc}$ can express

- $A$ has a strategy to eventually reach $\bigcirc$:
  - $\langle \cdot A \cdot \rangle F \bigcirc$
  - $(\langle \overline{\emptyset} \rangle \langle \cdot A \cdot \rangle F \bigcirc \equiv \langle A \rangle F \bigcirc$
  - $\langle A \rangle [\overline{A}] F \bigcirc$

- $\text{CTL}^*$ and $\text{ATL}^*$ properties:
  - $\langle \overline{\emptyset} \rangle G \langle \cdot \emptyset \cdot \rangle F \bigcirc \equiv E G F \bigcirc$
  - $\langle \overline{\emptyset} \rangle (G \langle \cdot \emptyset \cdot \rangle F \bigcirc \land G \langle \cdot \emptyset \cdot \rangle F \bigcirc) \equiv E (G F \bigcirc \land G F \bigcirc)$

$\rightsquigarrow$ memory needed!
What $\text{ATL}_{sc}$ can express
What $\text{ATL}_{sc}$ can express

- existence of a correct controller for the elevator:

$$\langle \cdot \text{ctrl} \cdot \rangle \text{G} \left[ \bigwedge_{i,j} \langle \cdot \text{user} \cdot \rangle_0 F (\text{op}_i \land F \text{op}_j) \right] \land \left[ \bigwedge_{i \in \{0,1,2\}} \text{op}_i \Rightarrow \text{serv}_i \right]$$
What $\text{ATL}_{sc}$ can express

- existence of a correct controller for the elevator:
- Client-server interactions for accessing a shared resource:

$$
\langle \cdot \text{Server} \cdot \rangle \ G \ \wedge
\begin{cases}
\bigwedge_{c \in \text{Clients}} \langle \cdot c \cdot \rangle F \text{access}_c \\
\neg \bigwedge_{c \neq c'} \text{access}_c \wedge \text{access}_{c'}
\end{cases}
$$
What $\text{ATL}_{sc}$ can express

- **existence of a correct controller for the elevator:**
- **Client-server interactions** for accessing a shared resource:

$$
\langle \cdot \text{Server} \cdot \rangle \ G \left[ \bigwedge \limits_{c \in \text{Clients}} \langle \cdot c \cdot \rangle F \text{access}_c \wedge \neg \bigwedge \limits_{c \neq c'} \text{access}_c \wedge \text{access}_{c'} \right]
$$

- **Existence of Nash equilibria:**

$$
\langle \cdot A_1, \ldots, A_n \cdot \rangle \bigwedge \limits_i (\langle \cdot A_i \cdot \rangle \varphi_{A_i} \Rightarrow \varphi_{A_i})
$$
What $\text{ATL}_{sc}$ can express

-existence of a correct controller for the elevator:

Client-server interactions for accessing a shared resource:

$$\langle \cdot \text{Server} \cdot \rangle \ G \left[ \bigwedge_{c \in \text{Clients}} \langle \cdot c \cdot \rangle F \text{ access}_c \right]$$

- Existence of Nash equilibria:

$$\langle \cdot A_1, \ldots, A_n \cdot \rangle \bigwedge_i \left( \langle \cdot A_i \cdot \rangle \varphi_{A_i} \Rightarrow \varphi_{A_i} \right)$$

- Existence of dominating strategy:

$$\langle \cdot A \cdot \rangle [B] (\neg \varphi \Rightarrow [A] \neg \varphi)$$
More expressiveness results

Theorem

$ATL_{sc}$ is strictly more expressive than $ATL$. 
Theorem

$ATL_{sc}$ is strictly more expressive than $ATL$.

Proof

- $\langle A \rangle \varphi \equiv (\langle \text{Agt} \rangle \langle A \rangle \hat{\varphi}$
More expressiveness results

Theorem

\( \text{ATL}_{sc} \) is strictly more expressive than ATL.

Proof

- \( \langle \text{A} \rangle \varphi \equiv \langle \text{Agt} \rangle \langle \cdot \text{A} \cdot \rangle \hat{\varphi} \)
- \( \langle 1 \rangle (\langle 2 \rangle \text{X} a \land \langle 2 \rangle \text{X} b) \) is true in \( s' \) and not in \( s \).
More expressiveness results

**Theorem**

\( \text{ATL}_{sc} \) is **strictly more expressive** than \( \text{ATL} \).

**Proof**

- \( \langle A \rangle \varphi \equiv \langle \text{Agt} \rangle \langle A \rangle \hat{\varphi} \)
- \( \langle 1 \rangle (\langle 2 \rangle X a \land \langle 2 \rangle X b) \) is true in \( s' \) and **not** in \( s \).
  But \( s \) and \( s' \) are **alternating bisimilar**.

\[
\begin{array}{c}
\text{s} \\
\langle 1.2 \rangle \\
\langle 1.1 \rangle, \langle 2.2 \rangle \\
\langle 2.1 \rangle \\
a \\
\end{array}
\quad
\begin{array}{c}
\text{s'} \\
\langle 1.2 \rangle, \langle 1.3 \rangle, \langle 3.2 \rangle \\
\langle 2.1 \rangle, \langle 2.3 \rangle, \langle 3.1 \rangle \\
a \\
\end{array}
\]
More expressiveness results

Theorem

- $\text{ATL}_{sc}$ is as expressive as $\text{ATL}^*_sc$;
- $\langle A \rangle$ does not add expressive power.
**Theorem**

- $\text{ATL}_{sc}$ is as expressive as $\text{ATL}^*_{sc}$;
- $\langle A \rangle$ does not add expressive power.

**Proof**

- Make implicit quantification explicit:
  $$\langle A \rangle \varphi \equiv \langle A \rangle [\text{Agt} \setminus A] \hat{\varphi}$$
More expressiveness results

Theorem

- $\text{ATL}_{sc}$ is as expressive as $\text{ATL}^*_{sc}$;
- $\langle A \rangle$ does not add expressive power.

Proof

- Make implicit quantification explicit:

  $$\langle A \rangle \varphi \equiv \langle A \rangle \ [\text{Agt } A] \hat{\varphi}$$

- always assume the context is full;
- keep track of which strategies are really useful.
More expressiveness results

Theorem

- $ATL_{sc}$ is as expressive as $ATL^*_sc$;
- $\langle A \rangle$ does not add expressive power.

Proof

- Make implicit quantification explicit:

  $$\langle A \rangle \varphi \equiv \langle A \rangle [Agt \ \backslash A] \, \widehat{\varphi}$$

  - always assume the context is full;
  - keep track of which strategies are really useful.

- $\langle A \rangle \varphi$ is then equivalent to $[A] \, \widehat{\varphi}$;
More expressiveness results

**Theorem**

- \( ATL_{sc} \) is as expressive as \( ATL^*_{sc} \);
- \(<A>\) does not add expressive power.

**Proof**

- Make implicit quantification explicit:
  
  \[
  <A> \varphi \equiv <A> [\text{Agt} \setminus A] \hat{\varphi}
  \]

- always assume the context is full;
- keep track of which strategies are really useful.

- \(<A>\) \( \varphi \) is then equivalent to \([A] \tilde{\varphi} \);
- for full context, insert \(<\emptyset>\) between temporal modalities:

  \[
  GF \varphi \equiv <\emptyset> G <\emptyset> F \varphi
  \]
Related works – Strategy logic [CHP07, MMV10]

Explicit manipulation of strategies

- **first-order quantification** over strategies;
  \[ \exists \sigma. \varphi(\sigma) \quad \text{there is a strategy } \sigma \text{ such that } \varphi \text{ holds} \]
- **assignment of strategies** to agents.
  \[ \text{assign}(A \mapsto \sigma) \psi \quad \text{if } A \text{ plays } \sigma \text{ then } \varphi \text{ holds} \]


Related works – Strategy logic [CHP07,MMV10]

Explicit manipulation of strategies

- first-order quantification over strategies;
  \[ \exists \sigma. \varphi(\sigma) \] there is a strategy \( \sigma \) such that \( \varphi \) holds

- assignment of strategies to agents.
  \[ \text{assign}(A \mapsto \sigma) \varphi \] if \( A \) plays \( \sigma \) then \( \varphi \) holds

Example

\[ \exists \sigma_1. \text{assign}(A \mapsto \sigma_1) \cdot \forall \sigma_2. \text{assign}(B \mapsto \sigma_2) \cdot F \]

Explicit manipulation of strategies

- first-order quantification over strategies;

\[ \exists \sigma. \varphi(\sigma) \quad \text{there is a strategy } \sigma \text{ such that } \varphi \text{ holds} \]

- assignment of strategies to agents.

\[ \text{assign}(A \mapsto \sigma) \varphi \quad \text{if } A \text{ plays } \sigma \text{ then } \varphi \text{ holds} \]

Example

\[
\begin{align*}
\exists \sigma_1. \text{assign}(A \mapsto \sigma_1). & \quad \forall \sigma_2. \text{assign}(B \mapsto \sigma_2). \\
\langle A \rangle & \quad [B] \\
\end{align*}
\]

\[ F \equiv \langle A \rangle F \]

Related works – Strategy logic \[\text{[CHP07,MMV10]}\]

Explicit manipulation of strategies

- **first-order quantification** over strategies;
  \[
  \exists \sigma. \varphi(\sigma) \quad \text{there is a strategy } \sigma \text{ such that } \varphi \text{ holds}
  \]

- **assignment of strategies** to agents.
  \[
  \text{assign}(A \mapsto \sigma) \varphi \quad \text{if } A \text{ plays } \sigma \text{ then } \varphi \text{ holds}
  \]

**Example**

\[
\exists \sigma_1. \begin{array}{l}
\text{assign}(A \mapsto \sigma_1).
\end{array} \quad \begin{array}{l}
\forall \sigma_2. \text{assign}(B \mapsto \sigma_2).
\end{array}
\frac{\langle A \rangle}{F} \equiv \langle A \rangle F
\]

\[
\exists \sigma_1. A \mathbf{G}(\text{assign}(A \mapsto \sigma_1). \forall \sigma_2. \text{assign}(B \mapsto \sigma_2) F)
\]


Related works – Strategy logic [CHP07,MMV10]

Explicit manipulation of strategies

- first-order quantification over strategies;

\[ \exists \sigma. \varphi(\sigma) \quad \text{there is a strategy } \sigma \text{ such that } \varphi \text{ holds} \]

- assignment of strategies to agents.

\[ \text{assign}(A \mapsto \sigma) \varphi \quad \text{if } A \text{ plays } \sigma \text{ then } \varphi \text{ holds} \]

Proposition

SL is strictly(?) more expressive than ATL_{sc}.

Explicit manipulation of strategies

- first-order quantification over strategies;

\[ \exists \sigma. \varphi(\sigma) \quad \text{there is a strategy } \sigma \text{ such that } \varphi \text{ holds} \]

- assignment of strategies to agents.

\[ \text{assign}(A \rightarrow \sigma) \varphi \quad \text{if } A \text{ plays } \sigma \text{ then } \varphi \text{ holds} \]

Proposition

SL is strictly(?) more expressive than ATL\(_{sc}\).

Theorem

SL model checking is decidable; SL satisfiability is undecidable.


Some other related approaches
Some other related approaches

(Basic) strategy interaction logic [WSH15]

- ATL augmented with strategy interaction quantifiers:

\[ \langle A \rangle (\langle +B \rangle \varphi_B \land \langle +C \rangle \varphi_C) \]

Some other related approaches

(Basic) strategy interaction logic [WSH15]

- ATL augmented with strategy interaction quantifiers:

\[ \langle A \rangle (\langle +B \rangle \varphi_B \land \langle +C \rangle \varphi_C) \]

- context is reset after temporal modalities

## Some other related approaches

### (Basic) strategy interaction logic [WSH15]

- ATL augmented with **strategy interaction quantifiers**:
  
  \[ [A] (\langle +B \rangle \varphi_B \land \langle +C \rangle \varphi_C) \]

  - **context is reset** after temporal modalities
  - **only existential** SIQ

---

Some other related approaches

(Basic) strategy interaction logic [WSH15]

- ATL augmented with strategy interaction quantifiers:

\[
\langle A \rangle (\langle +B \rangle \varphi_B \land \langle +C \rangle \varphi_c)
\]

- context is reset after temporal modalities
- only existential SIQ

Proposition ([WSH15])

Model checking BSIL is PSPACE-complete.

Some other related approaches

(Basic) strategy interaction logic [WSH15]

ATL with irrevocable strategies [ÅGJ07]

\[ \langle \langle A \rangle \rangle \mathbf{G} (\langle \langle A \rangle \rangle \mathbf{X} \quad \bullet) \]

Some other related approaches

(Basic) strategy interaction logic [WSH15]

ATL with irrevocable strategies [ÅGJ07]

• $\langle A \rangle G(\langle A \rangle X \diamond)$

• better use one single strategy per agent;

Some other related approaches

(Basic) strategy interaction logic [WSH15]

ATL with irrevocable strategies [ÅGJ07]

- $\langle \langle A \rangle \rangle \mathbf{G} (\langle \langle A \rangle \rangle \mathbf{X} \mathbf{O})$

- better use one single strategy per agent;

- strategies are persistent.

Some other related approaches

(Basic) strategy interaction logic [WSH15]

ATL with irrevocable strategies [ÅGJ07]

- $\langle\langle A \rangle\rangle \mathbf{G} (\langle\langle A \rangle\rangle \mathbf{X} \bigcirc)$

- better use one single strategy per agent;

- strategies are persistent.

Proposition

Model checking ATL with irrevocable memoryless strategies is PSPACE-complete.

Some other related approaches

(Basic) strategy interaction logic [WSH15]

ATL with irrevocable strategies [ÅGJ07]

ATL with explicit strategies [WHW07]

\[ \langle A \rangle_{\rho} \varphi: \rho \text{ explicitly imposes a strategy for some players}; \]

Some other related approaches

(Basic) strategy interaction logic [WSH15]

ATL with irrevocable strategies [ÅGJ07]

ATL with explicit strategies [WHW07]

\[ \langle A \rangle_\rho \varphi : \rho \text{ explicitly imposes a strategy for some players}; \]

Proposition

For memoryless explicit strategy contexts, model checking ATLES is PTIME-complete.

Some other related approaches

(Basic) strategy interaction logic [WSH15]

ATL with irrevocable strategies [ÅGJ07]

ATL with explicit strategies [WHW07]

Stochastic Game Logic [BBGK07]
- extension of $\text{ATL}_{sc}$ to stochastic setting;

Some other related approaches

- (Basic) strategy interaction logic [WSH15]
- ATL with irrevocable strategies [ÅGJ07]
- ATL with explicit strategies [WHW07]

**Stochastic Game Logic [BBGK07]**

- extension of $\text{ATL}_{sc}$ to stochastic setting;
- quantitative objectives make model checking undecidable;

Some other related approaches

(Basic) strategy interaction logic [WSH15]

ATL with irrevocable strategies [ÅGJ07]

ATL with explicit strategies [WHW07]

Stochastic Game Logic [BBGK07]

- extension of $\text{ATL}_{sc}$ to stochastic setting;
- quantitative objectives make model checking undecidable;

Proposition ([BBGK07])

Model checking Stochastic Game Logic with memoryless randomized strategies is decidable in EXPSPACE.

Decision procedures for ATL$_{sc}$

ESSLLI 2015

F. Laroussinie      N. Markey
Decision procedures for ATL$_{sc}$

Verification problems:

Satisfiability:
input: a formula $\phi$
output: yes iff there exists a model satisfying $\phi$. (+ a model if the answer is yes…)

Model-checking:
input: a model $S$ and a formula $\phi$
output: yes iff $S$ is a model for $\phi$.

+variants: turn-based, memoryless strategies, …
Model-checking algo. for ATL

\[ \phi = \langle a_1 \rangle (P_1 U P_2) \]

In which states of G, is \( \phi \) satisfied ?

→ standard fixpoint computation (as for CTL)…
complexity in \( O(|\phi|.|G|) \)
Model-checking algo. for ATL

\( \phi = \langle a_1 \rangle (P_1 U P_2) \)

In which states of G, is \( \phi \) satisfied?

\[ \text{→ standard fixpoint computation (as for CTL)… complexity in } O(|\phi|.|G|) \]
Model-checking algo. for ATL

\[ \phi = \langle a_1 \rangle (P_1 U P_2) \]

In which states of G, is \( \phi \) satisfied?

→ standard fixpoint computation (as for CTL)…
complexity in \( O(|\phi|.|G|) \)
Model-checking algo. for ATL

\[ \phi = \langle a_1 \rangle (P_1 \cup P_2) \]

\( G : \)

In which states of \( G \), is \( \phi \) satisfied?

\[ \rightarrow \text{standard fixpoint computation (as for CTL)} \ldots \]

complexity in \( O(|\phi|.|G|) \)
Model-checking algo. for ATL

\[ \phi = \langle a_1 \rangle (P_1 U P_2) \]

In which states of G, is \( \phi \) satisfied?

\[ \rightarrow \text{standard fixpoint computation (as for CTL)...} \]
\[ \text{complexity in } O(|\phi|.|G|) \]
Model-checking algo. for ATL

\[ \phi = \langle a_1 \rangle (P_1 U P_2) \]

In which states of G, is \( \phi \) satisfied?

→ standard fixpoint computation (as for CTL)…
complexity in \( O(|\phi|.|G|) \)
Decision procedures for \(\text{ATL}_{\text{sc}}\)

It is more difficult for \(\text{ATL}_{\text{sc}}\) :
\(\text{ATL}_{\text{sc}}\) formulas are interpreted over a state within a strategy context!

\[
\langle \cdot a_0 \cdot \rangle G (\langle \cdot a_1 \cdot \rangle F P_1 \land \langle \cdot a_2 \cdot \rangle F P_2)
\]

\(\implies\) the strategy for \(a_1\) depends on the strategy for \(a_0\)!

\textbf{ATL:}
\[
G,q \models \langle A \rangle \phi_p \iff \exists f_A \in \text{Strat}(A). \forall \pi \in \text{Outcome}(q,f_A). \pi \models \phi_p
\]

\textbf{ATL}_{\text{sc}}:\n\[
G,q \models_F \langle A \rangle \phi_p \iff \exists f_A \in \text{Strat}(A). \forall \pi \in \text{Outcome}(q,f_{A_0}F). \pi \models \phi_p
\]
Example of ATL\textsubscript{sc} model checking

$\mathcal{M} = \{1,2\}, \text{Agt}=\{a_0,a_1,a_2\}$

ATL: $\psi = \langle\langle a_0 \rangle\rangle G (\langle\langle a_1 \rangle\rangle F P_1 \land \langle\langle a_2 \rangle\rangle F P_2)$

$G, s_0 \models \psi$ ?

No ! $\langle\langle a_1 \rangle\rangle F P_1$ is only true at $s_4$ !

$\langle\langle a_2 \rangle\rangle F P_2$ is only true at $s_5$ !

ATL\textsubscript{sc}: $\phi = \langle \cdot a_0 \cdot \rangle G (\langle \cdot a_1 \cdot \rangle F P_1 \land \langle \cdot a_2 \cdot \rangle F P_2)$

$G, s_0 \models \phi$ ?
Example of ATL\textsubscript{sc} model checking

\[ \phi = \langle \cdot a_0 \cdot \rangle G (\langle \cdot a_1 \cdot \rangle F P_1 \land \langle \cdot a_2 \cdot \rangle F P_2) \]

\[ M=\{1,2\}, \text{Agt}=\{a_0, a_1, a_2\} \]

Assume \( a_0 \) plays “1” everywhere, \( G \) becomes:

Assume \( a_0 \) …, and \( a_1 \) starts to play “1”, \( G \) becomes:

Assume \( a_0 \) …, and \( a_2 \) starts to play “1”, \( G \) becomes:
Example of $\text{ATL}_{sc}$ model checking

$\phi = \langle \cdot a_0 \cdot \rangle G (\langle \cdot a_1 \cdot \rangle F P_1 \land \langle \cdot a_2 \cdot \rangle F P_2)$

$M = \{1, 2\}$, $\text{Agt} = \{a_0, a_1, a_2\}$

Assume $a_0$ plays “1” everywhere, $G$ becomes:

Assume $a_0 \ldots$, and $a_1$ starts to play “1”, $G$ becomes:

Assume $a_0 \ldots$, and $a_2$ starts to play “1”, $G$ becomes:

Thus, we have $G, s_0 \models \phi$
Decision procedures for $\text{ATL}_{sc}$

→ We use tree automata techniques to define the algorithms for $\text{ATL}_{sc}$.

More precisely, we

- define tree automata constructions for $\text{QCTL}$ (quantified CTL)
- and reduce $\text{ATL}_{sc}$ problems to $\text{QCTL}$ problems.

\[
\begin{align*}
\text{CGS} & \quad G \quad \rightarrow \quad \text{Reduction} \quad S_G \quad \text{a Kripke structure} \\
\text{ATL}_{sc} & \quad \phi \quad \rightarrow \quad \text{model-checking} \quad \psi_{G,\phi} \quad \text{a QCTL* formula}
\end{align*}
\]
Decision procedures for $\text{ATL}_{\text{sc}}$

→ We use tree automata techniques to define the algorithms for $\text{ATL}_{\text{sc}}$.

More precisely, we
- define tree automata constructions for $\text{QCTL}$ (quantified CTL)
- and reduce $\text{ATL}_{\text{sc}}$ problems to QCTL problems.

\[
\begin{align*}
\text{CGS} & \quad G \quad \rightarrow \quad \text{Reduction} \quad \rightarrow \quad S_G \text{ a Kripke structure} \\
\text{ATL}_{\text{sc}} & \quad \phi \quad \rightarrow \quad \text{model-checking} \quad \rightarrow \quad \psi_{G,\phi} \text{ a QCTL* formula}
\end{align*}
\]

such that $G \models \phi \iff S_G \models \psi_{G,\phi}$
Outlines

I- Quantified CTL
   - syntax and semantics, examples,…
   - expressivity
   - decision procedures for QCTL
     ➡ [ Tree automata techniques for QCTL ]

II- Application to ATLsc
   - ATLsc model-checking
     [reduction to QCTL model-checking]
   - ATLsc satisfiability
     [undecidable for the general case, reduction to QCTL satisf. for subcases]
I- Quantified CTL

- syntax and semantics, examples,…
- expressivity
- decision procedures for QCTL
  ➡ [ Tree automata techniques for QCTL ]
QCTL / QCTL*

Syntax:
State formulas:
\( \varphi, \psi ::= \text{P} \mid \lnot \varphi \mid \varphi \lor \psi \mid \text{E} \varphi_p \mid \text{A} \varphi_p \mid \exists P. \varphi \)

Path formulas:
QCTL*: \( \varphi_p, \psi_p ::= \varphi \mid \lnot \varphi_p \mid \varphi_p \lor \psi_p \mid \text{X} \varphi_p \mid \varphi_p \text{U} \psi_p \)
QCTL: \( \varphi_p ::= \text{X} \varphi \mid \varphi \text{U} \psi \)

+Abbrev.: \( \forall P. \varphi = \lnot \exists P. \lnot \varphi \)

Semantics: \( S, q \models \exists P. \varphi \)

idea: «there exists a P-labeling of the model s.t. \( \varphi \) holds at q»
Semantics

\[ \exists P. \varphi \text{ can be interpreted in different manners:} \]

- \( P \) labels the Kripke structure \( S = (Q,R,\ell) \), or
- \( P \) labels the execution tree \( T_S \).
-...

\text{structure sem.} \quad \text{tree sem.}
structure vs tree semantics

\[
\phi = \exists p. (p \land \neg \text{EX} p)
\]

\(\vdash q_1 \phi\)?

\(q_1 \not\models s \phi\)
\(q_1 \models t \phi\)

\(\phi\) is false with the structure semantics
\(\phi\) is true with the tree semantics

notation:
\(q_1 \not\models s \phi\)
\(q_1 \models t \phi\)
Formally...

- Kripke structure: $S = (Q, R, \ell)$:
  
  $[R \subseteq Q \times Q$ is a total relation, $\ell : Q \to 2^{AP}]$

- Given $P \subseteq AP$, $S = (Q, R, \ell)$ and $S' = (Q', R', \ell')$ are
  
  P-equivalent $(S \equiv_P S')$ iff $Q = Q'$, $R = R'$, $\ell \cap P = \ell' \cap P$

  **structure semantics:**
  
  $S, q \models_s \exists p. \phi$ iff $\exists S' \equiv_{AP \setminus \{p\}} S$ s.t. $S', q \models \phi$

- Let $T_S(q)$ be the unwinding of a finite KS $S$ from $q$.

  **tree semantics:**
  
  $S, q \models_t \phi$ iff $T_S(q), q \models_s \phi$
Expressiveness overview

Proposition: In both semantics, **EQCTL** and **QCTL** are equally expressive.

Proposition: In both semantics, **QCTL** and **MSO** are equally expressive.

Proposition: In both semantics, **QCTL** and **QCTL*** are equally expressive.

in Prenex norm. form
Examples

selfloop = ∀ z. (z ⇒ EX z)

q ⊨ selfloop ⇔ «q carries a self-loop»
selfloop ≡_{tree} ⊥ (for the tree semantics)

uniq(φ) = EF(φ) ∧ ∀ z. (EF (φ ∧ z) ⇒ AG(φ ⇒ z))

q ⊨ uniq(φ) ⇔ «there is exactly one reachable state satisfying φ»
q ⊨t uniq(φ) ⇔ «there is exactly one φ-node in the tree»

EX_{1}(φ) = EX φ ∧ ∀ z. (EX (φ ∧ z) ⇒ AX (φ ⇒ z))
EX_{2}(φ) = ...
Tree automata for QCTL

(From now on, we only consider the tree semantics)
Parity tree automata over binary trees

Infinite binary trees:

\[ \mathcal{T} = (T, \ell) \] with \( T = \{0,1\}^* \) and \( \ell : T \rightarrow \Sigma \)
(non-deterministic) **Parity** tree automata over **binary** trees

\[ \mathcal{A} = (Q, q_0, \Sigma, \tau, \Omega) \]

- \( Q \) is a set of states, \( q_0 \in Q \),
- \( \Sigma \) is an alphabet,
- \( \tau \) is the “transition function”,
- \( \Omega \) is the (parity) acceptance condition: \( \Omega : Q \rightarrow \{0, \ldots, k-1\} \).

\( \tau(q, \sigma) \) is a set of pairs \((q_i^0, q_i^1)\) describing different cases for the recognition of the two successors (0-succ. and 1-succ) of the root when it is labeled by \( \sigma \) and when the automaton state is \( q \).

Alternatively we write \( \tau(q, \sigma) \) as a Boolean formula:

\[
\tau(q, \sigma) = \bigvee_i (0, q_i^0) \land (1, q_i^1)
\]

and then, we have \( \tau : Q \times \Sigma \rightarrow \mathcal{B}([0,1] \times Q) \)
(non-deterministic) **Parity** tree automata over **binary** trees

\[ \mathcal{A} = (Q, q_0, \Sigma, \tau, \Omega) \]

- \( Q \) is a set of states, \( q_0 \in Q \),
- \( \Sigma \) is an alphabet,
- \( \tau \) is the “transition function”,
- \( \Omega \) is the (parity) acceptance condition: \( \Omega: Q \rightarrow \{0, \ldots, k-1\} \).

A \( \Sigma \)-labeled tree \( T = (T, \ell) \) is **accepted** by \( \mathcal{A} \) iff there exists an **accepting execution**, i.e. a \( Q \) labeling of \( T \) such that:

(1) every node \( r \) labeled by \( q \) has two sons \( r.0 \) and \( r.1 \) satisfying \( \tau(q, \ell(r)) \), and

(2) every branch \( \pi \) satisfies the parity condition:

\[ \min \{ \Omega(q) \mid q \text{ is repeated infinitely often along } \pi \} \text{ is even.} \]
Parity tree automata over binary trees

Example + a labeling of atomic prop. \( a, b \)

\[ \mathcal{A} = (Q, q_0, \Sigma, \tau, \Omega) \]
\[ \Sigma = \{\emptyset, \{a\}, \{b\}, \{a,b\}\} \]
\[ Q = \{q, q_{acc}, q_{rej}\} \]
\[ \Omega(q_{acc}) = 0 \]
\[ \Omega(q_{rej}) = \Omega(q) = 1 \]

\[
\tau(q, \sigma) = \begin{cases} 
(0, q_{acc}) \land (1, q_{acc}) & \text{if } b \in \sigma \\
(0, q) \land (1, q_{acc}) \lor (0, q_{acc}) \land (1, q) & \text{if } a \in \sigma \text{ (and } b \notin \sigma) \\
(0, q_{rej}) \land (1, q_{rej}) & \text{otherwise}
\end{cases}
\]

\[
\tau(q_{acc}, \sigma) = (0, q_{acc}) \land (1, q_{acc}) \\
\tau(q_{rej}, \sigma) = (0, q_{rej}) \land (1, q_{rej})
\]

\[ \forall \sigma \in \Sigma \]
Parity tree automata over binary trees

Example

+ a labeling of atomic prop. $a, b$

$\mathcal{A} = (Q, q, \Sigma, \tau, \Omega)$

$\Sigma = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$

$Q = \{q, q_{acc}, q_{rej}\}$

$\Omega(q_{acc}) = 0$

$\Omega(q_{rej}) = \Omega(q) = 1$

$\tau(q, \sigma) = \begin{cases} 
(0, q_{acc}) \land (1, q_{acc}) & \text{if } b \in \sigma \\
(0, q) \land (1, q_{acc}) \lor (0, q_{acc}) \land (1, q) & \text{if } a \in \sigma \text{ (and } b \notin \sigma) \\
(0, q_{rej}) \land (1, q_{rej}) & \text{otherwise}
\end{cases}$

$\tau(q_{acc}, \sigma) = (0, q_{acc}) \land (1, q_{acc})$

$\tau(q_{rej}, \sigma) = (0, q_{rej}) \land (1, q_{rej})$

$\forall \sigma \in \Sigma$

$\forall \sigma \in \Sigma$
Parity tree automata over binary trees

+ a labeling of atomic prop. $a, b$

$$\mathcal{A} = (Q, q, \Sigma, \tau, \Omega)$$

$$\Sigma = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$$

$$Q = \{q, q_{\text{acc}}, q_{\text{rej}}\}$$

$$\Omega(q_{\text{acc}}) = 0$$

$$\Omega(q_{\text{rej}}) = \Omega(q) = 1$$

$$\tau(q, \sigma) = \begin{cases} (0, q_{\text{acc}}) \land (1, q_{\text{acc}}) & \text{if } b \in \sigma \\ (0, q) \land (1, q_{\text{acc}}) \lor (0, q_{\text{acc}}) \land (1, q) & \text{if } a \in \sigma \text{ (and } b \notin \sigma) \\ (0, q_{\text{rej}}) \land (1, q_{\text{rej}}) & \text{otherwise} \end{cases}$$

$$\tau(q_{\text{acc}}, \sigma) = (0, q_{\text{acc}}) \land (1, q_{\text{acc}})$$

$$\forall \sigma \in \Sigma$$

$$\tau(q_{\text{rej}}, \sigma) = (0, q_{\text{rej}}) \land (1, q_{\text{rej}})$$

$$\forall \sigma \in \Sigma$$
Parity tree automata over binary trees

Example + a labeling of atomic prop. \(a, b\)

\[\mathcal{A} = (Q, q, \Sigma, \tau, \Omega)\]
\[\Sigma = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}\]
\[Q = \{q, q_{\text{acc}}, q_{\text{rej}}\}\]
\[\Omega(q_{\text{acc}}) = 0\]
\[\Omega(q_{\text{rej}}) = \Omega(q) = 1\]

\[\tau(q, \sigma) = \begin{cases} 
(0, q_{\text{acc}}) \land (1, q_{\text{acc}}) & \text{if } b \in \sigma \\
(0, q) \land (1, q_{\text{acc}}) \lor (0, q_{\text{acc}}) \land (1, q) & \text{if } a \in \sigma \text{ (and } b \notin \sigma) \\
(0, q_{\text{rej}}) \land (1, q_{\text{rej}}) & \text{otherwise} 
\end{cases}\]

\[\tau(q_{\text{acc}}, \sigma) = (0, q_{\text{acc}}) \land (1, q_{\text{acc}})\]
\[\tau(q_{\text{rej}}, \sigma) = (0, q_{\text{rej}}) \land (1, q_{\text{rej}})\]

\[\forall \sigma \in \Sigma\]
Parity tree automata over binary trees

Example + a labeling of atomic prop. $a,b$

$$\mathcal{A} = (Q,q,\Sigma,\tau,\Omega)$$
$$\Sigma = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$$
$$Q = \{q_{\text{acc}}, q_{\text{rej}}\}$$
$$\Omega(q_{\text{acc}}) = 0$$
$$\Omega(q_{\text{rej}}) = \Omega(q) = 1$$

$$\tau(q,\sigma) = \begin{cases} 
(0,q_{\text{acc}}) \land (1,q_{\text{acc}}) & \text{if } b \in \sigma \\
(0,q) \land (1,q_{\text{acc}}) \lor (0,q_{\text{acc}}) \land (1,q) & \text{if } a \in \sigma \text{ (and } b \notin \sigma) \\
(0,q_{\text{rej}}) \land (1,q_{\text{rej}}) & \text{otherwise}
\end{cases}$$

$$\tau(q_{\text{acc}},\sigma) = (0,q_{\text{acc}}) \land (1,q_{\text{acc}})$$
$$\tau(q_{\text{rej}},\sigma) = (0,q_{\text{rej}}) \land (1,q_{\text{rej}})$$

$$\forall \sigma \in \Sigma \quad \forall \sigma \in \Sigma$$
Parity tree automata over binary trees

Example

+ a labeling of atomic prop. \( a, b \)

\[ A = (Q, q, \Sigma, \tau, \Omega) \]
\[ \Sigma = \{ \emptyset, \{a\}, \{b\}, \{a, b\} \} \]
\[ Q = \{ q, q_{\text{acc}}, q_{\text{rej}} \} \]
\[ \Omega(q_{\text{acc}}) = 0 \]
\[ \Omega(q_{\text{rej}}) = \Omega(q) = 1 \]

\[ \tau(q, \sigma) = \begin{cases} 
(0, q_{\text{acc}}) \land (1, q_{\text{acc}}) & \text{if } b \in \sigma \\
(0, q) \land (1, q_{\text{acc}}) \lor (0, q_{\text{acc}}) \land (1, q) & \text{if } a \in \sigma \text{ (and } b \notin \sigma) \\
(0, q_{\text{rej}}) \land (1, q_{\text{rej}}) & \text{otherwise} 
\end{cases} \]

\[ \forall \sigma \in \Sigma \]

\[ \tau(q_{\text{acc}}, \sigma) = (0, q_{\text{acc}}) \land (1, q_{\text{acc}}) \]
\[ \forall \sigma \in \Sigma \]

\[ \tau(q_{\text{rej}}, \sigma) = (0, q_{\text{rej}}) \land (1, q_{\text{rej}}) \]
\[ \forall \sigma \in \Sigma \]
Parity tree automata over binary trees

Example

With a labeling of atomic prop. \(a,b\)

\[ \mathcal{A} = (Q,q,\Sigma,\tau,\Omega) \]

\[ \Sigma = \{\emptyset, \{a\}, \{b\}, \{a,b\}\} \]

\[ Q = \{q, q_{\text{acc}}, q_{\text{rej}}\} \]

\[ \Omega(q_{\text{acc}}) = 0 \]

\[ \Omega(q_{\text{rej}}) = \Omega(q) = 1 \]

\( \mathcal{A} \) accepts the binary trees satisfying the CTL property \(EaUb\)!
Alternating Parity tree automata over binary trees

\( \mathcal{A} \) accepts the binary trees satisfying the CTL property \( E_aU_b \).

\( \mathcal{A} = (\{q\}, q, \Sigma, \tau, \Omega) \) with \( \Omega(q) = 1 \) and:

\[
\tau(q, \sigma) = \begin{cases} 
T & \text{if } b \in \sigma \\
(0, q) \lor (1, q) & \text{if } a \in \sigma \text{ (and } b \notin \sigma) \\
\bot & \text{otherwise}
\end{cases}
\]
Alternating Parity tree automata over binary trees

\( \mathcal{A} \) accepts the binary trees satisfying the CTL property \( E_a U b \).

\( \mathcal{A} = (\{q\}, q, \Sigma, \tau, \Omega) \) with \( \Omega(q) = 1 \) and:

\[
\tau(q, \sigma) = \begin{cases} 
T & \text{if } b \in \sigma \\
(0, q) \lor (1, q) & \text{if } a \in \sigma \text{ (and } b \notin \sigma) \\
\bot & \text{otherwise}
\end{cases}
\]
**Alternating Parity tree automata over binary trees**

$\mathcal{A}$ accepts the binary trees satisfying the CTL property $EaUb$.

$\mathcal{A} = (\{q\}, q, \Sigma, \tau, \Omega)$ with $\Omega(q) = 1$ and:

$$\tau(q, \sigma) = \begin{cases} 
T & \text{if } b \in \sigma \\
(0, q) \lor (1, q) & \text{if } a \in \sigma \text{ (and } b \notin \sigma) \\
\bot & \text{otherwise}
\end{cases}$$

$(0, q) \land (1, q_{acc}) \lor (0, q_{acc}) \land (1, q)$

$\tau(q, \sigma)$ is a positive Boolean formula over terms in $\{0, 1\} \times Q$
Consider the CTL property $E_a U b \land E_c U d$:

$$\tau(q_0,\sigma) = \tau(q_1,\sigma) \land \tau(q_2,\sigma)$$

\[
\begin{align*}
\tau(q_1,\sigma) &= \begin{cases} 
\top & \text{if } b \in \sigma \\
(0,q_1) \lor (1,q_1) & \text{if } \sigma = \{a\} \\
\bot & \text{otherwise}
\end{cases} \\
\tau(q_2,\sigma) &= \begin{cases} 
\top & \text{if } d \in \sigma \\
(0,q_2) \lor (1,q_2) & \text{if } \sigma = \{c\} \\
\bot & \text{otherwise}
\end{cases}
\end{align*}
\]

$A = (\{q_0,q_1,q_2\},q_0,\Sigma,\tau,\Omega)$ with $\Omega(q_1) = \Omega(q_1) = 0$.
a tree $T=(T,\ell)$

an APT

$\mathcal{A} = (Q,q_0,\Sigma,\tau,\Omega)$

s.t. $(0,q_1),(0,q_2),(1,q_1) \vDash \tau(q,\sigma)$

ex: $(0,q_1) \land (1,q_1) \land (0,q_2) \lor (1,q_2)$

an execution of $\mathcal{A}$ over $T$?
Let $\mathcal{D} \subseteq \mathbb{N}$ be a finite set of degrees.

In the following, we consider alternating parity tree automata for $\mathcal{D}$-trees labeled by propositions in $\text{AP} \ (\Sigma = 2^{\text{AP}})$

$$\tau(q,\sigma) = \{ \tau_{d_1}(q,\sigma), \tau_{d_2}(q,\sigma), \tau_{d_3}(q,\sigma), \ldots \} \text{ when } \mathcal{D} = \{d_1,\ldots,d_k\}$$

$\rightarrow \langle \mathcal{D}, 2^{\text{AP}} \rangle$-Alternating Parity Trees automaton $= \langle \mathcal{D}, 2^{\text{AP}} \rangle$-APT.

A **Non-deterministic** Parity Tree automaton has a transition function $\tau$ for $d$-nodes as follows:

$$\tau(q,\sigma) = \bigvee_i \left( \bigwedge_{0 \leq c < d} (c, q_i, c) \right)$$
### Properties:

- Boolean operations are easy to handle!
  
  $\mathcal{A}_1 \cup \mathcal{A}_2, \quad \mathcal{A}_1 \cap \mathcal{A}_2, \quad \neg \mathcal{A}$

\[ \begin{align*}
\mathcal{A}_1 \cup \mathcal{A}_2 & : \ (Q^1 \cup Q^2,q_0,\Sigma,\tau,\Omega^1 \cup \Omega^2) \\
\tau(q_0,\sigma) & = \tau(q_0^1,\sigma) \lor \tau(q_0^2,\sigma)
\end{align*} \]

\[ \begin{align*}
\mathcal{A}_1 \cap \mathcal{A}_2 & : \ (Q^1 \cap Q^2,q_0,\Sigma,\tau,\Omega^1 \cup \Omega^2) \\
\tau(q_0,\sigma) & = \tau(q_0^1,\sigma) \land \tau(q_0^2,\sigma)
\end{align*} \]

**Example:** For the CTL property $EaUb \land EcUd$, we had:

\[ \begin{align*}
\mathcal{A} & = (\{q_0,q_1,q_2\},q_0,\Sigma,\tau,\Omega) \text{ with } \tau(q_0,\sigma) = \tau(q_1,\sigma) \land \tau(q_2,\sigma) \\
\mathcal{A}_1 & = (Q^1,q_0^1,\Sigma,\tau^1,\Omega^1) \\
\mathcal{A}_2 & = (Q^2,q_0^2,\Sigma,\tau^2,\Omega^2)
\end{align*} \]

\[ \begin{align*}
\tau(q_1,\sigma) & = \begin{cases} 
\top & \text{if } b \in \sigma \\
(0,q_1) \lor (1,q_1) & \text{if } \sigma = \{a\} \\
\bot & \text{otherwise}
\end{cases} \\
\tau(q_2,\sigma) & = \begin{cases} 
\top & \text{if } d \in \sigma \\
(0,q_2) \lor (1,q_2) & \text{if } \sigma = \{c\} \\
\bot & \text{otherwise}
\end{cases}
\end{align*} \]

$\mathcal{A}_1 \rightarrow EaUb$  \quad  $\mathcal{A}_2 \rightarrow EcUd$
Properties:

- Boolean operations are easy to handle!
- $A_1 \cup A_2$, $A_1 \cap A_2$, $\neg A$

$A = (Q, q_0, \Sigma, \tau, \Omega)$

$\neg A$: $(Q, q_0, \Sigma, \overline{\tau}, \Omega')$

$\overline{\tau}(q, \sigma) = \text{dual of } \tau(q, \sigma)$:

- $\overline{T} = \bot$, $\overline{\bot} = T$, $\theta \lor \theta' = \overline{\theta \land \theta'}$, $\overline{\theta \land \theta'} = \theta \lor \theta'$

(and no change for $(c,q)$)

And $\Omega'(q) = \Omega(q) + 1$ (to change the parity)
Example for $\neg A$:

$A = (\{q\}, q, \Sigma, \tau, \Omega)$

$\Omega(q) = 1$

$\neg A = (\{q\}, q, \Sigma, \neg \tau, \Omega')$

$\Omega'(q) = 2$

Trees in $L(A)$:

$\varepsilon, \{\} \quad \{a\} \quad \{a\}$

$
\{\} \quad \{\} \quad \{\} \quad \{\} 
\{\} \quad \{\} \quad \{\} \quad \{\}
$

$\rightarrow$ trees satisfying

$A (a \land \neg b) W (\neg a \land \neg b)$

$\equiv A (\neg b) W (\neg a \land \neg b)$

$\equiv \neg E a U b$ !!!
Theorem:

• Checking the emptiness of $\mathcal{A}$ is EXPTIME-C. [Löding 13]

• If $\mathcal{A}$ accepts some infinite tree, then it accepts a regular one. [Rabin 72]

• The problem whether an APT $\mathcal{A}$ with $d$ priorities accepts a regular tree $T_S$ can be solved in time $O((|\mathcal{A}|.|S|)^d)$. [Löding 13]

How to check $T_S \in L(\mathcal{A})$?
From $T_s$ and $\mathcal{A}$, one builds a parity game $G_{\mathcal{A},T_s}$ for two players $E$ and $A$: $E$ has a winning strategy in $G_{\mathcal{A},T_s}$ iff $T_S \in L(\mathcal{A})$ (see below)
Theorem: [Kupferman, Vardi, Wolper 00]
Given a CTL formula \( \phi \) over AP and set \( D \), we can build a \( \langle D, 2^{AP}\rangle \)-APT \( A_\phi \) accepting the \( 2^{AP} \)-labeled \( D \)-trees satisfying \( \phi \).
\( |A_\phi| \) is linear in \( |\phi| \), the nb of priorities is constant.

W.l.o.g. assume the negations in \( \phi \) only deal with atomic prop.
\( A_\phi = (Q_\phi, \phi, \Sigma, \tau, \Omega) \) with:
- \( Q_\phi \) is the set of \( \phi \) subformulas,
- for \( d \in D, \psi \in Q_\phi, \sigma \in 2^{AP} \), we define \( \sigma(\psi, \sigma) \) inductively:

\[
\tau_d(P, \sigma) = \begin{cases} 
\top & \text{if } P \in \sigma \\
\bot & \text{otherwise}
\end{cases}
\]

\[
\tau_d(\psi_1 \land \psi_2, \sigma) = \tau_d(\psi_1, \sigma) \land \tau_d(\psi_2, \sigma)
\]

\[
\tau_d(\psi_1 \lor \psi_2, \sigma) = \tau_d(\psi_1, \sigma) \lor \tau_d(\psi_2, \sigma)
\]

\[
\tau_d(\text{EX} \psi, \sigma) = \bigvee_{0 \leq c < d} (c, \psi)
\]

\[
\tau_d(\text{E} \psi_1 \text{U} \psi_2, \sigma) = \tau_d(\psi_2, \sigma) \lor \left( \tau_d(\psi_1, \sigma) \land \bigvee_{0 \leq c < d} (c, \text{E} \psi_1 \text{U} \psi_2) \right)
\]

\[
\tau_d(\text{A} \psi_1 \text{U} \psi_2, \sigma) = \tau_d(\psi_2, \sigma) \lor \left( \tau_d(\psi_1, \sigma) \land \bigwedge_{0 \leq c < d} (c, \text{A} \psi_1 \text{U} \psi_2) \right)
\]

\[
\Omega(\text{E} \psi_1 \text{U} \psi_2) = \Omega(\text{A} \psi_1 \text{U} \psi_2) = 1 \quad \text{and} \quad \Omega(\text{E} \psi_1 \text{W} \psi_2) = \Omega(\text{A} \psi_1 \text{W} \psi_2) = 2
\]
APT and CTL model-checking

\( \mathcal{A} \) accepts the binary trees satisfying the CTL property \( E_a U_b \).

\( \mathcal{A} = (\{q\},q,\Sigma,\tau,\Omega) \) with \( \Omega(q)=1 \) and:

\[
\tau(q,\sigma) = \begin{cases} 
T & \text{if } b \in \sigma \\
(0,q)\vee(1,q) & \text{if } a \in \sigma \text{ (and } b \notin \sigma) \\
\perp & \text{otherwise}
\end{cases}
\]

Previous def.:

\[
\tau_2(E \psi_1 U \psi_2,\sigma) = \tau_2(\psi_2,\sigma) \vee (\tau_2(\psi_1,\sigma) \wedge (0,E \psi_1 U \psi_2) \vee (1,E \psi_1 U \psi_2))
\]

\[
\tau_2(E a U b,\sigma) = \tau_2(b,\sigma) \vee (\tau_2(a,\sigma) \wedge (0,E a U b) \vee (1,E a U b))
\]
**Application**

APT and CTL model-checking

**input:** $S$ a Kripke structure + $\phi$ a CTL formula

$S$ gives a set of degrees $D$.
The execution tree of $S$ is $2^{AP}$-labeled $D$-tree $T_S$.

And: $S \models \phi$ iff $T_S \in L(A_\phi)$

build $A_\phi$ for $D$ + decision procedure for APT

$\rightarrow$ algorithm in $O((|\phi|.|S|)^2)$

(and in $O(|\phi|.|S|)$ if we proceed carefully !)
APT and CTL model-checking

Application

$\mathcal{A}_\phi = (Q,q_0,\Sigma,\tau,\Omega)$ and $\mathcal{S} = (S,R_s,\ell)$

How can we check $T_s \in L(\mathcal{A}_\phi)$?

From $\mathcal{A}$ and $\mathcal{S}$, one builds a finite Parity Game $G_{s,\phi} = \langle V_E, V_A, R, c \rangle$ s.t.

$$T_s \in L(\mathcal{A}_\phi) \text{ iff Eve has a winning strategy in } G_{s,\phi}$$

**Definition “parity game”:** $G = \langle V_E, V_A, R, c \rangle$ $V = V_E \cup V_A$, $R \subseteq V \times V$, $c: V \rightarrow \mathbb{N}$

two players game: **Eve** plays (ie chooses a transition in $R$) from $V_E$, **Adam** plays from $V_A$ states.

A play in $G$ is:

- a *finite* sequence of states ending in a state without succ.
- an *infinite* sequence of states.

Eve wins iff Adam cannot move or if the minimal priority that appears infinitely often is even.
Application

**APT and CTL model-checking**

\[ \mathcal{A}_\phi = (Q,q_0,\Sigma,\tau,\Omega) \quad \text{and} \quad S = (S,R_s,\ell) \]

How can we check \( T_S \in L(\mathcal{A}_\phi) ? \)

From \( \mathcal{A} \) and \( S \), one builds a finite Parity Game \( G_{S,\phi} = \langle V_E, V_A, R, c \rangle \) s.t.

\( T_S \in L(\mathcal{A}_\phi) \) iff Eve has a winning strategy in \( G_{S,\phi} \)

---

**Def of \( G_{S,\phi} : \)**

\[ V = (S \times Q) \cup \{ (s,\theta) \mid s \in S \text{ and } \theta \text{ is a subform. of some } \tau_{d(s)}(q,\sigma) \} \]

\[ (s,q) \rightarrow (s,\tau_{d(s)}(q,\ell(s)) \quad (s,\theta_1 \land \theta_2) \rightarrow (s,\theta_1) \quad (s,\theta_1 \land \theta_2) \rightarrow (s,\theta_2) \]

\[ (s,(c,q')) \rightarrow (s',q') \text{ if } s \rightarrow_c s' \quad (s,\theta_1 \lor \theta_2) \rightarrow (s,\theta_1) \quad (s,\theta_1 \lor \theta_2) \rightarrow (s,\theta_2) \]

\[ V_E = (s,q) \text{ states} + (s,\bot) + (s,\theta_1 \lor \theta_2) \quad VA = V \setminus V_E \]

Eve plays for disjunctions… and for \( (s,q) \) and \( (s,\bot) \).
Example

APT and CTL model-checking

\[ \phi = E(\text{EX} P_1) \cup P_2 \]

\[ \mathcal{D} = \{1, 2\} \]

\[ \tau_2(\phi, \sigma) = \begin{cases} 
T & \text{if } P_2 \in \sigma \\
((0, P_1) \lor (1, P_1)) \land ((0, \phi) \lor (1, \phi)) & \text{otherwise}
\end{cases} \]

\[ \tau_1(\phi, \sigma) = \begin{cases} 
T & \text{if } P_2 \in \sigma \\
(0, P_1) \land (0, \phi) & \text{otherwise}
\end{cases} \]
APT and CTL model-checking

Example

\[ \phi = E (EX P_1) U P_2 \]

\[ \mathcal{D} = \{1, 2\} \]

\[ s_0, ((0, P_1) v (1, P_1)) \wedge ((0, \phi) v (1, \phi)) \]
APT and CTL model-checking

Example

\[ \phi = E (EX P_1) U P_2 \quad \mathcal{D} = \{1,2\} \]

\[ s_0, ( (0, P_1) \lor (1, P_1) ) \land ( (0, \phi) \lor (1, \phi) ) \]

\[ \begin{align*}
  s_0, (0, P_1) & \rightarrow s_0, (1, P_1) \\
  s_1, P_1 & \rightarrow s_2, \bot \\
  s_2, \bot & \rightarrow s_1, T \\
  s_1, \phi & \rightarrow s_2, (0, P_1) \land (0, \phi) \\
  s_2, \bot & \rightarrow s_3, \bot \\
  s_2, (0, P_1) & \rightarrow s_3, P_1 \\
  s_3, \bot & \rightarrow s_1, (0, P_1) \land (0, \phi) \\
  s_3, \phi & \rightarrow s_3, T
\end{align*} \]

Wining for Adam
APT and CTL model-checking

Example

$\phi = \mathbb{E}(\mathbb{E}P_1) \cup P_2$

$\mathcal{D} = \{1, 2\}$

$S$

Wining for Adam
Example

APT and CTL model-checking

\[ \phi = E (EX P_1) U P_2 \]

\[ D = \{1,2\} \]

\[ s_0, ((0, P_1) \lor (1, P_1)) \land ((0, \phi) \lor (1, \phi)) \]

\[ s_0, (0, P_1) \lor (1, P_1) \rightharpoonup s_1, P_1 \]
\[ s_0, (0, P_1) \rightharpoonup s_2, P_1 \]
\[ s_0, (1, P_1) \rightharpoonup s_1, T \]
\[ s_0, (0, \phi) \lor (1, \phi) \rightharpoonup s_1, \phi \]
\[ s_0, (0, \phi) \rightharpoonup s_1, (0, P_1) \land (0, \phi) \]
\[ s_0, (1, \phi) \rightharpoonup s_2, (0, P_1) \land (0, \phi) \]
\[ s_1, (0, P_1) \land (0, \phi) \rightharpoonup s_1, (0, P_1) \land (0, \phi) \]
\[ s_2, (0, P_1) \land (0, \phi) \rightharpoonup s_2, (0, P_1) \land (0, \phi) \]
\[ s_3, P_1 \rightharpoonup s_3, \bot \]
\[ s_3, \phi \rightharpoonup s_3, \top \]

Wining for Adam

Example

APT and CTL model-checking

\[ \phi = E (E X P_1) U P_2 \]

\[ D = \{1, 2\} \]

\[ s_0, (0, P_1) V (1, P_1) \land (0, \phi) V (1, \phi) \]

\[ s_0, (0, P_1) \]

\[ s_0, (1, P_1) \]

\[ s_1, P_1 \]

\[ s_2, P_1 \]

\[ s_1, T \]

\[ s_2, \bot \]

Wining for Adam
Example

\( \phi = E (\text{EX } P_1) U P_2 \)

\( \mathcal{D} = \{1,2\} \)

\[ s_0, ((0,P_1) \lor (1,P_1)) \land ((0,\phi) \lor (1,\phi)) \]

\( s_0, (0,P_1) \lor (1,P_1) \)

\( s_0, (0,P_1) \quad s_0, (1,P_1) \)

\( s_1, P_1 \quad s_2, P_1 \)

\( s_1, T \quad s_2, \bot \)

Wining for Adam
$\phi = E (EX P_1) U P_2 \quad \mathcal{D} = \{1, 2\}$

Example

$\phi = E (EX P_1) U P_2$

$s_0, (0, P_1) \lor (1, P_1) \wedge ((0, \phi) \lor (1, \phi))$

$s_0, (0, P_1) \lor (1, P_1)$

$s_0, (0, P_1) \quad s_0, (1, P_1)$

$s_1, P_1 \quad s_2, P_1$

$s_1, T \quad s_2, \bot$

$s_0, (0, \phi) \quad s_0, (1, \phi)$

$s_1, P_1 \quad s_2, \bot$

$s_1, (0, P_1) \wedge (0, \phi)$

$s_2, (0, P_1) \wedge (0, \phi)$

$s_2, (0, P_1) \quad s_2, (0, \phi)$

$s_2, (0, P_1) \quad s_2, (0, \phi)$

$S_1, (0, \phi)$

$s_1, (0, \phi)$

$s_2, (0, \phi)$

$s_2, (0, \phi)$

$s_3, P_1 \quad s_3, \bot$

$s_3, T$

$s_0, \phi$

Wining for Adam
Example

APT and CTL model-checking

\( \phi = E (EX P_1) U P_2 \)

\( \mathcal{D} = \{1, 2\} \)

\[
\begin{align*}
s_0, \left( (0, P_1) \lor (1, P_1) \right) \land \left( (0, \phi) \lor (1, \phi) \right) \\
s_0, (0, P_1) \lor (1, P_1) \\
s_0, (0, P_1) \\
s_0, (1, P_1) \\
s_1, P_1 \\
s_1, T \\
s_2, P_1 \\
s_2, \bot \\
s_0, (0, \phi) \\
s_0, (1, \phi) \\
s_1, \phi \\
s_1, (0, P_1) \land (0, \phi) \\
s_1, (0, P_1) \\
s_2, \phi \\
s_2, (0, P_1) \land (0, \phi) \\
s_2, (0, P_1) \\
s_3, P_1 \\
s_3, \bot \\
s_3, T \\
s_3, \phi \\
s_3, T
\end{align*}
\]

Wining for Adam
Example

\[ \varphi = E (EX P_1) U P_2 \]

\[ D = \{1, 2\} \]

\[ s_0, (0, P_1) \lor (1, P_1) \land ((0, \varphi) \lor (1, \varphi)) \]

\[ s_0, (0, P_1) \lor (1, P_1) \]

\[ s_0, (0, P_1) \]

\[ s_0, (1, P_1) \]

\[ s_1, P_1 \]

\[ s_2, P_1 \]

\[ s_1, T \]

\[ s_2, \bot \]

\[ s_1, (0, P_1) \land (0, \varphi) \]

\[ s_1, (0, \varphi) \land (1, \varphi) \]

\[ s_1, (1, \varphi) \]

\[ s_2, (0, P_1) \]

\[ s_2, (0, \varphi) \]

\[ s_2, (1, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, P_1) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]

\[ s_2, (0, \varphi) \land (0, \varphi) \]
APT and CTL model-checking

Example

\[ \phi = E (EX P_1) U P_2 \]

\[ D = \{1, 2\} \]

\[ s_0, (0, P_1) \lor (1, P_1) \]

\[ s_0, (0, \phi) \lor (1, \phi) \]

\[ s_1, P_1 \]

\[ s_2, P_1 \]

\[ s_0, (0, P_1) \]

\[ s_0, (1, P_1) \]

\[ s_1, \top \]

\[ s_2, \bot \]

\[ s_1, (0, P_1) \land (0, \phi) \]

\[ s_2, (0, P_1) \land (0, \phi) \]

\[ s_2, (0, P_1) \]

\[ s_2, (0, \phi) \]

\[ s_3, P_1 \]

\[ s_3, \bot \]

Wining for Adam
Example

APT and CTL model-checking

\[ \phi = E (EX P_1) U P_2 \]

\( D = \{1, 2\} \)
Example

APT and CTL model-checking

$\phi = E (EX P_1) U P_2$

$D = \{1,2\}$

Adam has a winning strategy!

$\phi$ is not satisfied by $s_0$!
APT and CTL model-checking

Example

$$\phi = E (E X P_1) U P_2$$

$$\mathcal{D} = \{1, 2\}$$

\[s_0, ((0, P_1) \lor (1, P_1)) \land ((0, \phi) \lor (1, \phi))\]

\[s_0, (0, P_1) \lor (1, P_1)\] and \[s_0, (0, \phi) \lor (1, \phi)\]

\[s_1, P_1\] and \[s_2, \bot\]

\[s_1, T\] and \[s_2, \bot\]

\[s_1, (0, P_1) \land (0, \phi)\] and \[s_1, (0, P_1) \land (0, \phi)\]

\[s_2, (0, P_1) \lor (1, P_1)\] and \[s_2, (0, \phi) \lor (1, \phi)\]

\[s_3, P_1\] and \[s_3, \bot\]

\[s_3, T\] and \[s_3, \bot\]
Example

$$\phi = E (EX P_1) U P_2$$

$$D = \{1, 2\}$$

$$s_0, ((0,P_1) \lor (1,P_1)) \land ((0,\phi) \lor (1,\phi))$$

$$s_2, ((0,P_1) \lor (1,P_1)) \land ((0,\phi) \lor (1,\phi))$$

$$s_0, (0,P_1) \lor (1,P_1)$$

$$s_0, (0,\phi) \lor (1,\phi)$$

$$s_1, P_1$$

$$s_2, P_1$$

$$s_1, \phi$$

$$s_2, \phi$$

$$s_1, (0,P_1) \land (0,\phi)$$

$$s_1, (0,P_1) \land (0,\phi)$$

$$s_1, (0,P_1) \land (0,\phi)$$

$$s_1, (0,P_1) \land (0,\phi)$$

$$s_2, (0,P_1)$$

$$s_2, (0,P_1)$$

$$s_2, (0,\phi)$$

$$s_3, P_1$$

$$s_3, \perp$$

$$s_3, \phi$$

$$s_3, T$$

win. for $E$.
Example

APT and CTL model-checking

$\phi = E (EX P_1) U P_2$

$\mathcal{D} = \{1,2\}$

$s_0, \phi$

$s_0, ((0,P_1) \lor (1,P_1)) \land ((0,\phi) \lor (1,\phi))$

$s_0, (0,P_1) \lor (1,P_1)$

$s_0, (0,\phi) \lor (1,\phi)$

$s_1, P_1$

$s_2, P_1$

$s_1, \phi$

$s_1, (0,P_1) \land (0,\phi)$

$s_2, \phi$

$s_2, (0,P_1) \lor (1,P_1)$

$s_2, (0,\phi) \lor (1,\phi)$

$s_2, (1,P_1)$

$s_2, (0,P_1)$

$s_2, (1,\phi)$

$s_1, (0,P_1)$

$s_3, P_1$

$s_3, \bot$

$s_1, (0,\phi)$

$s_3, \phi$

$s_3, T$

win. for E.
Example

\[ \phi = E (EX P_1) U P_2 \]

\[ D = \{1,2\} \]

**APT and CTL model-checking**

![Diagram](image_url)

\[ s_0, ((0,P_1) \lor (1,P_1)) \land ((0,\phi) \lor (1,\phi)) \]

\[ s_0, (0,P_1) \lor (1,P_1) \]

\[ s_0, (0,\phi) \lor (1,\phi) \]

\[ s_2, ((0,P_1) \lor (1,P_1)) \land ((0,\phi) \lor (1,\phi)) \]

\[ s_2, (0,P_1) \lor (1,P_1) \]

\[ s_2, (0,\phi) \lor (1,\phi) \]

\[ s_1, (0,P_1) \land (0,\phi) \]

\[ s_1, P_1 \]

\[ s_1, \bot \]

\[ s_2, \bot \]

\[ s_2, \phi \]

\[ s_3, P_1 \]

\[ s_3, \bot \]

\[ s_3, \phi \]

\[ s_3, T \]

\[ \text{win. for E.} \]
Example

APT and CTL model-checking

\[ \phi = E (EX P_1) U P_2 \]

\[ \mathcal{D} = \{1, 2\} \]

\[ s_0, ((0, P_1) \lor (1, P_1)) \land ((0, \phi) \lor (1, \phi)) \]

\[ s_2, ((0, P_1) \lor (1, P_1)) \land ((0, \phi) \lor (1, \phi)) \]

\[ s_1, (0, P_1) \land (0, \phi) \]

\[ s_2, (0, P_1) \lor (1, P_1) \]

\[ s_3, P_1 \]

\[ s_3, \perp \]

\[ \text{win. for E.} \]
Example

\[ \phi = E (EX P_1) U P_2 \]
\[ D = \{1,2\} \]

\[ s_0, ((0,P_1) \lor (1,P_1)) \land ((0,\phi) \lor (1,\phi)) \]

\[ s_1, P_1 \]
\[ s_2, P_1 \]
\[ s_1, \phi \]
\[ s_2, (0,P_1) \lor (1,P_1) \]
\[ s_2, (0,\phi) \land (1,\phi) \]
\[ s_1, (0,P_1) \land (0,\phi) \]
\[ s_2, (0,P_1) \lor (1,P_1) \]
\[ s_2, (0,\phi) \lor (1,\phi) \]
\[ s_1, (0,P_1) \land (0,\phi) \]

Win. for E.
Example

$$\phi = E \ (\text{EX} \ P_1) \ U \ P_2$$

$$\mathcal{D} = \{1, 2\}$$

APT and CTL model-checking

$$s_0, ((0, P_1) \lor (1, P_1)) \land ((0, \phi) \lor (1, \phi))$$

$$s_0, (0, P_1) \lor (1, P_1)$$

$$s_0, (0, \phi) \lor (1, \phi)$$

$$s_0, \phi$$

$$s_1, P_1$$

$$s_1, T$$

$$s_2, P_1$$

$$s_2, \bot$$

$$s_1, \phi$$

$$s_2, \phi$$

$$s_2, ((0, P_1) \lor (1, P_1)) \land ((0, \phi) \lor (1, \phi))$$

$$s_2, (0, P_1) \lor (1, P_1)$$

$$s_2, (0, \phi) \lor (1, \phi)$$

$$s_2, (1, P_1)$$

$$s_2, (1, \phi)$$

$$s_2, (0, \phi)$$

$$s_3, P_1$$

$$s_3, \bot$$

$$s_3, T$$

Win. for E.
Example

\( \phi = E (\text{EX } P_1) U P_2 \)

\( D = \{1, 2\} \)

\[
\begin{align*}
S_0, \phi & \\
S_0, (0, P_1) v (1, P_1) & \\
S_0, (0, \phi) v (1, \phi) & \\
S_1, P_1 & \\
S_1, T & \\
S_1, (0, P_1) & \\
S_1, (1, P_1) & \\
S_2, \bot & \\
S_2, (0, P_1) & \\
S_2, (1, P_1) & \\
S_3, P_1 & \\
S_3, \bot & \\
S_3, \phi & \\
S_3, T & \\
\end{align*}
\]
Example

\[ \phi = E (\text{EX } P_1) \cup P_2 \]

\[ \mathcal{D} = \{1, 2\} \]

\[ s_0, \left( (0, P_1) \lor (1, P_1) \right) \land \left( (0, \phi) \lor (1, \phi) \right) \]

\[ s_2, \left( (0, P_1) \lor (1, P_1) \right) \land \left( (0, \phi) \lor (1, \phi) \right) \]

APT and CTL model-checking
Example

\[ \phi = E (\text{EX } P_1) \cup P_2 \]\n
\[ D = \{1, 2\} \]

\[ s_0, ((0, P_1) \lor (1, P_1)) \land ((0, \phi) \lor (1, \phi)) \]

\[ s_2, ((0, P_1) \lor (1, P_1)) \land ((0, \phi) \lor (1, \phi)) \]

\[ s_1, (0, P_1) \land (0, \phi) \]

\[ s_3, (0, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (0, \phi) \]

\[ s_3, (0, P_1) \land (0, \phi) \]

\[ s_3, (0, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (0, \phi) \]

\[ s_3, (1, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (0, \phi) \]

\[ s_3, (0, P_1) \land (0, \phi) \]

\[ s_3, (0, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (0, \phi) \]

\[ s_3, (1, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (0, \phi) \]

\[ s_3, (0, P_1) \land (0, \phi) \]

\[ s_3, (0, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (0, \phi) \]

\[ s_3, (1, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (0, \phi) \]

\[ s_3, (0, P_1) \land (0, \phi) \]

\[ s_3, (0, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (0, \phi) \]

\[ s_3, (1, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (0, \phi) \]

\[ s_3, (0, P_1) \land (0, \phi) \]

\[ s_3, (0, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (0, \phi) \]

\[ s_3, (1, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (0, \phi) \]

\[ s_3, (0, P_1) \land (0, \phi) \]

\[ s_3, (0, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (0, \phi) \]

\[ s_3, (1, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (0, \phi) \]

\[ s_3, (0, P_1) \land (0, \phi) \]

\[ s_3, (0, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (0, \phi) \]

\[ s_3, (1, P_1) \land (1, \phi) \]

\[ s_3, (1, P_1) \land (0, \phi) \]
Example

$\phi = E (EX P_1) U P_2$

$\mathcal{D} = \{1, 2\}$

$\begin{cases}
    s_0, (0, P_1) \lor (1, P_1) \\
    s_0, (0, \phi) \lor (1, \phi)
\end{cases}
$

$\begin{cases}
    s_0, (0, P_1) \\
    s_0, (1, P_1) \\
    s_0, (0, \phi) \\
    s_0, (1, \phi)
\end{cases}$

$\begin{cases}
    s_1, P_1 \\
    s_2, P_1 \\
    s_1, \phi \\
    s_2, \bot
\end{cases}$

$\begin{cases}
    s_2, (0, P_1) \lor (1, P_1) \\
    s_2, (0, \phi) \lor (1, \phi)
\end{cases}$

$\begin{cases}
    s_2, (0, P_1) \\
    s_2, (1, P_1) \\
    s_2, (0, \phi) \\
    s_2, (1, \phi)
\end{cases}$

$\begin{cases}
    s_1, (0, P_1) \lor (0, \phi) \\
    s_1, (0, \phi)
\end{cases}$

$\begin{cases}
    s_3, P_1 \\
    s_3, \bot
\end{cases}$

$\begin{cases}
    s_3, \phi \\
    s_3, T
\end{cases}$

win. for E.
Example

APT and CTL model-checking

\[ \phi = E(\text{EX } P_1) U P_2 \quad \mathcal{D} = \{1, 2\} \]

\[ s_0, (0, P_1) \lor (1, P_1) \]
\[ s_0, (0, \phi) \lor (1, \phi) \]
\[ s_0, (0, P_1) \land (0, \phi) \]
\[ s_0, (1, P_1) \land (1, \phi) \]

\[ s_1, P_1 \rightarrow s_1, T \]
\[ s_2, P_1 \rightarrow s_2, \bot \]
\[ s_1, \phi \rightarrow s_1, (0, P_1) \lor (0, \phi) \]
\[ s_2, \phi \rightarrow s_2, (0, P_1) \lor (1, P_1) \]
\[ s_2, (0, \phi) \lor (1, \phi) \]

\[ s_1, (0, P_1) \rightarrow s_3, P_1 \rightarrow s_3, \bot \]
\[ s_1, (0, \phi) \rightarrow s_3, \phi \rightarrow s_3, T \]

\[ s_2, (0, P_1) \rightarrow s_2, (0, P_1) \](for E.)

win.
Example

\[ \phi = E (EX P_1) U P_2 \]

\[ D = \{1, 2\} \]

\[ s_0, (0, P_1) \lor (1, P_1) \]

\[ s_0, (0, \phi) \lor (1, \phi) \]

\[ s_0, (0, P_1) \lor (1, P_1) \]

\[ s_0, (1, P_1) \]

\[ s_0, (0, \phi) \]

\[ s_0, (1, \phi) \]

\[ s_1, P_1 \]

\[ s_2, P_1 \]

\[ s_2, \bot \]

\[ s_1, P_1 \]

\[ s_2, (0, P_1) \]

\[ s_2, (1, P_1) \]

\[ s_2, (0, \phi) \]

\[ s_2, (1, \phi) \]

\[ s_1, (0, P_1) \lor (1, P_1) \]

\[ s_1, (0, P_1) \land (0, \phi) \]

\[ s_1, (0, P_1) \land (0, \phi) \]

\[ s_2, (0, P_1) \lor (1, P_1) \]

\[ s_2, (0, \phi) \lor (1, \phi) \]

\[ s_2, (0, P_1) \]

\[ s_2, (1, P_1) \]

\[ s_2, (0, \phi) \]

\[ s_2, (1, \phi) \]

\[ s_3, P_1 \]

\[ s_3, \bot \]

\[ s_3, P_1 \]

\[ s_3, \phi \]

\[ s_3, T \]

\[ \text{win. for E.} \]
Example

APT and CTL model-checking

\( \phi = E (EX P_1) U P_2 \)

\( D = \{1, 2\} \)

\( s_0, (0, P_1) \lor (1, P_1) \)

\( s_0, (0, P_1) \lor (1, P_1) \land ((0, \phi) \lor (1, \phi)) \)

\( s_0, (0, P_1) \lor (1, P_1) \)

\( s_0, (0, \phi) \lor (1, \phi) \)

\( s_0, (0, \phi) \lor (1, \phi) \land ((0, \phi) \lor (1, \phi)) \)

\( s_0, (0, P_1) \lor (1, P_1) \land ((0, \phi) \lor (1, \phi)) \)

\( s_1, P_1 \)

\( s_2, P_1 \)

\( s_1, \phi \)

\( s_2, \phi \)

\( s_2, (0, P_1) \lor (1, P_1) \)

\( s_2, (0, \phi) \lor (1, \phi) \)

\( s_2, (0, \phi) \lor (1, \phi) \land ((0, \phi) \lor (1, \phi)) \)

\( s_2, (0, P_1) \lor (1, P_1) \land ((0, \phi) \lor (1, \phi)) \)

\( s_2, (0, P_1) \lor (1, P_1) \)

\( s_2, (0, \phi) \lor (1, \phi) \)

\( s_2, (0, \phi) \lor (1, \phi) \land ((0, \phi) \lor (1, \phi)) \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)

\( s_3, \bot \)

\( s_3, P_1 \)
Example

APT and CTL model-checking

$\phi = \text{E} (\text{EX } P_1) \cup P_2$

$D = \{1, 2\}$

$s_0, ((0, P_1) \lor (1, P_1)) \land ((0, \phi) \lor (1, \phi))$

$s_2, ((0, P_1) \lor (1, P_1)) \land ((0, \phi) \lor (1, \phi))$

$s_1, P_1$

$s_2, P_1$

$s_1, \phi$

$s_2, \perp$

$s_1, (0, P_1) \land (0, \phi)$

$s_1, (0, P_1)$

$s_0, (0, P_1) \lor (1, P_1)$

$s_0, (0, \phi) \lor (1, \phi)$

$s_3, P_1$

$s_3, \perp$

$s_3, \phi$

$s_3, T$

win. for E.
Example

\[ \phi = E(\text{EX} P_1) U P_2 \]

\[ \mathcal{D} = \{1, 2\} \]
Example

\[ \phi = E (EX P_1) U P_2 \]

\[ \mathcal{D} = \{1, 2\} \]

\[
\begin{align*}
    s_0, \phi \\
    s_0, (0, P_1) \lor (1, P_1) \\
    s_0, (0, \phi) \lor (1, \phi) \\
    s_1, P_1 \\
    s_2, P_1 \\
    s_1, \phi \\
    s_2, \bot \\
    s_1, (0, P_1) \land (0, \phi) \\
    s_1, (0, \phi) \\
    s_2, (0, P_1) \\
    s_2, (0, \phi) \\
    s_2, (1, P_1) \\
    s_2, (0, P_1) \\
    s_2, (1, \phi) \\
    s_2, (0, \phi) \\
    s_3, P_1 \\
    s_3, \bot \\
    s_3, \phi \\
    s_3, T
\end{align*}
\]
Example

\( \phi = E \ (EX \ P_1) \ U \ P_2 \)

\[ \mathcal{D} = \{1, 2\} \]

Eve has a winning strategy! \( \phi \) is satisfied by \( s_0 \)!
Model-checking procedure (overview)

CTL
OK for CTL!

And for QCTL?
How to deal with formula of the form $\exists P. \phi$?
Property: [Muller & Schupp 85]
Let $\mathcal{A}$ be a $\langle \mathcal{D}, 2^{\mathcal{A}} \rangle$-NPT with $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$.
For $i = 1, 2$ we can build a $\langle \mathcal{D}, 2^{\mathcal{A}_i} \rangle$-NPT $\mathcal{B}_i$ s.t.
for any $2^{\mathcal{A}_i}$-labeled $\mathcal{D}$-tree $\mathcal{T}$, we have:
$$\mathcal{T} \in L(\mathcal{B}_i) \text{ iff } \exists \mathcal{T}' \in L(\mathcal{A}), \mathcal{T} \equiv_{\mathcal{A}_i} \mathcal{T}'.$$

Theorem: [Muller & Schupp 95]
Let $\mathcal{A}$ be a $\langle \mathcal{D}, 2^{\mathcal{A}} \rangle$-APT.
We can build an $\langle \mathcal{D}, 2^{\mathcal{A}} \rangle$-NPT $\mathcal{N}$ accepting the same language as $\mathcal{A}$.
$|\mathcal{N}| \leq 2^{O(|\mathcal{A}|.\text{idx}(\mathcal{A}) \log(|\mathcal{A}|.\text{idx}(\mathcal{A})))}$ and $\text{idx}(\mathcal{N}) \in O(|\mathcal{A}|.\text{idx}(\mathcal{A}))$
Projection

\[ \phi = A_a U (b \lor c) \]

\[ L(\mathcal{A}_\phi) = \{ T \mid T \models A_a U (b \lor c) \} \]

We use the previous prop. to define \( \mathcal{B}_1 \) s.t.

\[ L(\mathcal{B}_1) = \{ T \mid \exists T' \in L(\mathcal{A}_\phi) . T' \equiv \{a,b\} T \} \]
Projection

\[ \phi = AaU(b \lor c) \]

\[ L(\mathcal{A}_\phi) = \{ \mathcal{T} \mid \mathcal{T} \models AaU(b \lor c) \} \]

We use the previous prop. to define \( \mathcal{B}_1 \) s.t.

\[ L(\mathcal{B}_1) = \{ \mathcal{T} \mid \exists \mathcal{T}' \in L(\mathcal{A}_\phi) . \mathcal{T}' \equiv_{\{a,b\}} \mathcal{T} \} \]

Thus \( \mathcal{B}_1 \) recognizes the models of \( \exists c . AaU(b \lor c) \)
Projection and QCTL

Consider $\exists c. \phi$ with:

$$\phi = E(a \lor EX c) \cup b$$

Yes:

$\in L(\mathcal{B}_1)$

with $AP_1=\{a,b\}$ $AP_2=\{c\}$

Yes:

$\models \phi$
From QCTL to tree automata

CTL context

∃ p₁. ∃ p₂. ∃ p₃. φ ∈ QCTL
From QCTL to tree automata

CTL context

\[ \exists p_1. \exists p_3. \exists p_2. \]

CTL context  \( \phi \in \text{QCTL} \)

Alternating Parity Tree Automata

A_1

A_2

Alternating Parity Tree Automata
From QCTL to tree automata

- CTL context \( \varphi \in QCTL \)
- Alternating Parity Tree Automata
  - \( A_1 \)
  - \( A_2 \)
  - \( A'_1 \)
  - \( A'_2 \)
- Non-det. Parity Tree Automata

Simulation!
From QCTL to tree automata

CTL context  \( \phi \in \text{QCTL} \)

Non-det. Parity Tree Automata

Projection !

Alternating Parity Tree Automata

\( \exists p_1. \exists p_3. \exists p_2. \)
From QCTL to tree automata

CTL context

$\phi \in \text{QCTL}$

Non-det. Parity Tree Automata

CTL context+

automata

Alternating Parity Tree Automata

$\exists p_1. \exists p_3. \exists p_2.$
From QCTL to tree automata

$$\exists p_1. \exists p_3. \exists p_2.$$  

CTL context $\phi \in QCTL$

Alternating Parity Tree Automata

$A_1$

$A_2$

$A'_1$

$A'_2$

Non-det. Parity Tree Automata

$B_1$

$B_2$

$A_3$

CTL context+ automata

$A'_3$
From QCTL to tree automata

∃ \phi \in QCTL

CTL context

Alternating Parity Tree Automata

Non-det. Parity Tree Automata

CTL context+

automata
From QCTL to tree automata

∃ p₁. ∃ p₃.
∃ p₂.
CTL context ɸ ∈ QCTL

Alternating Parity Tree Automata

A₁
A₂
A'₁
A'₂

Non-det. Parity Tree Automata

B₁
B₂
A₃

CTL context+ automata

A'₃
B₃

A_ɸ
From QCTL to tree automata

CTL context

$\exists p_1. \exists p_3. \exists p_2. \phi \in QCTL$
From QCTL to tree automata

 CTL context

 $\exists p_1. \exists p_2. \exists p_3. \forall \phi \in \text{QCTL}

 A_1

 A_2

 $\phi \in \text{QCTL}$
From QCTL to tree automata

CTL context

\( \exists p_1. \exists p_3. \exists p_2. \)

\( \phi \in \text{QCTL} \)

Expon. blow-up!
From QCTL to tree automata

CTL context

$\exists p_1. \exists p_3. \exists p_2.$

CTL context $\phi \in QCTL$

Expon. blow-up!
From QCTL to tree automata

CTL context

$\exists p_1.
\exists p_3.
\exists p_2.
\phi \in QCTL$

Expon. blow-up!
From QCTL to tree automata

CTL context

\[ \exists p_1. \exists p_3. \exists p_2. \exists \phi \in \text{QCTL} \]

Expon. blow-up !
From QCTL to tree automata

CTL context

$\exists p_1. \exists p_3. \exists p_2.$

$\phi \in \text{QCTL}$

Expon. blow-up!

$B_1$

$B_4$
From QCTL to tree automata

CTL context

$$\exists p_1. \exists p_3. \exists p_2. \phi \in QCTL$$

$$A_1$$
$$A_2$$
$$A_1'$$
$$A_2'$$

Expon. blow-up!

$$B_1$$
$$B_2$$

$$A_3$$
$$A_3'$$

Expon. blow-up!

$$B_4$$

$$A_\phi$$
Theorem:
- Model-checking $Q^k\text{CTL}$ is $k$-EXPTIME-C.
- Model-checking $Q^k\text{CTL}^*$ is $(k+1)$-EXPTIME-C.
- Model-checking $Q\text{CTL}$ or $Q\text{CTL}^*$ is non-elementary.

Algorithm for $Q\text{CTL}$:
- build (exponent-sized) tree automaton for CTL part,
  
  + use projection for existential quantification,

  ➡ build product with the KS and check emptiness.

Hardness:
encoding of $k$-1-exponential-space alternating-Turing machine.
**Theorem:**
- Satisfiability of $Q^k\text{CTL}$ is $(k+1)$-EXPTIME-C.
- Satisfiability of $Q^k\text{CTL}^*$ is $(k+2)$-EXPTIME-C.
- Satisfiability of $\text{QCTL}$ or $\text{QCTL}^*$ is non-elementary.

For model-checking, the automata construction uses the degrees of the KS.

**Lemma:** $(\exists S . S \models \phi) \iff (\exists S_2 . S_2 \models \phi')$

- $\text{KS}$ with degrees in $\{1,2\}$
- A simple translation of $\phi$

$$E \phi U \psi \rightarrow E (\text{aux} \lor \phi) U (\neg \text{aux} \land \psi)$$
II- Application to ATLsc

- $\text{ATL}_{\text{sc}}$ model-checking
  [reduction to QCTL model-checking]

- $\text{ATL}_{\text{sc}}$ satisfiability
  [undecidability or reduction to QCTL satisf.]
From ATL$_{sc}$ to QCTL model checking

\[ G \models \varphi \]

finite CGS $\in$ ATL$_{sc}$

\[ S_G \models_t \varphi' \]

finite KS underlying $G$

+ each state $q$ is labeled by $P_q$

$\varphi'$ is built from $\varphi$ and the transition table of $G$.

Idea: labeling the nodes with propositions $m_{a,i}$ is used to mark a strategy for agent $a$
To simplify (a bit !) the construction, we reduce the problem to QCTL* model-checking.
What is a strategy for $a_1$?

$$f: \{s_0, s_1, s_2, s_3\}^* \rightarrow \{1, 2\}$$
What is a strategy for $a_1$?

$$f: \{s_0, s_1, s_2, s_3\}^* \rightarrow \{1, 2\}$$

$\star$: action 1

$\diamond$: action 2
What is a strategy for $a_1$?

$$f: \{s_0, s_1, s_2, s_3\}^* \rightarrow \{1, 2\}$$

- $\star$: action 1
- $\diamond$: action 2

$\mathcal{M} = \{1, 2\}$

$\text{Agt} = \{a_1, a_2\}$
From $\text{ATL}_{sc}$ to $\text{QCTL}^*$ model checking

Example:
Player $a$ has two moves: $m_1$ and $m_2$

\[
\langle a \rangle F \varphi
\]

\[
\exists m_1. \exists m_2. \text{AG}(\Phi_{\text{Strat}}(a)) \land A(\Phi_{\text{out}}^{[a]} \Rightarrow F \varphi')
\]

«exactly one $m_i$ for every reachable node»

«ensures that any two consecutive nodes are compatible with the transition table given the $m$ labeling»

ie defines the outcomes wrt the strategy.
From $\text{ATL}_{sc}$ to $\text{QCTL}^*$ model checking

**General idea:**  
Actions $m_1 \ldots m_k$ for $a$

\[
\langle \cdot a \rangle \mathbf{F} \varphi = \exists m_1^a \ldots m_k^a. \mathbf{AG} \left( \Phi_{\text{strat}}(a) \right) \land \mathbf{A} \left( \Phi_{\text{out}}^a \Rightarrow \mathbf{F} \overline{\varphi} \right)
\]

\[
\Phi_{\text{strat}}(a) = \bigvee_{q \in Q} \left( p_q \land \bigvee_{m_i \in \text{Mov}(q,a)} \left( m_i^a \land \bigwedge_{l \neq i} \neg m_l^a \right) \right)
\]

\[
\Phi_{\text{out}}^a = \mathbf{G} \left[ \bigwedge_{q \in Q} \left( (p_q \land m^a) \Rightarrow \mathbf{X} \left( \bigvee_{q' \in \text{Next}(q,a,m)} p_{q'} \right) \right) \right]
\]
From ATLsc to QCTL* model checking

In details…

Actions \( m_1 \ldots m_k \) for the agents \( A=\{a_1, \ldots, a_l\} \)

\[
\langle \cdot A \cdot \rangle \varphi_{\text{path}}^C = \exists m_1 \ldots m_k \ldots m_1^l \ldots m_k^l. \bigwedge_{a \in A} AG \left( \Phi_{\text{strat}}(a) \right) \land A \left( \Phi_{\text{out}}^{[CUA]} \Rightarrow \varphi_{\text{path}}^{CUA} \right)
\]

\( \Phi_{\text{strat}}(a) = \bigvee_{q \in Q} \left( p_q \land \bigvee_{m_i \in \text{Mov}(q,a)} (m_i^i \land \bigwedge_{l \neq i} \neg m_l^i) \right) \)

\( \Phi_{\text{out}}^{[CUA]} = G \left[ \bigwedge_{q \in Q} \left( (p_q \land \neg m) \Rightarrow X \left( \bigvee_{q' \in \text{Next}(q,CUA,\overline{m})} p_{q'} \right) \right) \right] \)

\( X \varphi^C = X \overline{\varphi}^C \)

\( \varphi U \psi^C = \overline{\varphi}^C U \overline{\psi}^C \)
From ATL\textsubscript{sc} to QCTL* model checking

Example...

$$\phi = \langle \cdot a_0 \cdot \rangle G ( \langle \cdot a_1 \cdot \rangle F P_1 \land \langle \cdot a_2 \cdot \rangle F P_2 )$$

\[ \mathcal{M} = \{1, 2\}, \text{Agt} = \{a_0, a_1, a_2\} \]

Assume \(a_0\) plays “1” everywhere, \(G\) becomes:

Assume \(a_0\) …, and \(a_1\) starts to play “1”, \(G\) becomes:

Assume \(a_0\) …, and \(a_2\) starts to play “1”, \(G\) becomes:
From ATLS\textsubscript{C} to QCTL* model checking

Example...

\[ \phi = \langle \cdot a_0 \cdot \rangle G ( \langle \cdot a_1 \cdot \rangle F P_1 \land \langle \cdot a_2 \cdot \rangle F P_2 ) \]

\( M = \{1,2\}, \text{Agt} = \{a_0, a_1, a_2\} \)

Assume \( a_0 \) plays "1" everywhere, \( G \) becomes:

Assume \( a_0 \) …, and \( a_2 \) starts to play "1", \( G \) becomes:

Assume \( a_0 \)…, and \( a_1 \) starts to play "1", \( G \) becomes:

Thus, we have \( G \models \phi \).
Example…

\[ \phi = (\cdot a_0 \cdot) G \left( (\cdot a_1 \cdot) F P_1 \land (\cdot a_2 \cdot) F P_2 \right) \]

\[ \begin{align*}
\Phi_{\text{strat}}(a_0) & \quad \phi \\
\exists m_1^0 m_2^0. & \quad \mathcal{A} G (m_1^0 \leftrightarrow -m_2^0) \land \mathcal{A} (\phi_{\text{out}} \Rightarrow G (\psi_1 \land \psi_2))
\end{align*} \]
Example...

\[
\phi = \langle \cdot a_0 \rangle G (\langle \cdot a_1 \rangle F P_1 \land \langle \cdot a_2 \rangle F P_2)
\]

\[
\phi_{strat}(a_0) \Rightarrow \phi
\]

\[
\exists m_1^0 m_2^0. AG (m_1^0 \leftrightarrow \neg m_2^0) \land A(\phi_{out} \Rightarrow G (\psi_1 \land \psi_2))
\]
Example…

$$\phi = \langle \cdot a_0 \cdot \rangle G \left( \langle \cdot a_1 \cdot \rangle F P_1 \land \langle \cdot a_2 \cdot \rangle F P_2 \right)$$

$$\psi_1$$

$$\psi_2$$

$$\phi = \exists m_1^0 m_2^0. AG (m_1^0 \leftrightarrow \neg m_2^0) \land A(\phi_{out} \Rightarrow G (\psi_1 \land \psi_2))$$

$$G: s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5 \rightarrow s_6$$

$$\phi_{strat}(a_0)$$

$$\phi = \emptyset$$

$$G (p_{s0} \land m_1^0 \Rightarrow X(p_{s1}) \land$$

$$p_{s0} \land m_2^0 \Rightarrow X(p_{s3}) \land$$

$$p_{s1} \land m_1^0 \Rightarrow X(p_{s4} \lor p_{s5}) \land$$

$$p_{s1} \land m_2^0 \Rightarrow X(p_{s3}) \land$$

$$p_{s2} \Rightarrow X(p_{s2}) \land$$

$$p_{s3} \Rightarrow X(p_{s3}) \land$$

$$p_{s4} \land m_1^0 \Rightarrow X(p_{s4} \lor p_{s5}) \land$$

$$p_{s4} \land m_2^0 \Rightarrow X(p_{s3}) \land$$

$$p_{s5} \land m_1^0 \Rightarrow X(p_{s4} \lor p_{s5}) \land$$

$$p_{s5} \land m_2^0 \Rightarrow X(p_{s3})$$

$$G (p_{s0} \land m_1^0 \land m_1^1 \Rightarrow X(p_{s1}) \land$$

$$p_{s0} \land m_1^0 \land m_2^1 \Rightarrow X(p_{s1}) \land$$

$$p_{s1} \land m_1^0 \land m_1^1 \Rightarrow X(p_{s4}) \land$$

$$p_{s1} \land m_1^0 \land m_2^1 \Rightarrow X(p_{s5}) \land$$

$$\cdots$$
Example...

\[ \phi = (\cdot a_0) G ( (\cdot a_1) F P_1 \land (\cdot a_2) F P_2) \]

\[ \phi_{\text{strat}}(a_0) \]

\[ \phi = \exists m_1^0 m_2^0. AG (m_1^0 \leftrightarrow \neg m_2^0) \land A(\phi_{\text{out}} \Rightarrow G (\psi_1 \land \psi_2)) \]

\[ G : s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5 \]

\[ \begin{align*}
G (p_{s0} \land m_1^0 & \Rightarrow X(p_{s1}) \land p_{s0} \land m_2^0 \Rightarrow X(p_{s3}) \land \exists m_1^1 m_2^1. AG (m_1^1 \leftrightarrow \neg m_2^1) \land A(\phi_{\text{out}} \Rightarrow F P_1) \land \\
\end{align*} \]

\[ \begin{align*}
\exists m_1^2 m_2^2. AG (m_1^2 \leftrightarrow \neg m_2^2) \land A(\phi_{\text{out}} \Rightarrow F P_2) \land \\
\end{align*} \]
Example...

\[ \phi = \langle \cdot a_0 \cdot \rangle G (\langle \cdot a_1 \cdot \rangle F P_1 \land \langle \cdot a_2 \cdot \rangle F P_2) \]

Let's see what happens when the players plays \( m_1 \) as before.

\[ \exists m_1^0 m_2^0. AG (m_1^0 \iff \neg m_2^0) \land A(\phi_{out} \Rightarrow G (\psi_1 \land \psi_2)) \]
Example...

\[ \phi = \langle \cdot a_0 \cdot \rangle G ( \langle \cdot a_1 \cdot \rangle F P_1 \land \langle \cdot a_2 \cdot \rangle F P_2 ) \]

Let's see what happens when the players plays \( m_1 \) as before.

\[ \rightarrow A(\phi_{out} \Rightarrow G (\psi_1 \land \psi_2)) \]
Example...

\[ \phi = \langle \cdot a_0 \cdot \rangle G ( \langle \cdot a_1 \cdot \rangle F P_1 \land \langle \cdot a_2 \cdot \rangle F P_2 ) \]

Let's see what happens when the players play \( m_1 \) as before.

\[ \rightarrow A(\phi_{out} \Rightarrow G (\psi_1 \land \psi_2)) \]

\[ G (p_{s0} \Rightarrow X(p_{s1}) \land p_{s1} \Rightarrow X(p_{s4} \lor p_{s5}) \land p_{s2} \Rightarrow X(p_{s2}) \land p_{s3} \Rightarrow X(p_{s3}) \land (p_{s4} \lor p_{s5}) \Rightarrow X(p_{s4} \lor p_{s5}) ) \]

\[ A(\phi_{out} \Rightarrow F P_1) \]

\[ G (p_{s0} \Rightarrow X(p_{s1}) \land p_{s1} \Rightarrow X(p_{s4}) \land p_{s2} \Rightarrow X(p_{s2}) \land p_{s3} \Rightarrow X(p_{s3}) \land p_{s4} \Rightarrow X(p_{s4} \lor p_{s5}) \land p_{s5} \Rightarrow X(p_{s4} ) ) \]

\[ A(\phi_{out} \Rightarrow F P_2) \]
Example...

\[ \phi = \langle \cdot a_0 \cdot \rangle G ( \langle \cdot a_1 \cdot \rangle F P_1 \land \langle \cdot a_2 \cdot \rangle F P_2 ) \]

Let's see what happens when the players play \( m_1 \) as before.

\[ \rightarrow A(\phi_{\text{out}} \Rightarrow G (\psi_1 \land \psi_2)) \]

\[ G (p_{s0} \Rightarrow X(p_{s1}) \land p_{s1} \Rightarrow X(p_{s4} \lor p_{s5}) \land p_{s2} \Rightarrow X(p_{s2}) \land p_{s3} \Rightarrow X(p_{s3}) \land (p_{s4} \lor p_{s5}) \Rightarrow X(p_{s4} \lor p_{s5}) ) \]

\[ A(\phi_{\text{out}} \Rightarrow F P_1) \]

\[ G (p_{s0} \Rightarrow X(p_{s1}) \land p_{s1} \Rightarrow X(p_{s4}) \land p_{s2} \Rightarrow X(p_{s2}) \land p_{s3} \Rightarrow X(p_{s3}) \land p_{s4} \Rightarrow X(p_{s4} \lor p_{s5}) \land p_{s5} \Rightarrow X(p_{s4}) ) \]

\[ A(\phi_{\text{out}} \Rightarrow F P_2) \]
Example…

1- from $s_0$, no selected run goes through $s_2$ or $s_3$!

2- from $s_0$, $s_1$, $s_4$, and $s_5$, any selected run visit eventually $s_4$!

2- from $s_0$, $s_1$, $s_4$, and $s_5$, any selected run visit eventually $s_5$!

$\phi = \langle .a_0. \rangle \text{G} (\langle .a_1. \rangle \text{F} P_1 \land \langle .a_2. \rangle \text{F} P_2)$

$A(\phi_{out} \Rightarrow \text{F} P_1)$

$G (p_{s0} \Rightarrow X(p_{s1}) \land p_{s1} \Rightarrow X(p_{s4}) \land p_{s2} \Rightarrow X(p_{s2}) \land p_{s3} \Rightarrow X(p_{s3}) \land p_{s4} \Rightarrow X(p_{s4} \lor p_{s5}) \land p_{s5} \Rightarrow X(p_{s4}) )$
Example…

1- from \( s_0 \), no selected run goes through \( s_2 \) or \( s_3 \)!

2- from \( s_0 \), \( s_1 \), \( s_4 \), and \( s_5 \), any selected run visits eventually \( s_5 \)!

\[ \phi \text{ holds for } S_G \text{ from } s_0! \]
ATL\textsubscript{sc} Model-checking $\rightarrow$ QCTL

**Theorem:**
Given $\phi \in \text{ATL}_{\text{sc}}$ and $G$ a CGS, we have:

$$G \models \phi \iff S_G \models \phi^\emptyset$$

**Theorem:**
Model-checking $\text{ATL}_{\text{sc}}$ and $\text{ATL}_{\text{sc}*}$ is non-elementary.

- The complexity for the fragment of $\text{ATL}_{\text{sc}}$ with $k$ non-trivial nested strategy quantifiers is $k$-EXPTIME-complete.
- And $(k+1)$-EXPTIME-complete for $\text{ATL}_{\text{sc}*}$…
What about satisfiability?

**Satisfiability:**

**input:** a formula $\phi$

**output:** yes iff there exists a model satisfying $\phi$.

**Several variants:**

**variant 1**

**input:** a formula $\phi$, a set of agents $\text{Agt}$ and an action alphabet $\mathcal{M}$

**variant 2**

**input:** a formula $\phi$

**output:** yes iff there exists a turn-based CGS satisfying $\phi$.

**variant 3**

**output:** yes iff there exists a finite model satisfying $\phi$.

...
Property:
An ATL-sc formula $\phi$ is satisfiable iff it is satisfiable in a CGS with $|\text{Agt}(\phi)|+1$ agents.

NB: $\text{Agt}(\phi) = \text{all agents involved in } \langle \cdot \cdot A \cdot \rangle \text{ modalities in } \phi$.

Proof idea:
Assume $\phi$ is satisfied by a CGS based on $\text{Agt}$ with $|\text{Agt}| > |\text{Agt}(\phi)|+1$. We can replace the $k$ agents in $\text{Agt} \setminus \text{Agt}(\phi)$ by a unique agent mimicking the actions of the removed players.

Note 1: the set of moves of the new player has size $M^k$...

Note 2: it is also true for the turn-based games...
What about satisfiability?

Can we use the same kind of reductions to QCTL problems?

No!

Reduction

CGS $G$ → $S_G$ a Kripke structure

ATL$_{sc}$ $\phi$ → $\psi_{G,\phi}$ a QCTL$^*$ formula

model-checking
What about satisfiability?

Can we use the same kind of reductions to QCTL problems?

No!

Reduction

\[ G \to \phi \to \text{model-checking} \]

CGS

ATL_{sc}

\[ S_G \text{ a Kripke structure} \]

\[ \psi_{G,\phi} \text{ a QCTL* formula} \]

depends on G and \( \phi \)!
Satisfiability

**Theorem** [Troquard and Walther]:
Satisfiability of $\text{ATL}_{sc}$ is undecidable.

Reduction from $S5^n$ satisfiability to $\text{ATL}_{sc}$ satisfiability.

**$S5^n$ modal logic:**
- formulas built from: $\wedge$, $\neg$, $P \in \text{AP}$, $\Diamond_i$ with $1 \leq i \leq n$
- interpreted over $M=(\mathcal{F}, \mathcal{V})$ with:
  - $\mathcal{F} = W_1 \times \ldots \times W_n$ and the transition relation over $W_i$ is universal:
    \[
    (w_1, \ldots, w_{i-1}, w_i, w_{i+1}, w_n) \rightarrow_i (w_1, \ldots, w_{i-1}, w'_i, w_{i+1}, \ldots, w_n)
    \]
  - $\mathcal{V}: \mathcal{F} \rightarrow 2^{\text{AP}}$

And:
\[
M, w \models \Diamond_i \phi \iff \exists w'_i \in W_i \text{ s.t. } M, w[w_i \rightarrow w'_i] \models \phi
\]

When $n>2$:
- satisfiability over (in)finite models is undecidable for $S5^n$. [A. Kurucz, 2003]
- $S5^n$ does not have the finite model property. [R. Maddux, 1980]
Satisfiability

Reduction from $S5^n$ satisfiability to $\text{ATL}_{sc}$ satisfiability [Troquard and Walther]:

Let $\phi$ be an $S5^n$ formula. We define $\overset{\sim}{\phi} \in \text{ATL}_{sc}$ as follows:

$$\overset{\sim}{\phi_1 \land \phi_2} = \overset{\sim}{\phi_1} \land \overset{\sim}{\phi_2} \quad \neg \overset{\sim}{\phi} = \neg \overset{\sim}{\phi} \quad \overset{\sim}{P} = \langle \cdot \emptyset \cdot \rangle X P \quad \overset{\sim}{\Diamond_i \phi} = \langle \cdot a_i \cdot \rangle \overset{\sim}{\phi}$$

Example: $\overset{\sim}{\Diamond_1 (\overset{\sim}{\Diamond_2 P} \land \overset{\sim}{\Diamond_2 P'} \land \neg \overset{\sim}{\Diamond_1 P})}$

$\rightarrow \langle \cdot a_1 \cdot \rangle (\langle \cdot a_2 \cdot \rangle \langle \cdot \emptyset \cdot \rangle X P \land \langle \cdot a_2 \cdot \rangle \langle \cdot \emptyset \cdot \rangle X P' \land \neg \langle \cdot a_1 \cdot \rangle \langle \cdot \emptyset \cdot \rangle X P)$

$n$ agents: Agent $i$ chooses strategies to follow the “moves” corresponding to the modality $\overset{\sim}{\Diamond_i}$.

Prop: $\phi$ is satisfiable iff $\langle \cdot \text{Agt.} \cdot \rangle \overset{\sim}{\phi}$ is satisfiable.
Reduction from \( \text{S5}^n \) satisfiability to \( \text{ATL}_{sc} \) satisfiability

[Troquard and Walther]:

\[
\phi_1 \land \phi_2 = \phi_1 \land \phi_2 \quad \neg \phi = \neg \phi \quad \langle \emptyset \rangle \mathbf{X} \ P \quad \langle \cdot a_i \cdot \rangle \phi
\]

1. Assume \( \phi \) is satisfiable.

\( M,w \models \phi \) with \( M = (\mathcal{F},\mathcal{V}) \) and \( \mathcal{F} = W_1 \times \ldots \times W_n \) \( W_i = \{ w_i^1, w_i^2, \ldots \} \)

\( C_M = \langle Q, R, \ell, \text{Agt}, \mathcal{M}, \text{Mov}, \text{Edge} \rangle \)

\( Q = \mathcal{F} \), \( R = Q \times Q \), \( \ell(w) = \mathcal{V}(w) \) \( (C_M \text{ is based on } M) \)

\( M = \{1, \ldots, \text{max } |W_i|\} \) \( (M \text{ is finite iff every } \mathcal{F} \text{ is finite}) \)

- Player \( a_i \) can choose the next position in \( W_i \).
  - thus: \( \text{Mov}(w, a_i) = \{1, \ldots, |W_i|\} \)
- And \( \text{Edge}(w, m) = w_m \)
  - with: \( w_m = \langle m_{m_1^1}, w_{m_2^2}, \ldots, w_{mn^n} \rangle \) if \( m = (m_1, \ldots, m_n) \)

A strategy for \( \text{Agt} \) fixes the next state.

And we get: \( M,w \models \phi \iff C_M,w' \models \langle \cdot \text{Agt.} \rangle \phi \)
Example: \( \phi = \lozenge_1 (\lozenge_2 P \land \lozenge_3 P' \land \neg \lozenge_1 P) \)

\[ \begin{array}{c}
W_1 \\
w_2^1 \\
w_1^2 \\
w_1^3 \\
\end{array} \quad \begin{array}{c}
W_2 \\
w_2^1 \\
w_2^2 \\
w_3^1 \\
\end{array} \quad \begin{array}{c}
W_3 \\
w_3^1 \\
w_3^2 \\
\end{array} \]

\( P \in \mathcal{V}(w_1^2w_2^2w_3^1) \)

\( P \in \mathcal{V}(w_1^1w_2^1w_3^2) \)

\( P' \in \mathcal{V}(w_1^2w_2^1w_3^2) \)

\( M, (w_1^1w_2^1w_3^1) \models \phi \)

\[ \langle \text{Agt.} \rangle \langle a_1 \rangle (\langle a_2 \rangle \langle \emptyset \rangle \mathbf{X} P \land \langle a_3 \rangle \langle \emptyset \rangle \mathbf{X} P' \land \neg \langle a_1 \rangle \langle \emptyset \rangle \mathbf{X} P) \]
Example: \( \Phi = \Diamond_1 (\Diamond_2 P \land \Diamond_3 P' \land \neg \Diamond_1 P) \)

\[ \begin{array}{c}
W_1 \\
\begin{array}{c}
\text{w}_1^1 \\
\text{w}_1^2 \\
\text{w}_1^3 \\
\end{array}
\end{array} \begin{array}{c}
W_2 \\
\begin{array}{c}
\text{w}_2^1 \\
\text{w}_2^2 \\
\end{array}
\end{array} \begin{array}{c}
W_3 \\
\begin{array}{c}
\text{w}_3^1 \\
\text{w}_3^2 \\
\end{array}
\end{array} \]

\( M, (w_1^1 w_2^1 w_3^1) \models \Phi \)

\( P \in \mathcal{V}(w_1^2 w_2^2 w_3^1) \)
\( P \in \mathcal{V}(w_1^1 w_2^1 w_3^2) \)
\( P' \in \mathcal{V}(w_1^2 w_2^1 w_3^2) \)

\( M_1 = \{1,2,3\} \)
\( M_2 = \{1,2\} \)
\( M_3 = \{1,2\} \)

\[ \langle \text{Agt.} \rangle \langle \cdot a_1 \cdot \rangle (\langle \cdot a_2 \cdot \rangle \langle \cdot \emptyset \cdot \rangle X P \land \langle \cdot a_3 \cdot \rangle \langle \cdot \emptyset \cdot \rangle X P' \land \neg \langle \cdot a_1 \cdot \rangle \langle \cdot \emptyset \cdot \rangle X P) \]
Example: \( \phi = \lozenge_1 (\lozenge_2 P \land \lozenge_3 P' \land \neg \lozenge_1 P) \)

\[
\begin{array}{c}
M_1 = \{1, 2, 3\} \\
M_2 = \{1, 2\} \\
M_3 = \{1, 2\}
\end{array}
\]

\[
\langle \text{Agt.} \rangle \langle \cdot a_1 \cdot \rangle (\langle \cdot a_2 \rangle \langle \cdot \emptyset \cdot \rangle X P \land \langle \cdot a_3 \rangle \langle \cdot \emptyset \cdot \rangle X P' \land \neg \langle \cdot a_1 \rangle \langle \cdot \emptyset \cdot \rangle X P)
\]
Reduction from $\text{S5}^n$ satisfiability to $\text{ATL}_{sc}$ satisfiability

[Troquard and Walther]:

\[ \phi_1 \land \phi_2 = \phi_1 \land \phi_2 \quad \neg \phi = \neg \phi \quad \mathcal{P} = \langle \emptyset \rangle \mathcal{X} \mathcal{P} \quad \Diamond_i \phi = \langle a_i \rangle \phi \]

2. Assume $\langle \cdot \text{Agt.} \rangle \phi$ is satisfiable.
There exists $\mathcal{C} = \langle Q, R, \ell, \text{Agt}, \mathcal{M}, \text{Mov}, \text{Edge} \rangle$ such that $\mathcal{C}, q \models \langle \cdot \text{Agt.} \rangle \phi$

We can assume $|\text{Agt}| = n$
$\mathcal{M}_\mathcal{C} = (\mathcal{F}, \mathcal{V})$ with $\mathcal{F} = W_1 \times \ldots \times W_n$ and: $W_i = \text{Mov}(q, a_i)$
$\rightarrow$ a world $w \in \mathcal{F}$ is a (complete) move in $\mathcal{C}$.
$\mathcal{V}(w) = \ell(\text{Edge}(q, w))$
The modality $\langle \cdot \text{Agt.} \rangle$ fixes a strategy $F$ for $\text{Agt}$ which defines a world $w_{F(q)}$ in $\mathcal{M}$.

And we have: $\mathcal{C}, q \models_F \phi \iff \mathcal{M}_\mathcal{C}, w_{F(q)} \models \phi$
Reduction from $S5^n$ satisfiability to $\text{ATL}_{sc}$ satisfiability

[Troquard and Walther]:

\[
\phi_1 \land \phi_2 = \phi_1 \land \phi_2 \quad \neg \phi = \neg \phi \quad P = \langle \emptyset \rangle \boxtimes P \quad \Diamond_i \phi = \langle a_i \rangle \phi
\]

2. Assume $\langle \cdot \text{Agt.}\rangle \overline{\phi}$ is satisfiable. There exists $C = \langle Q, R, \ell, \text{Agt}, \mathcal{M}, \text{Mov}, \text{Edge} \rangle$ such that $C, q \models \langle \cdot \text{Agt.}\rangle \overline{\phi}$

Remark:
- $M$ is infinite iff $C_M$ has an infinite action alphabet.
- $C$ is infinite iff $M_C$ is infinite.

We can assume $|\text{Agt}| = n$

$M_C = (\mathcal{F}, \mathcal{V})$ with $\mathcal{F} = W_1 \times \ldots \times W_n$ and:

$W_i = \text{Mov}(q, a_i)$

$\rightarrow$ a world $w \in \mathcal{F}$ is a (complete) move in $C$.

$\mathcal{V}(w) = \ell(\text{Edge}(q, w))$

The modality $\langle \cdot \text{Agt.}\rangle$ fixes a strategy $F$ for $\text{Agt}$ which defines a world $w_{F(q)}$ in $M$.

And we have:

$C, q \models F \overline{\phi} \iff M_C, w_{F(q)} \models \phi$
Example: \( \phi = \Diamond_1 (\Diamond_2 P \land \Diamond_3 P' \land \neg \Diamond_1 P) \)

with \( \widehat{\phi} = \langle .Agt. \rangle \langle .a_1. \rangle (\langle .a_2. \rangle \langle .\emptyset. \rangle X P \land \langle .a_3. \rangle \langle .\emptyset. \rangle X P' \land \neg \langle .a_1. \rangle \langle .\emptyset. \rangle X P) \)

- let’s take 1.1.1 for the strategy for Agt.
- let’s take 2 for the strategy for \( a_2 \).
- let’s take 2 for the strategy for \( a_3 \).
- let’s assume that all other moves lead to the yellow state.

\[ \models \neg P \land \neg P' \]
\[ \models P \land \neg P' \]
\[ \models \neg P \land P' \]

\( \Rightarrow \) The formula \( \widehat{\phi} \) is satisfiable!
Example: $\phi = \Diamond_1 (\Diamond_2 P \land \Diamond_3 P' \land \neg \Diamond_1 P)$

with $\widehat{\phi} = \langle \cdot \text{Agt.} \rangle \langle \cdot a_1 \rangle (\langle \cdot a_2 \rangle \langle \cdot \emptyset \rangle X P \land \langle \cdot a_3 \rangle \langle \cdot \emptyset \rangle X P' \land \neg \langle \cdot a_1 \rangle X P)$

$P \in \mathcal{V}(w_1^1w_2^2w_1^1)$
$P' \in \mathcal{V}(w_1^1w_2^1w_3^2)$

initial state $= w_1^1w_2^1w_3^1$

$\phi$ is satisfied!
Satisfiability

**Theorem** [Troquard and Walther]: Satisfiability of $\text{ATL}_{sc}$ is undecidable.

+ $\text{ATL}_{sc}$ does not have the finite-model property.
  $\text{ATL}_{sc}$ does not have the finite-branching property.

end of the story ?

No !
We can consider variants and subcases !
$\rightarrow$ turn-based structures, restrictions over strategies, ATL,....
ATL$_{sc}$ satisfiability

Why the reduction to QCTL is not possible? We need informations about the CGS in the QCTL formula in order to deal with strategies.

More precisely we need...
- either to know the transitions selected by a strategy,
- or the moves selected by a strategy (and played by the agents).

⇒ this is possible in two particular cases...

1. in turn-based games, or
2. when the action alphabet is fixed.
**ATL_{sc} satisfiability**

**Theorem**:
- Satisfiability of $\text{ATL}_{sc}$ is decidable for turn-based games.
- Satisfiability of $\text{ATL}_{sc}$ is decidable when action alphabet is bounded.

(Reduction to QCTL satisfiability !)

Both problems are non elementary.

(Reduction to QCTL satisfiability !)
Turn-based game:

The strategy for $a_1$ in $q$ is to go in $q'$.

marking the strategy = labeling by $\text{mov}_1$

1 prop. for every agent is enough!

$\text{mov}_i \rightarrow$ for the strat. of $a_i$
ATLsc satisifiability

\[ \langle \cdot A \rangle \phi_{\text{path}}^C = \exists \text{mov}_{a_1} \ldots \text{mov}_{a_l} \land \bigwedge_{a \in A} \text{AG} \left( \Phi_{\text{strat}}(a) \right) \land A \left( \Phi_{\text{out}}^{[\text{CUA}]} \Rightarrow \phi_{\text{path}}^C \right) \]

with

\[ \Phi_{\text{strat}}(a) = \left( \text{turn}_a \Rightarrow \text{EX}_1 \text{mov}_a \right) \]

\[ \Phi_{\text{out}}^{[\text{CUA}]} = \text{G} \left( \bigwedge_{a \in A} (\text{turn}_a \Rightarrow \text{X} \text{mov}_a) \right) \]

same approach as for model-checking…
**ATL_{sc} satisfiability**

Let $\phi$ be $\phi \wedge \phi_{tb}$ with $\phi_{tb} = \text{AG} \left( \bigvee_{a \in \text{Agt}} (\text{turn}_{a} \wedge \bigwedge_{a' \neq a} \neg \text{turn}_{a'}) \right)$

specifies a turn-based game

**Proposition** [LM2013]
Let $\phi$ be an $\text{ATL}_{sc}^*$ formula.

$\phi$ is satisfiable in a turn-based CGS iff the $\text{QCTL}^* \phi$ is satisfiable (in the tree semantics).

**Theorem** [LM2013]
Satisfiability of $\text{ATL}_{sc}$ and $\text{ATL}_{sc}^*$ formulas over turn-based CGS is non-elementary.
Bounded action alphabet
(Agt,M)-satisfiability

Problem:
Input: a finite set of moves \( \mathcal{M} \), a set of agents \( \text{Agt} \), and \( \phi \).
Question: \( \exists C = \langle \text{Agt}, Q, \ell, \mathcal{M}, \text{Mov}, \text{Edge} \rangle \) and \( q \in Q \) s.t. \( C, q \models \phi \)?

\[ \rightarrow \] This fixes the structure of the execution tree... and the reduction to QCTL is possible.

**Theorem** [LM2013]
(Agt,\( \mathcal{M} \))-Satisfiability for ATL\(_{sc} \) and ATL\(_{sc}^* \) is non-elementary.
Satisfiability overview

**Theorem:**
- Satisfiability of $\text{ATL}_{sc}$ is undecidable. [Troquard and Walther]
- Satisfiability of $\text{ATL}$ is decidable. [Goranko and van Drimmelen, Walther, Lutz, Wolter and Wooldridge]
- Satisfiability of $\text{ATL}_{sc}$ is decidable for turn-based games. [LM2013]
- $\langle \text{Agt}, M \rangle$-Satisfiability of $\text{ATL}_{sc}$ is decidable. [LM2013]
### Model-checking/satisfiability overview

<table>
<thead>
<tr>
<th></th>
<th>Model-checking</th>
<th>Satisfiability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ATL</strong></td>
<td>P-C</td>
<td>EXPTIME-C</td>
</tr>
<tr>
<td><strong>ATL^</strong>*</td>
<td>2-EXPTIME-C</td>
<td>2-EXPTIME-C</td>
</tr>
<tr>
<td><strong>ATL_{sc}</strong></td>
<td>non-elementary</td>
<td>undecidable</td>
</tr>
</tbody>
</table>

- and decidable..  
  - for turn-based CGS, or  
  - when Agt and $M$ are fixed.

(over CGS...)
Conclusion

There is a continuity from LTL/CTL to ATL, ATL_{sc}

ATL_{sc} is very expressive.
- Model-checking is decidable.
- Satisfiability is decidable for particular cases

QCTL is a powerful temporal logics. A natural extension of CTL. As expressive as MSO. A nice tool to encode other problems (for ex. the multi-agents logics ATL_{sc}, SL, …)

Other related topics:
memoryless strategies, bounded memory strategies, partial observation,…