Temporal logics for multi-agent systems

Expressiveness and algorithms

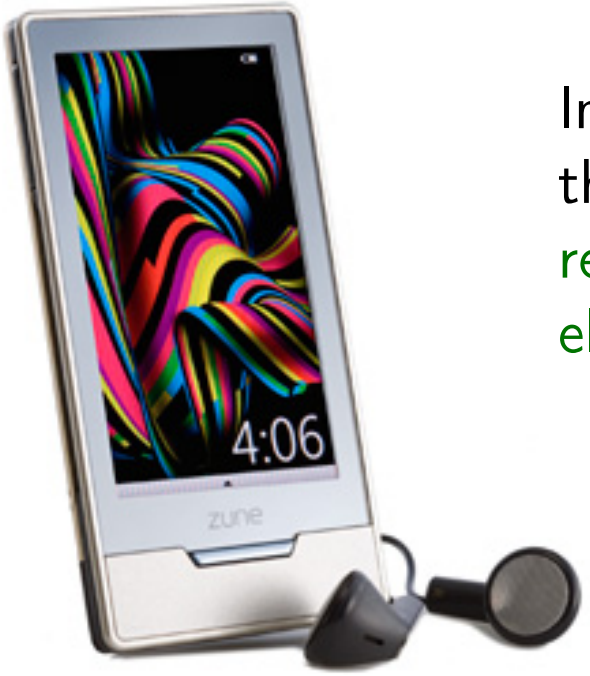
François Laroussinie — LIAFA – Univ. Paris-Diderot
Nicolas Markey — LSV – ENS Cachan

ESSLLI 2015

Barcelona, 10-14 august 2015

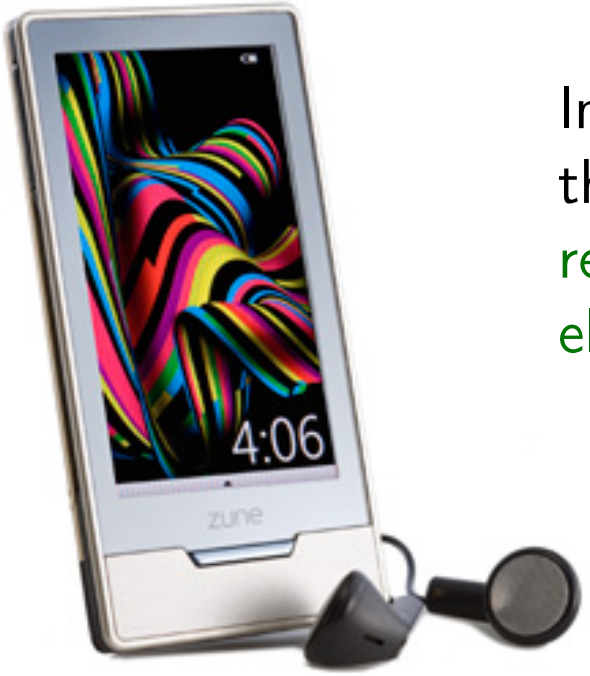
Verification of computerized systems

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.

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.

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);
```

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 Feb. 15th, 1982:

$$\boxed{\begin{array}{c} \texttt{days} = 777 \\ \texttt{year} = 1980 \end{array}}$$

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 Feb. 15th, 1982:

$$\boxed{\begin{array}{c}\texttt{days} = 777\\ \texttt{year} = 1980\end{array}} \longrightarrow \boxed{\begin{array}{c}\texttt{days} = 411\\ \texttt{year} = 1981\end{array}}$$

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 Feb. 15th, 1982:

$$\boxed{\begin{array}{c}\texttt{days} = 777\\ \texttt{year} = 1980\end{array}}$$

$$\downarrow$$

$$\boxed{\begin{array}{c}\texttt{days} = 411\\ \texttt{year} = 1981\end{array}}$$

$$\downarrow$$

$$\boxed{\begin{array}{c}\texttt{days} = 46\\ \texttt{year} = 1982\end{array}}$$

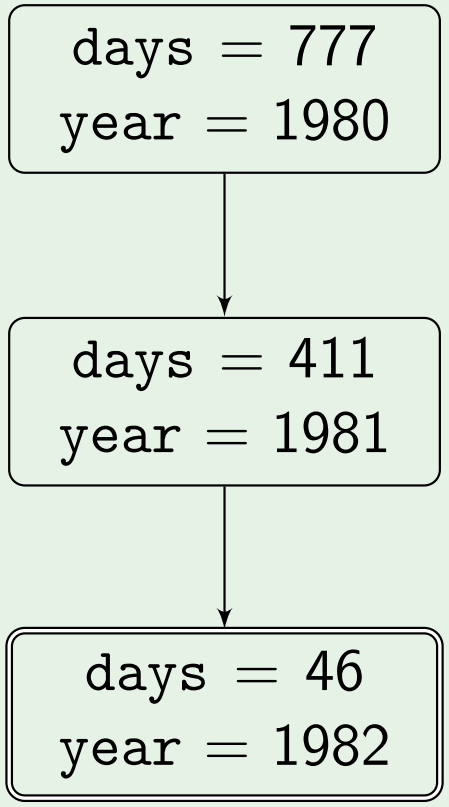
Verification of computerized systems – The Zune bug

```
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 Feb. 15th, 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:

$$\boxed{\begin{array}{l} \texttt{days} = 9132 \\ \texttt{year} = 1980 \end{array}}$$

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:

$$\boxed{\begin{array}{l} \texttt{days} = 9132 \\ \texttt{year} = 1980 \end{array}}$$

$$\boxed{\begin{array}{l} \texttt{days} = 366 \\ \texttt{year} = 2008 \end{array}}$$

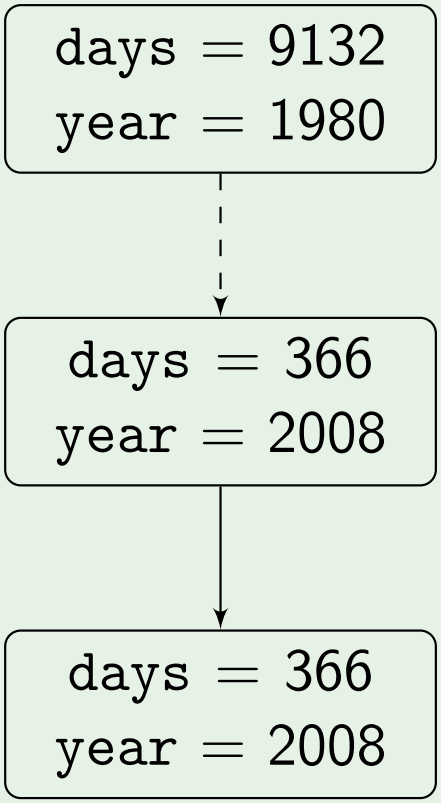
Verification of computerized systems – The Zune bug

```
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:



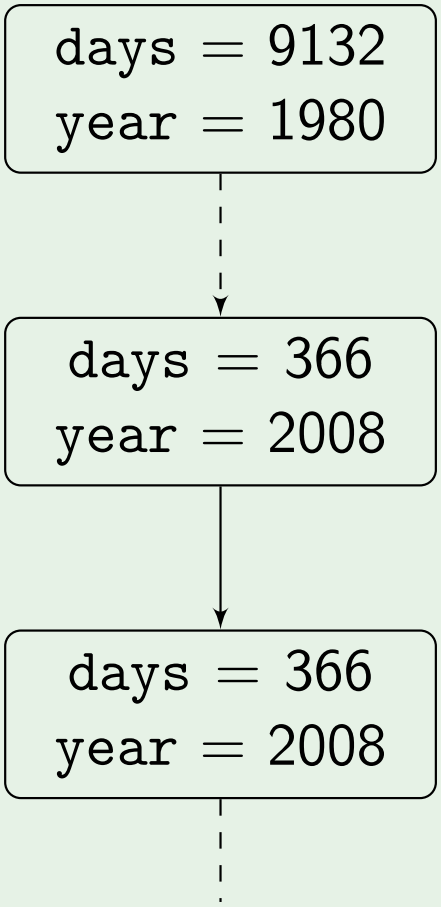
Verification of computerized systems – The Zune bug

```
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:



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

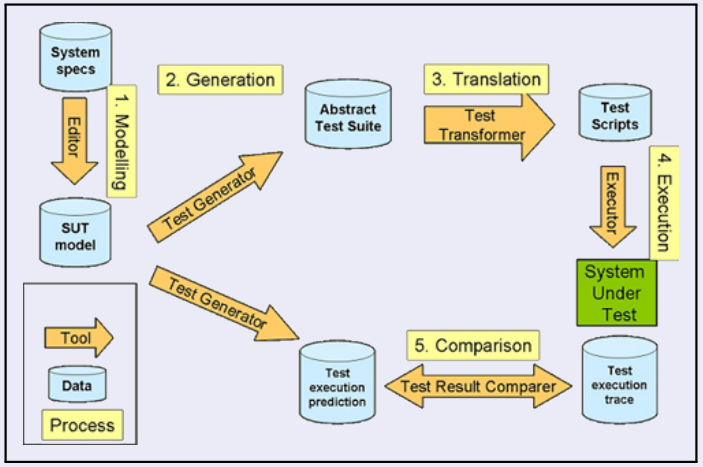
- proofs of correctness;
- exhaustive methods;
- automated techniques.

Theorem (Turing, 1936)

The halting problem for Turing machines is undecidable.

Different techniques

- (model-based) testing, statistical verification



Formal verification

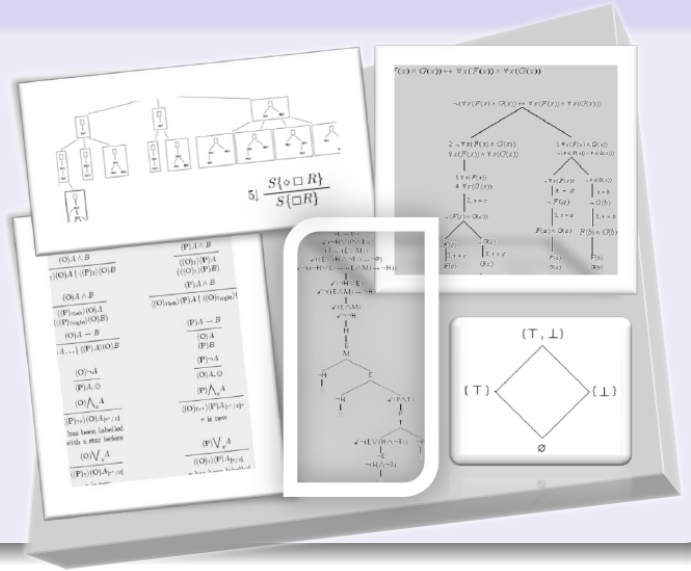
- proofs of correctness;
- exhaustive methods;
- automated techniques.

Theorem (Turing, 1936)

The halting problem for Turing machines is undecidable.

Different techniques

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



Formal verification

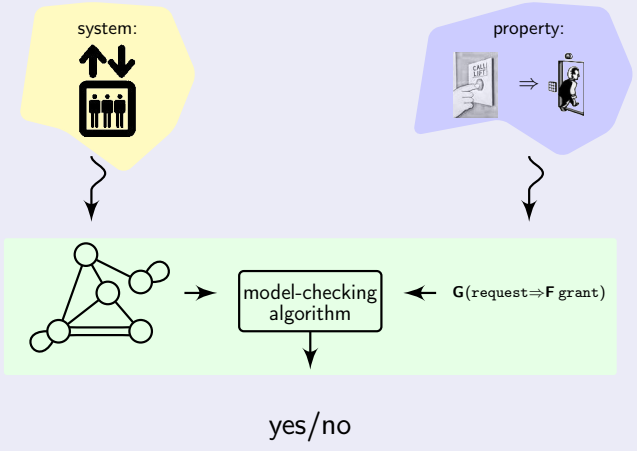
- proofs of correctness;
- exhaustive methods;
- automated techniques.

Theorem (Turing, 1936)

The halting problem for Turing machines is undecidable.

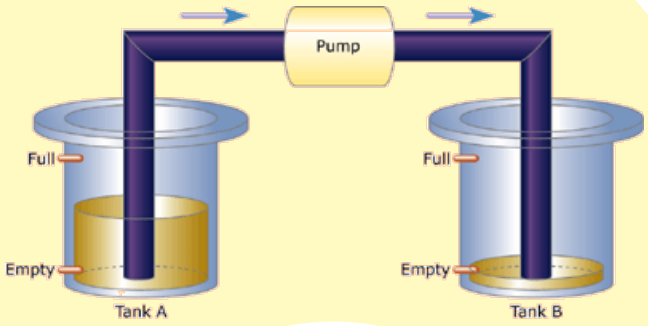
Different techniques

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



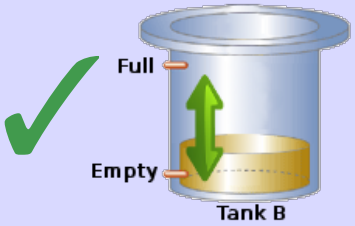
Model checking and synthesis

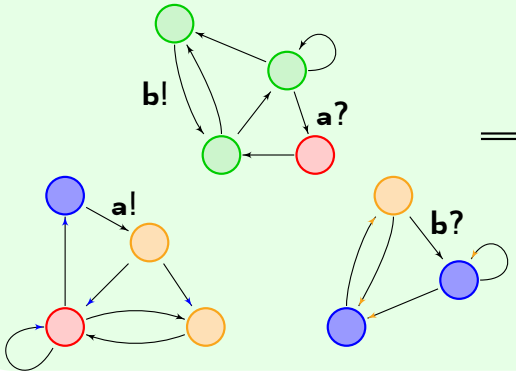
system:



[http://www.embedded.com]

property

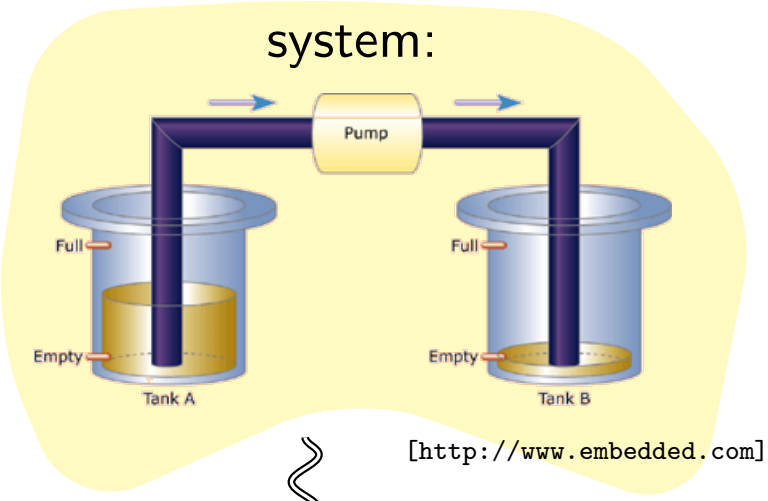


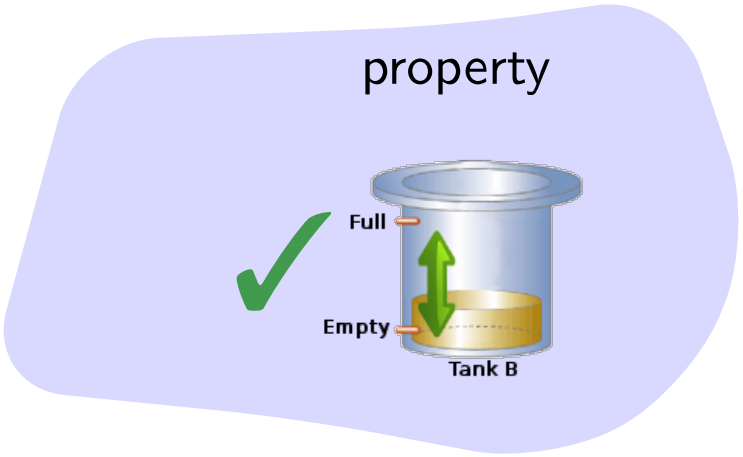


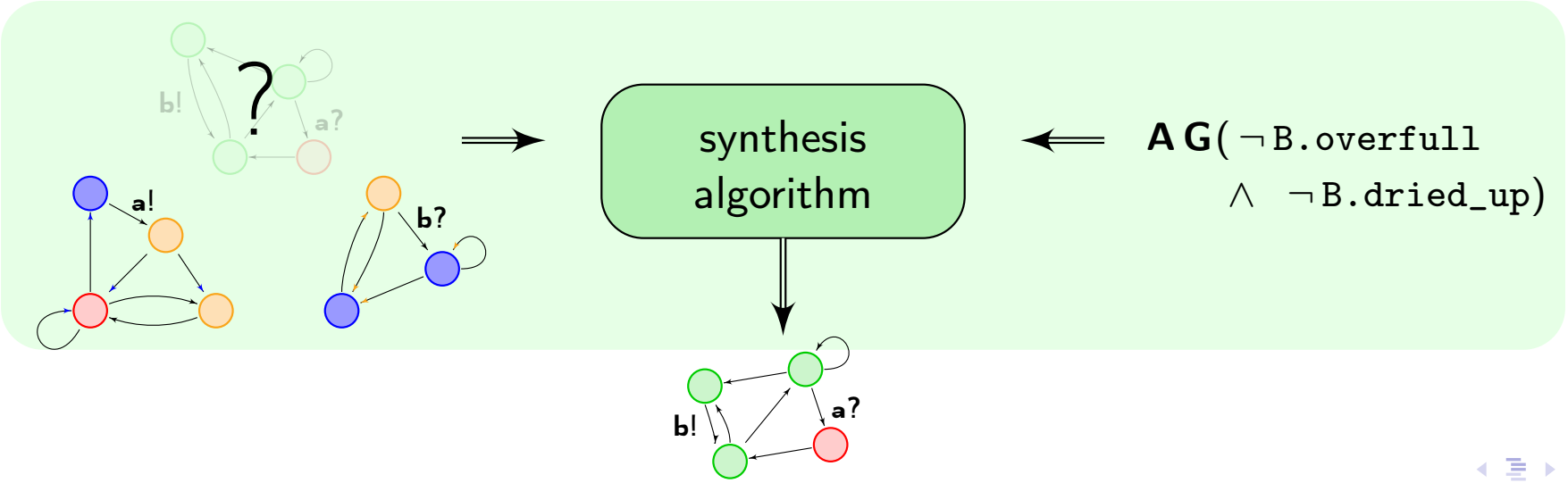
$\Longrightarrow$ model-checking algorithm $\Longleftarrow$ $\mathbf{A}\,\mathbf{G}(\neg\,\texttt{B.overfull} \wedge \neg\,\texttt{B.dried\_up})$

yes/no

Model checking and synthesis







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

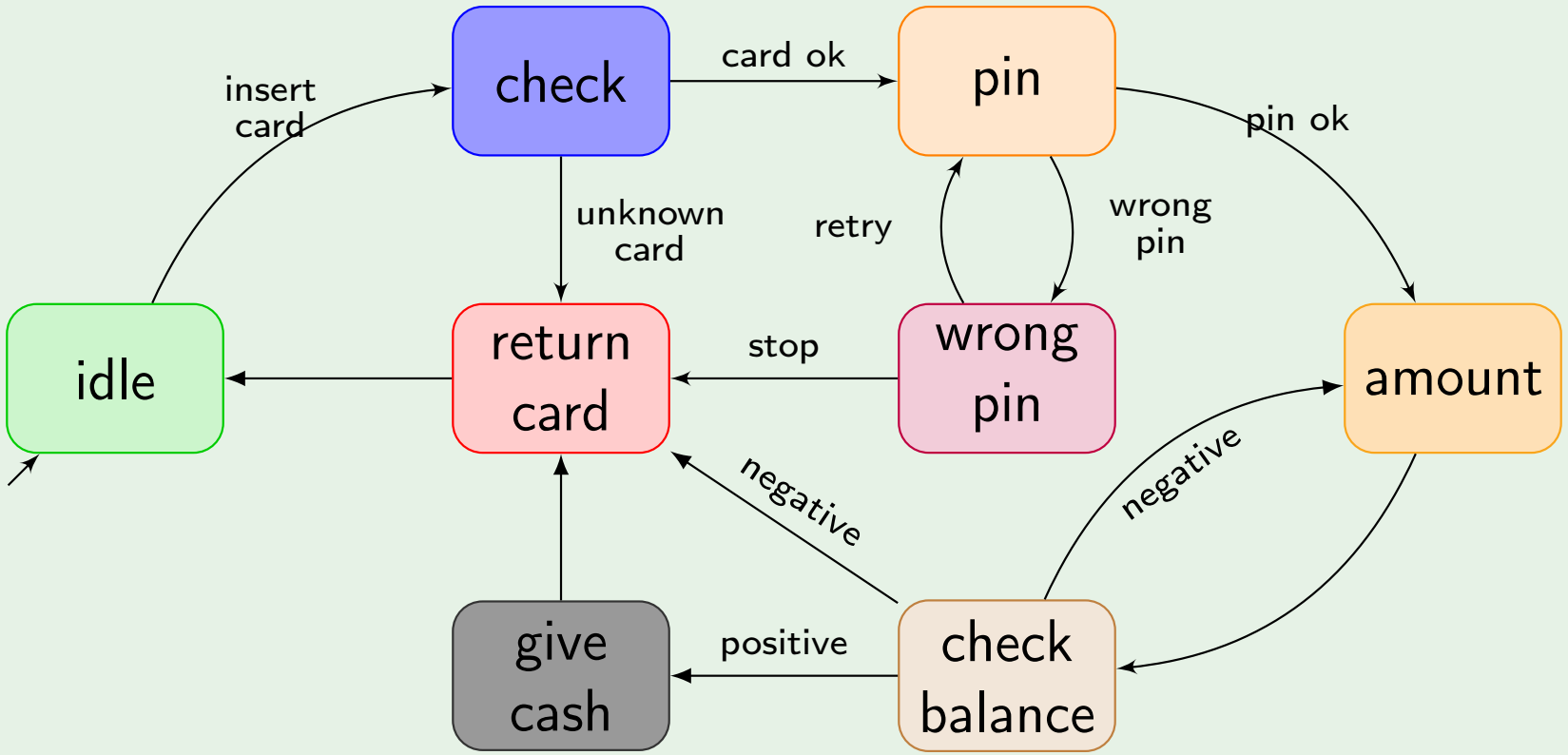
Modelling reactive systems: finite-state transition systems

We consider finite-state machines for modelling reactive systems.

Modelling reactive systems: finite-state transition systems

We consider finite-state machines for modelling reactive systems.

Example (An ATM)



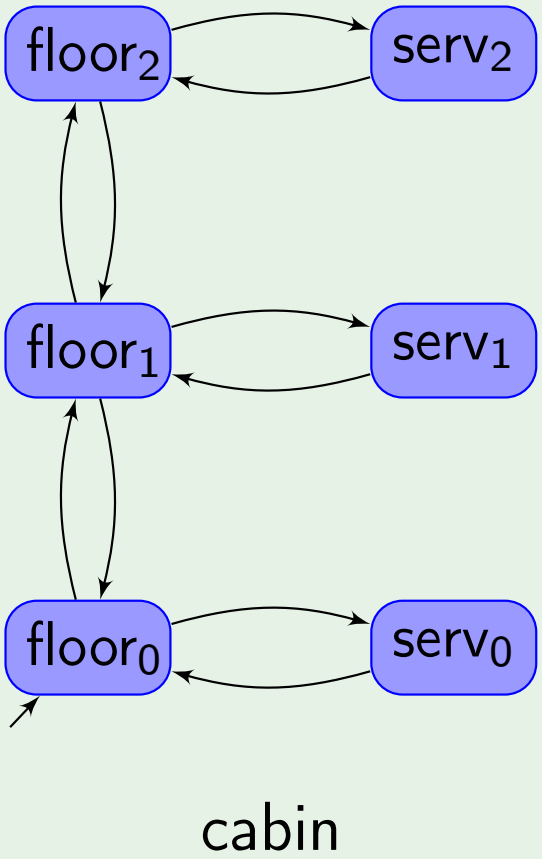
Modelling reactive systems: finite-state transition systems

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

Modelling reactive systems: finite-state transition systems

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

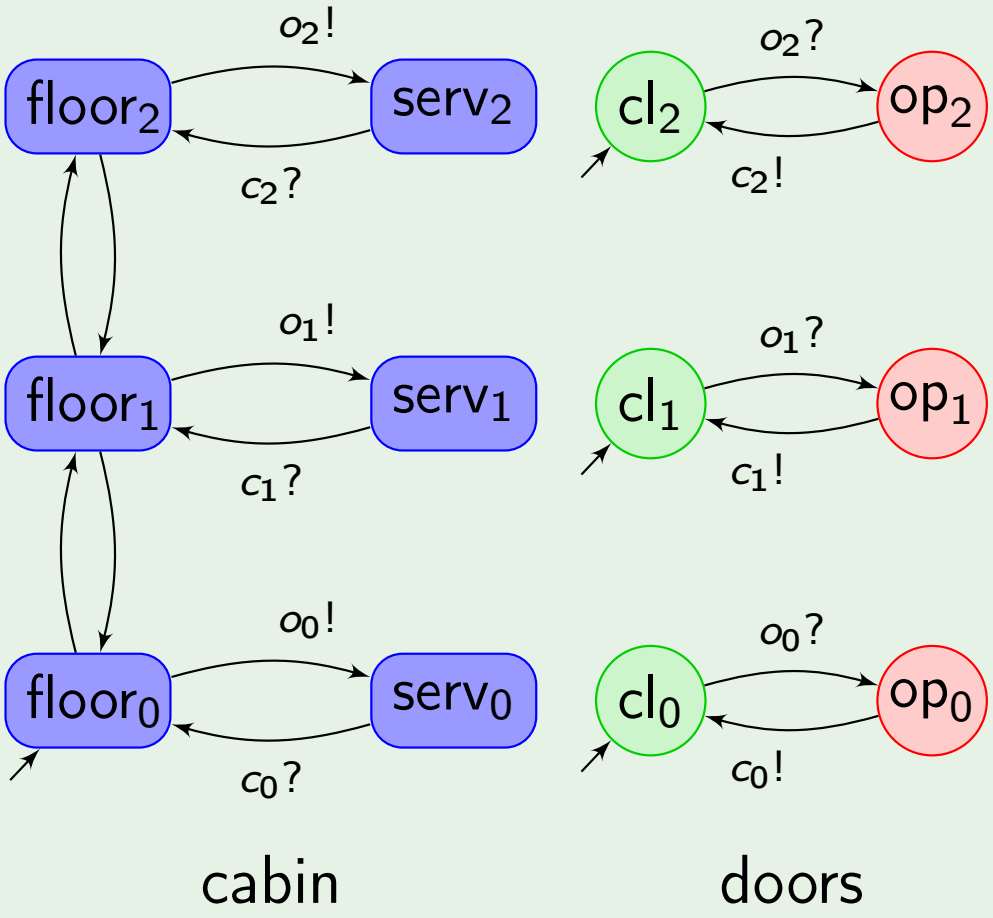
Example (An elevator)



Modelling reactive systems: finite-state transition systems

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

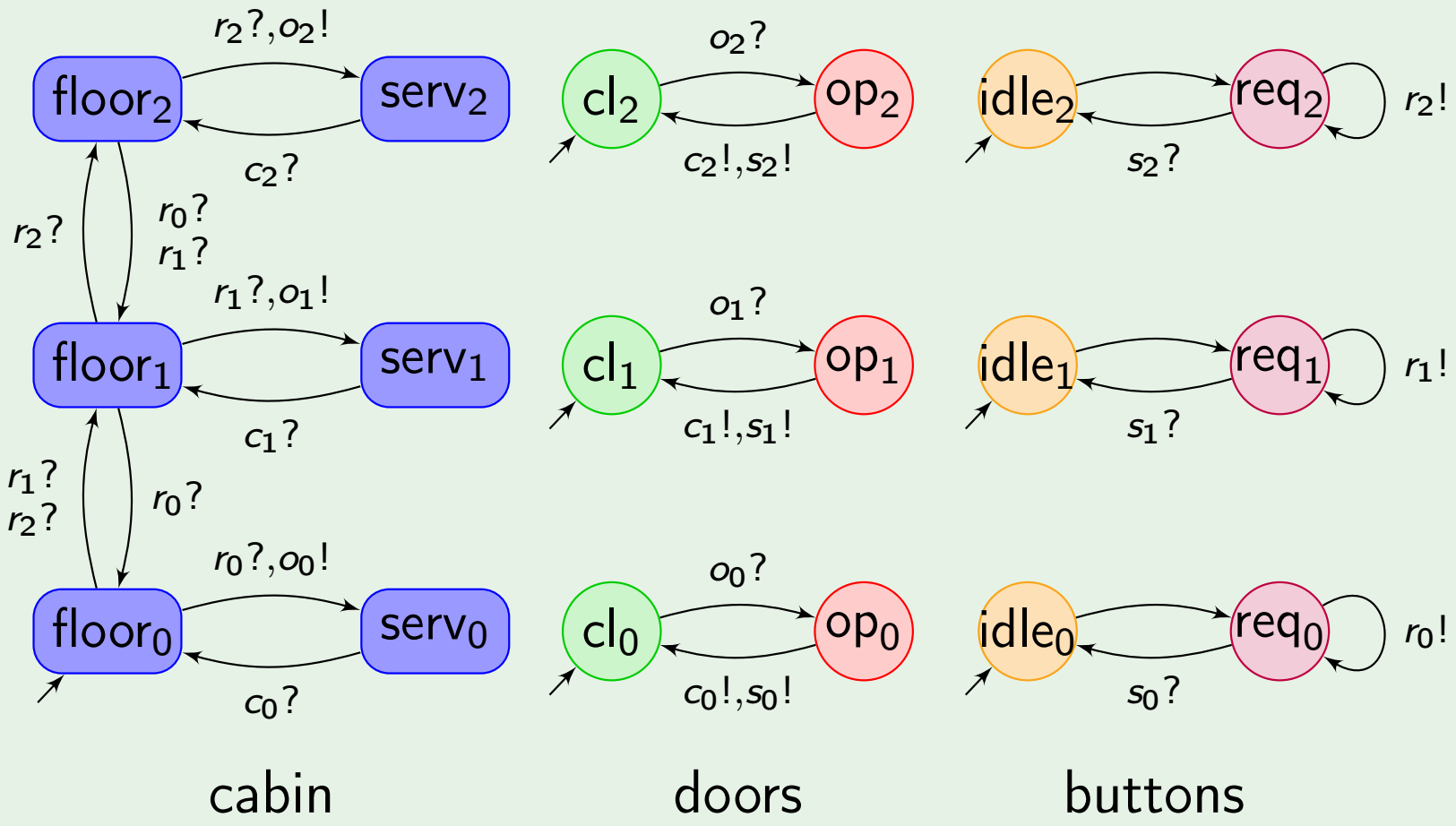
Example (An elevator)



Modelling reactive systems: finite-state transition systems

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

Example (An elevator)



Modelling 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
$$\forall q \in Q.\ \exists q' \in Q.\ (q, q') \in R$$
- $\ell \colon Q \to 2^{\mathsf{AP}}$ labels states with atomic propositions.

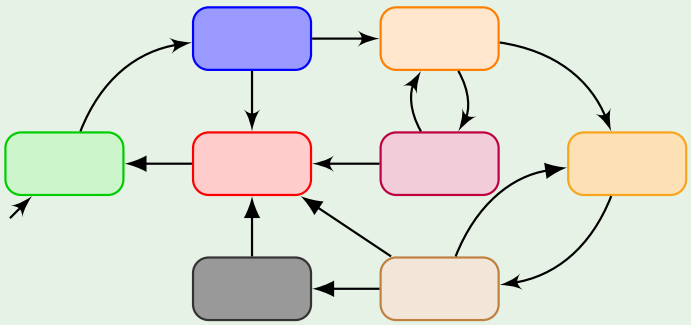
Modelling reactive systems: finite-state transition systems

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.

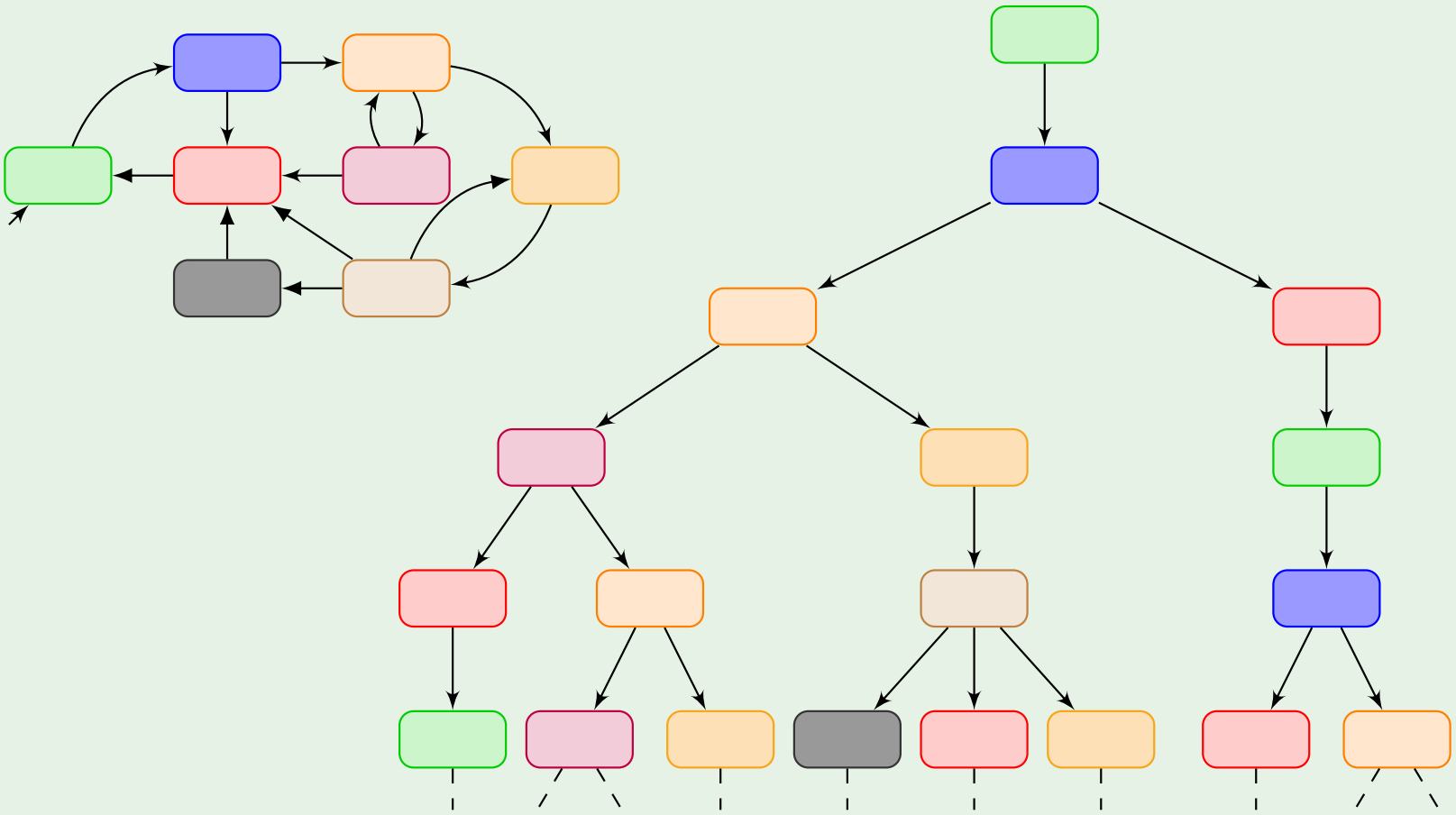
Example (Execution tree of a finite-state machine)



Modelling reactive systems: finite-state transition systems

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

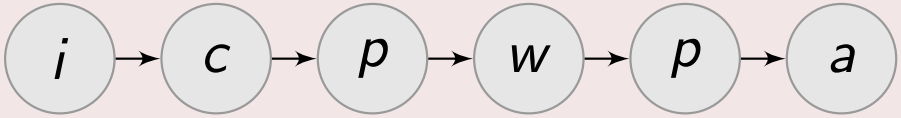
- A finite run of $\mathcal{M}$ from q is a pair (w, ℓ) where
  - $w = (q_i)_{0 \leq i \leq n}$ is a finite sequence of states with
    - $q_0 = q$
    - $(q_i, q_{i+1}) \in R$ for all $0 \leq i \leq n - 1$;
  - ℓ labels states with atomic propositions;

Modelling reactive systems: finite-state transition systems

Formally:

Definition

- A finite run of $\mathcal{M}$ from q is a pair (w, ℓ) 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 $0 \leq i \leq n - 1$;
  - ℓ labels states with atomic propositions;

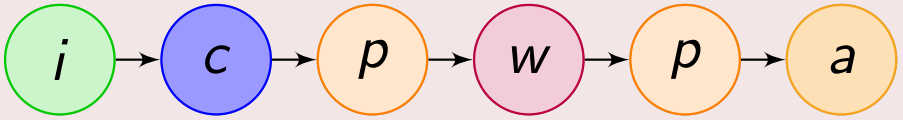


Modelling reactive systems: finite-state transition systems

Formally:

Definition

- A finite run of $\mathcal{M}$ from q is a pair (w, ℓ) 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 $0 \leq i \leq n - 1$;
  - ℓ labels states with atomic propositions;

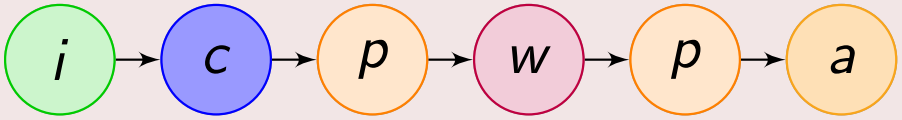


Modelling reactive systems: finite-state transition systems

Formally:

Definition

- A finite run of $\mathcal{M}$ from q is a pair (w, ℓ) 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 $0 \leq i \leq n-1$;
  - ℓ labels states with atomic propositions;



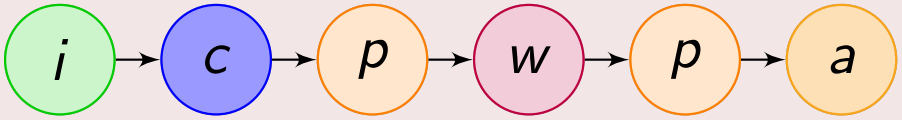
- The execution tree of $\mathcal{M}$ from q is a pair (T, ℓ) where
  - T is the set of finite runs of $\mathcal{M}$ from q;
  - ℓ labels runs with the label of their last state.

Modelling reactive systems: finite-state transition systems

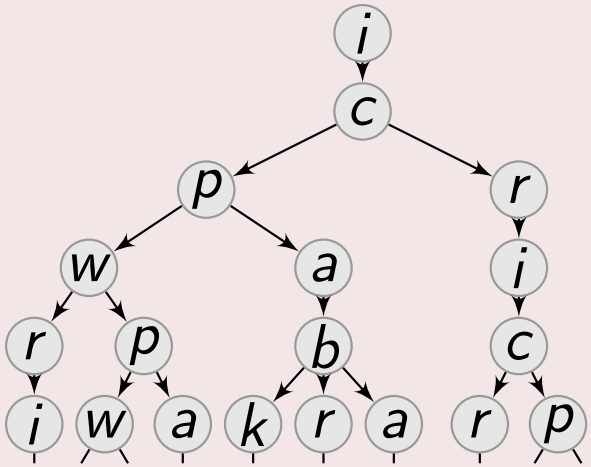
Formally:

Definition

- A finite run of $\mathcal{M}$ from q is a pair (w, ℓ) 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 $0 \leq i \leq n-1$;
  - ℓ labels states with atomic propositions;



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

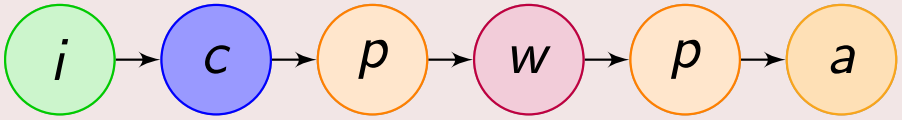


Modelling reactive systems: finite-state transition systems

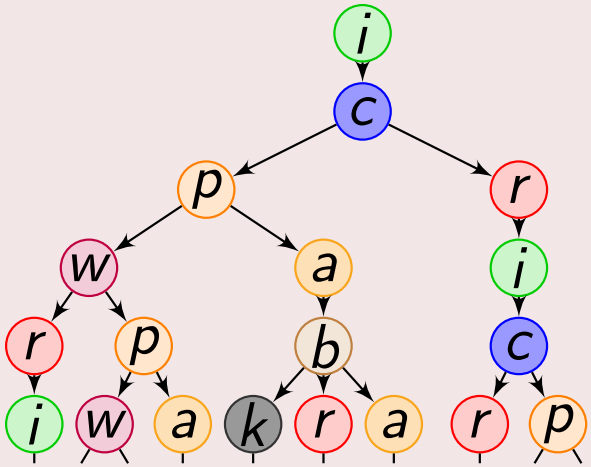
Formally:

Definition

- A finite run of $\mathcal{M}$ from q is a pair (w, ℓ) 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 $0 \leq i \leq n-1$;
  - ℓ labels states with atomic propositions;



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



Linear-time Temporal Logic (LTL)

Linear-time Temporal Logic (LTL)

- **atomic propositions:**  ,  , ...

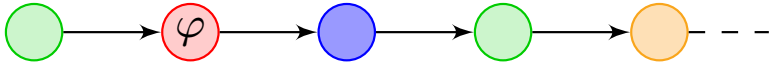
Linear-time Temporal Logic (LTL)

- **atomic propositions:**  ,  , ...
- **boolean combinators:** $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, ...

Linear-time Temporal Logic (LTL)

- **atomic propositions:** ●, ●, ...
- **boolean combinators:** $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, ...
- **temporal modalities:**

$\mathbf{X}\,\varphi$



"next φ"

$\varphi\ \mathbf{U}\ \psi$

"φ until ψ"

Linear-time Temporal Logic (LTL)

- **atomic propositions:** ●, ●, ...
- **boolean combinators:** $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, ...
- **temporal modalities:**

$\mathbf{X}\,\varphi$	$\bullet \to \varphi \to \bullet \to \bullet \to \bullet \dashrightarrow$	"next φ"
$\varphi\ \mathbf{U}\ \psi$	$\varphi \to \varphi \to \psi \to \bullet \to \bullet \dashrightarrow$	"φ until ψ"
$\texttt{true}\ \mathbf{U}\ \varphi \equiv \mathbf{F}\,\varphi$	$\bullet \to \bullet \to \bullet \to \varphi \to \bullet \dashrightarrow$	"eventually φ"
$\neg\ \mathbf{F}\ \neg\varphi \equiv \mathbf{G}\,\varphi$	$\varphi \to \varphi \to \varphi \to \varphi \to \varphi \dashrightarrow$	"always φ"

Linear-time Temporal Logic (LTL)

- **atomic propositions:** ◯, ◯, ...
- **boolean combinators:** $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, ...
- **temporal modalities:**

$\mathbf{X}\,\varphi$	$\bigcirc \to \bigcirc^{\varphi} \to \bigcirc \to \bigcirc \to \bigcirc \cdots$	"next φ"
$\varphi\ \mathbf{U}\ \psi$	$\bigcirc^{\varphi} \to \bigcirc^{\varphi} \to \bigcirc^{\psi} \to \bigcirc \to \bigcirc \cdots$	"φ until ψ"
$\texttt{true}\ \mathbf{U}\ \varphi \equiv \mathbf{F}\,\varphi$	$\bigcirc \to \bigcirc \to \bigcirc \to \bigcirc^{\varphi} \to \bigcirc \cdots$	"eventually φ"
$\neg\ \mathbf{F}\ \neg\varphi \equiv \mathbf{G}\,\varphi$	$\bigcirc^{\varphi} \to \bigcirc^{\varphi} \to \bigcirc^{\varphi} \to \bigcirc^{\varphi} \to \bigcirc^{\varphi} \cdots$	"always φ"

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

$$\mathcal{M} \models \varphi \quad \Longleftrightarrow \quad \forall \pi \in \mathcal{T}_{\mathcal{M}}.\ \pi \models \varphi.$$

Examples of LTL formulas

$\mathbf{F}\ \boxed{\text{floor}_1}$ *the cabin eventually reaches the first floor*

Examples of LTL formulas

$\mathbf{F}\ \mathsf{floor}_1$	*the cabin eventually reaches the first floor*
$\mathbf{G}(\mathsf{op}_1 \Rightarrow \mathsf{serv}_1)$	*if the door is open, the cabin is present*

Examples of LTL formulas

$\mathbf{F}\ \mathrm{floor}_1$	*the cabin eventually reaches the first floor*
$\mathbf{G}(\mathrm{op}_1 \Rightarrow \mathrm{serv}_1)$	*if the door is open, the cabin is present*
$\mathbf{G}(\mathrm{req}_0 \Rightarrow \mathbf{F}\ \mathrm{serv}_0)$	*any request is eventually served*

Examples of LTL formulas

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

Examples of LTL formulas

$\mathbf{F}\ \mathsf{floor}_1$	*the cabin eventually reaches the first floor*
$\mathbf{G}(\mathsf{op}_1 \Rightarrow \mathsf{serv}_1)$	*if the door is open, the cabin is present*
$\mathbf{G}(\mathsf{req}_0 \Rightarrow \mathbf{F}\ \mathsf{serv}_0)$	*any request is eventually served*
$\mathbf{G}\ \mathbf{F}\ \mathsf{idle}_0$	*the button is off infinitely many times*
$(\mathbf{G}\ \mathbf{F}\ \mathsf{serv}_0) \Rightarrow (\mathbf{G}\ \mathbf{F}\ \mathsf{req}_0)$	*serve infinitely many times only if there are infinitely many requests*

Satisfiability and model checking

Two main algorithmic problems

Satisfiability and model checking

Two main algorithmic problems

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

Verifying LTL properties

Verifying LTL properties

Theorem ([SC85])

LTL satisfiability and model checking are logspace-equivalent.

[SC85] Sistla, Clarke. The complexity of propositional linear temporal logics. J.ACM, 1985.

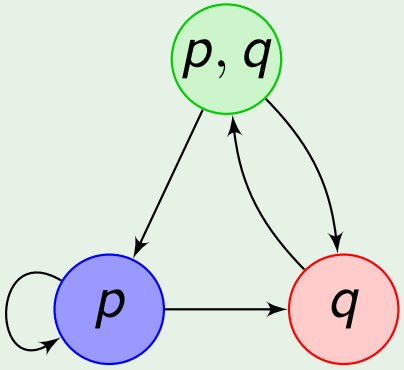
Verifying LTL properties

Theorem ([SC85])

LTL satisfiability and model checking are logspace-equivalent.

Proof (Sketch)

Encode behaviour of Kripke structure as an LTL formula:



[SC85] Sistla, Clarke. The complexity of propositional linear temporal logics. J.ACM, 1985.

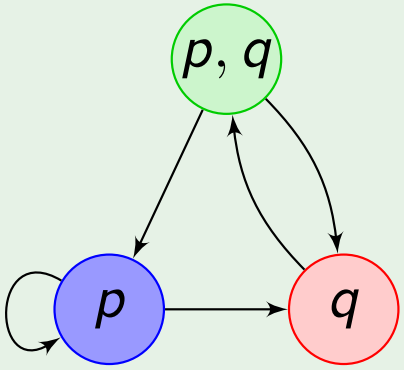
Verifying LTL properties

Theorem ([SC85])

LTL satisfiability and model checking are logspace-equivalent.

Proof (Sketch)

Encode behaviour of Kripke structure as an LTL formula:



$$\mathbf{G}\left[\qquad\qquad\qquad \right]$$

[SC85] Sistla, Clarke. The complexity of propositional linear temporal logics. J.ACM, 1985.

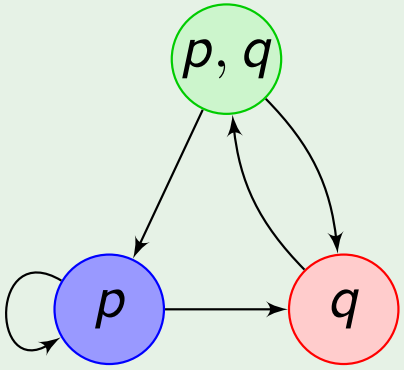
Verifying LTL properties

Theorem ([SC85])

LTL satisfiability and model checking are logspace-equivalent.

Proof (Sketch)

Encode behaviour of Kripke structure as an LTL formula:



$$\mathbf{G}\left[\ \text{exactly one state-proposition in } \{\bigcirc, \bigcirc, \bigcirc\}\ \right]$$

[SC85] Sistla, Clarke. The complexity of propositional linear temporal logics. J.ACM, 1985.

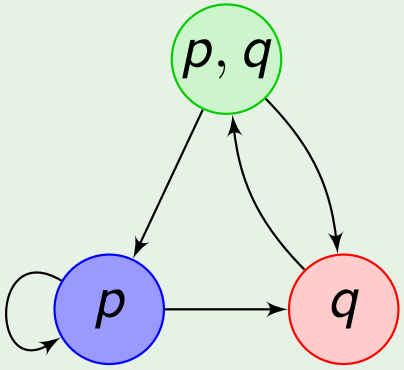
Verifying LTL properties

Theorem ([SC85])

LTL satisfiability and model checking are logspace-equivalent.

Proof (Sketch)

Encode behaviour of Kripke structure as an LTL formula:



$$\mathbf{G}\left[\bigwedge \begin{array}{l} \text{exactly one state-proposition in } \{\bigcirc, \bigcirc, \bigcirc\} \\ (p \Leftrightarrow (\bigcirc \vee \bigcirc) \wedge (q \Leftrightarrow (\bigcirc \vee \bigcirc) \end{array}\right]$$

[SC85] Sistla, Clarke. The complexity of propositional linear temporal logics. J.ACM, 1985.

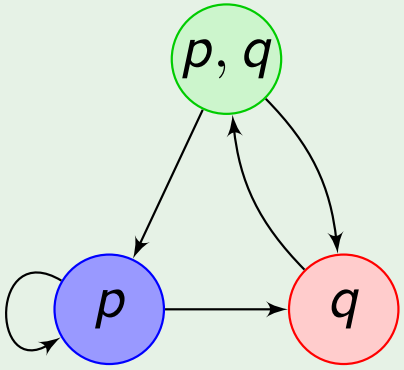
Verifying LTL properties

Theorem ([SC85])

LTL satisfiability and model checking are logspace-equivalent.

Proof (Sketch)

Encode behaviour of Kripke structure as an LTL formula:



$$\mathbf{G}\left[\begin{array}{ll} & \text{exactly one state-proposition in } \{\bigcirc, \bigcirc, \bigcirc\} \\ \wedge & (p \Leftrightarrow (\bigcirc \vee \bigcirc) \wedge (q \Leftrightarrow (\bigcirc \vee \bigcirc) \\ \wedge & [(\bigcirc \vee \bigcirc) \Rightarrow \mathbf{X}(\bigcirc \vee \bigcirc)[\wedge[\bigcirc \Rightarrow \mathbf{X}\bigcirc] \end{array}\right]$$

[SC85] Sistla, Clarke. The complexity of propositional linear temporal logics. J.ACM, 1985.

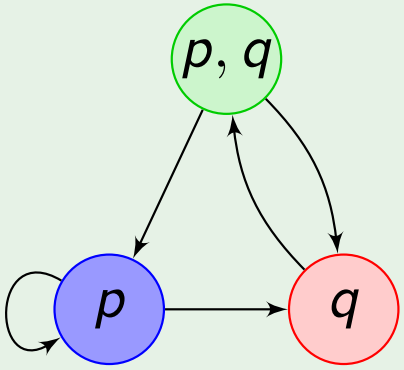
Verifying LTL properties

Theorem ([SC85])

LTL satisfiability and model checking are logspace-equivalent.

Proof (Sketch)

Encode behaviour of Kripke structure as an LTL formula:



$$\mathbf{G}\left[\begin{array}{ll} & \text{exactly one state-proposition in } \{\bigcirc, \bigcirc, \bigcirc\} \\ \wedge & (p \Leftrightarrow (\bigcirc \vee \bigcirc) \wedge (q \Leftrightarrow (\bigcirc \vee \bigcirc) \\ \wedge & [(\bigcirc \vee \bigcirc) \Rightarrow \mathbf{X}(\bigcirc \vee \bigcirc)[\wedge[\bigcirc \Rightarrow \mathbf{X}\bigcirc] \end{array}\right]$$

Lemma

$$\mathcal{M}, \bigcirc \not\models \varphi \iff \neg\varphi \wedge \bigcirc \wedge \Phi_{\mathcal{M}} \text{ is satisfiable.}$$

[SC85] Sistla, Clarke. The complexity of propositional linear temporal logics. J.ACM, 1985.

Verifying LTL properties

Theorem ([SC85])

LTL satisfiability can be solved in polynomial space.

[SC85] Sistla, Clarke. The complexity of propositional linear temporal logics. J.ACM, 1985.

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.

[SC85] Sistla, Clarke. The complexity of propositional linear temporal logics. J.ACM, 1985.

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

[SC85] Sistla, Clarke. The complexity of propositional linear temporal logics. J.ACM, 1985.

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

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

[SC85] Sistla, Clarke. The complexity of propositional linear temporal logics. J.ACM, 1985.

Verifying LTL properties

Theorem ([SC85])

LTL satisfiability can be solved in polynomial space.

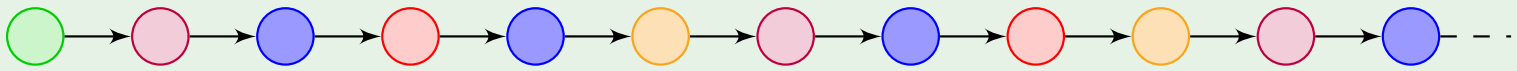
Proof (Sketch)

Lemma (Small-model property)

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

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

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



[SC85] Sistla, Clarke. The complexity of propositional linear temporal logics. J.ACM, 1985.

Verifying LTL properties

Theorem ([SC85])

LTL satisfiability can be solved in polynomial space.

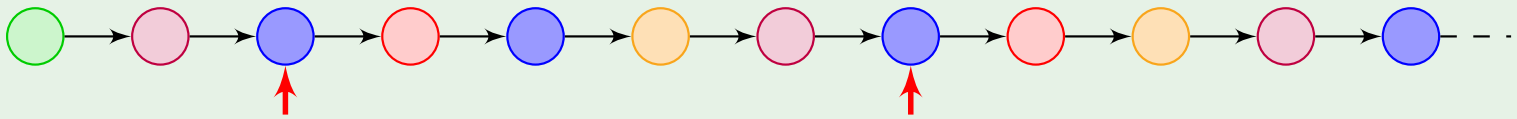
Proof (Sketch)

Lemma (Small-model property)

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

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

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



same subformulas hold at both positions;
all $\mathbf{U}$-formulas fulfilled inbetween

[SC85] Sistla, Clarke. The complexity of propositional linear temporal logics. J.ACM, 1985.

Verifying LTL properties

Theorem ([SC85])

LTL satisfiability can be solved in polynomial space.

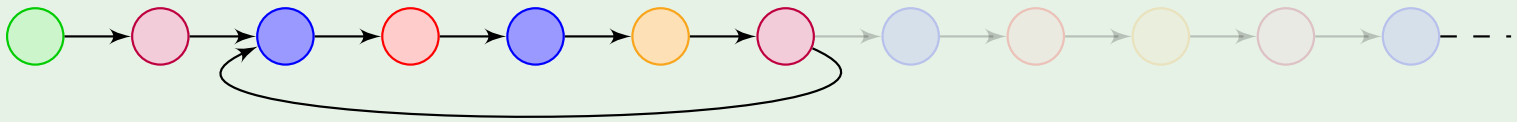
Proof (Sketch)

Lemma (Small-model property)

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

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

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



[SC85] Sistla, Clarke. The complexity of propositional linear temporal logics. J.ACM, 1985.

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.

[SC85] Sistla, Clarke. The complexity of propositional linear temporal logics. J.ACM, 1985.

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

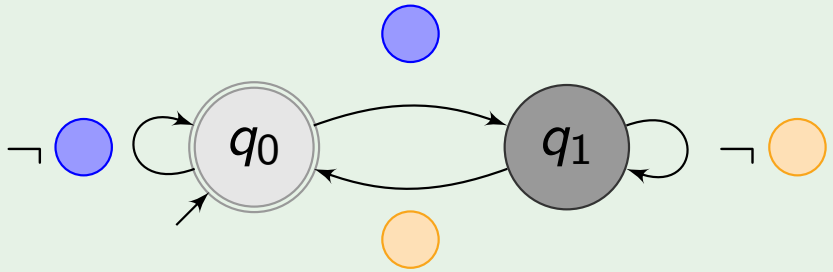
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

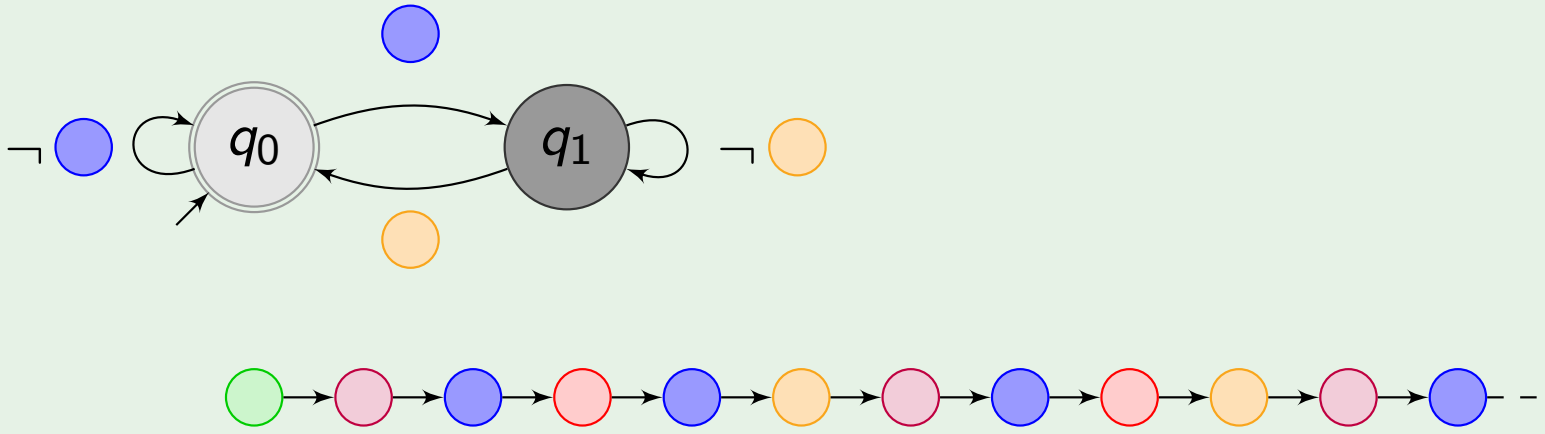
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

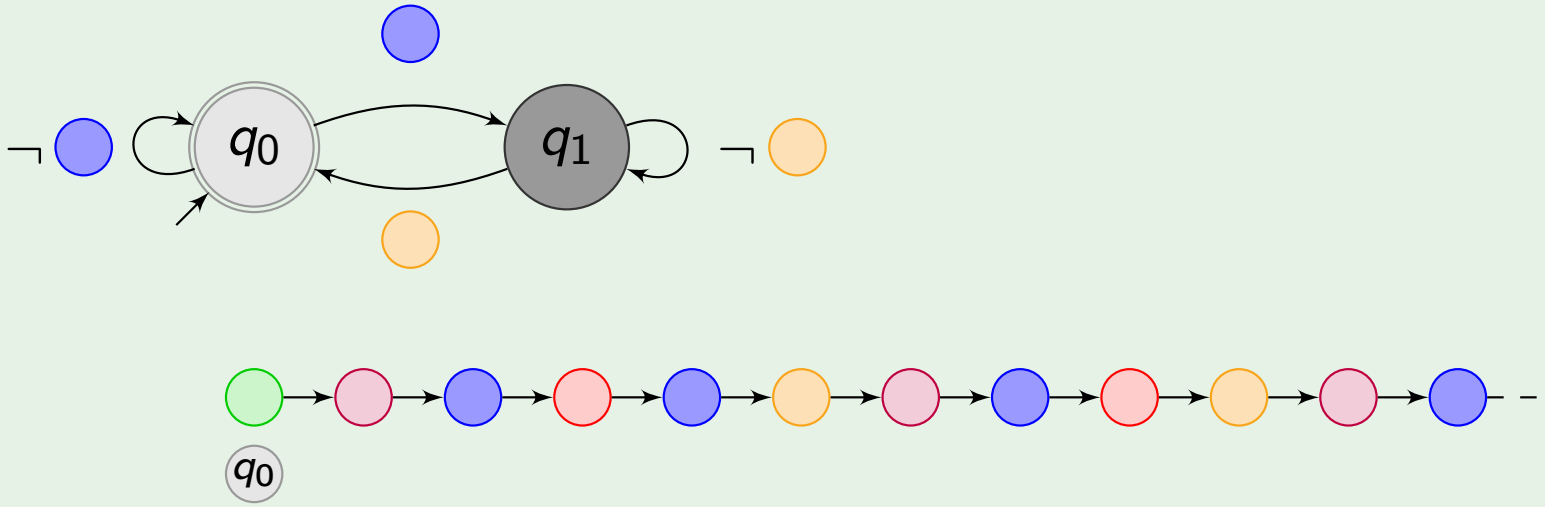
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

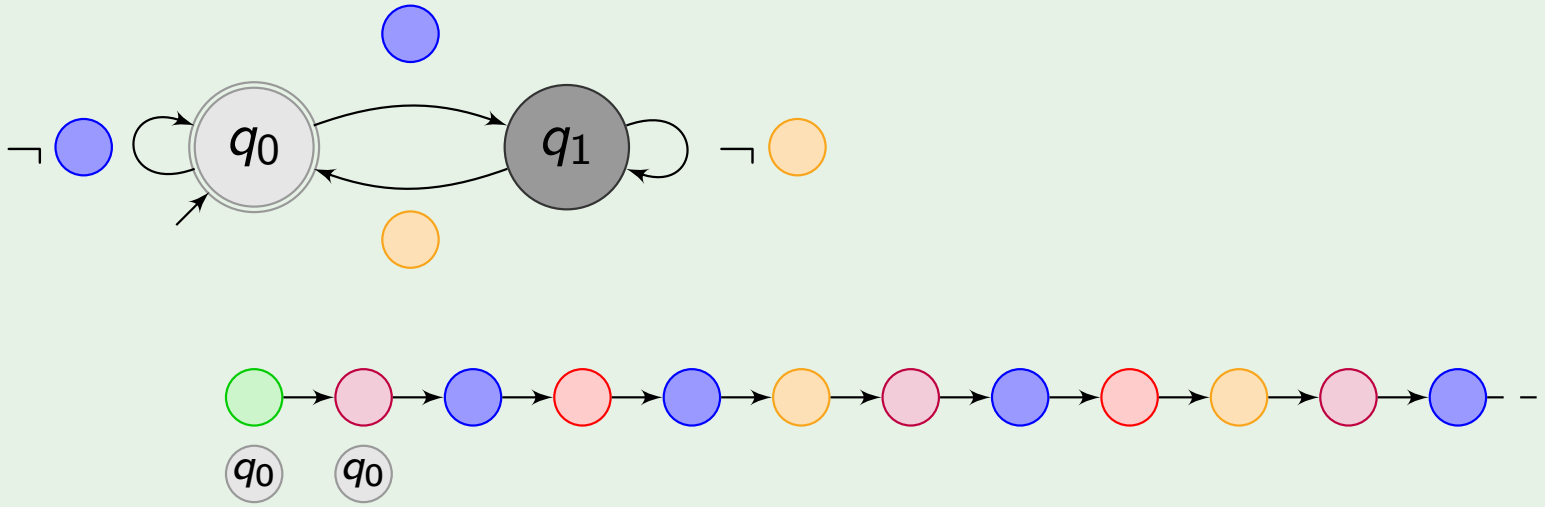
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

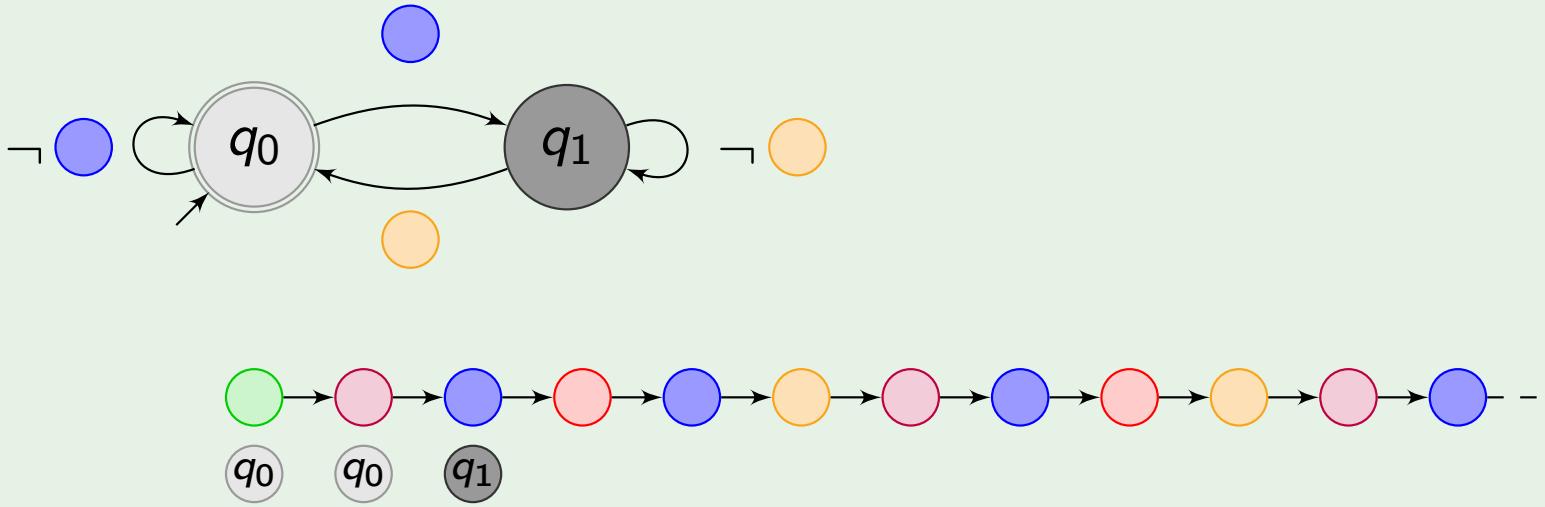
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

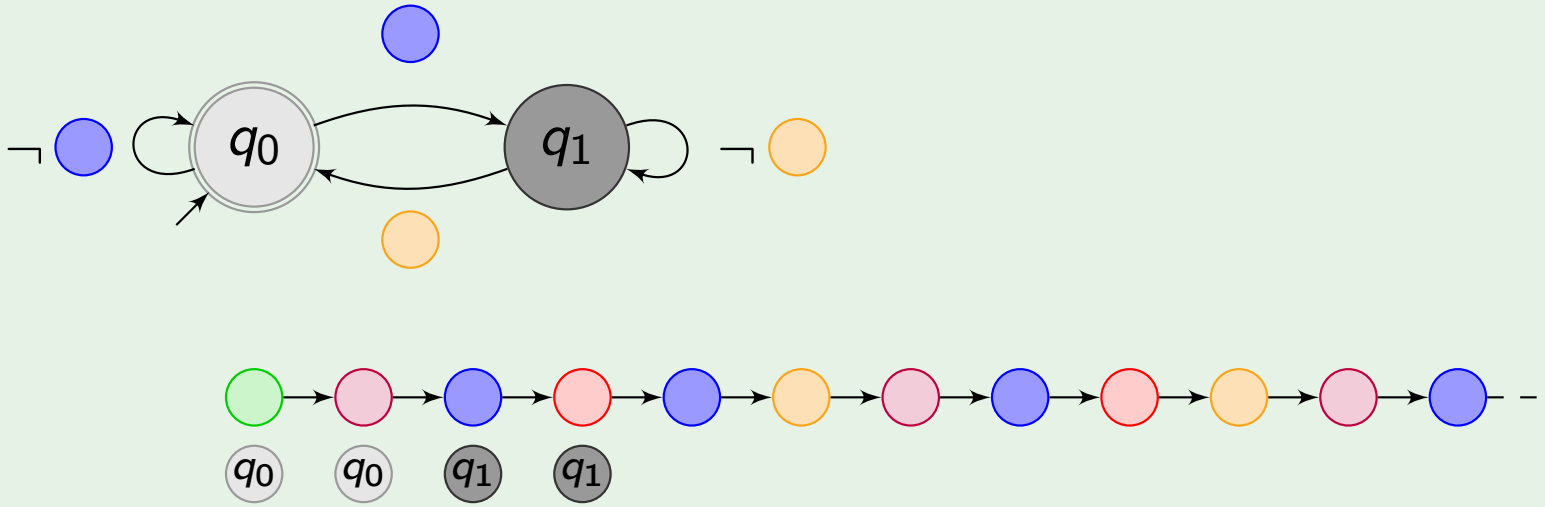
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

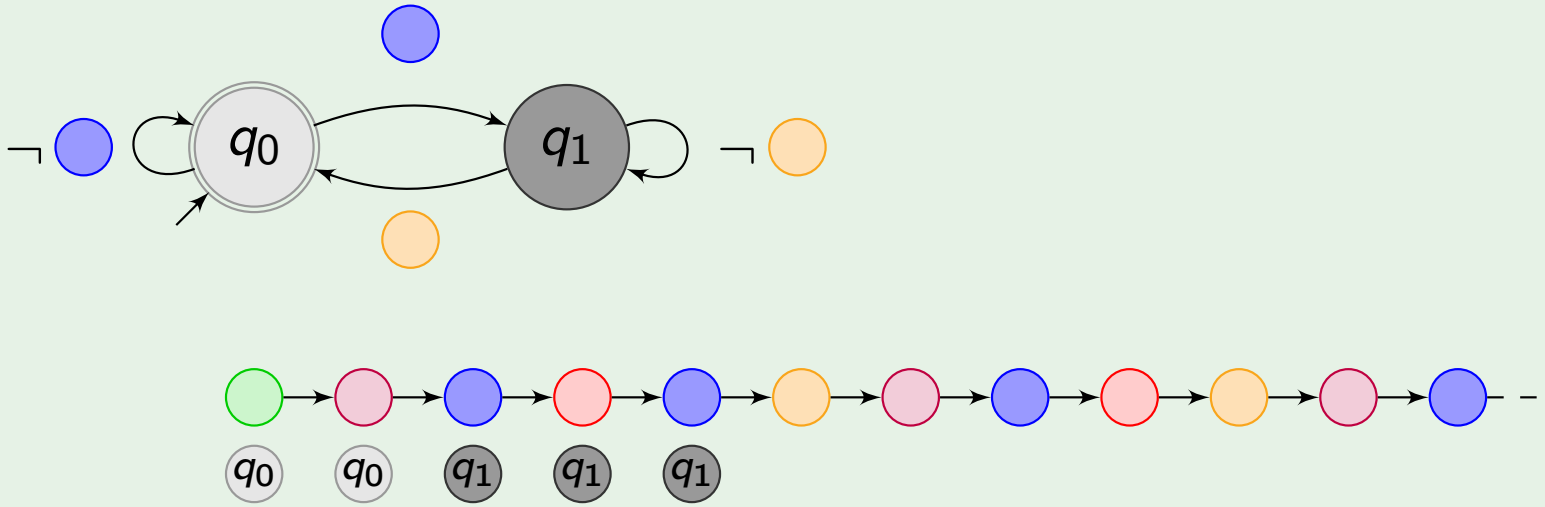
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

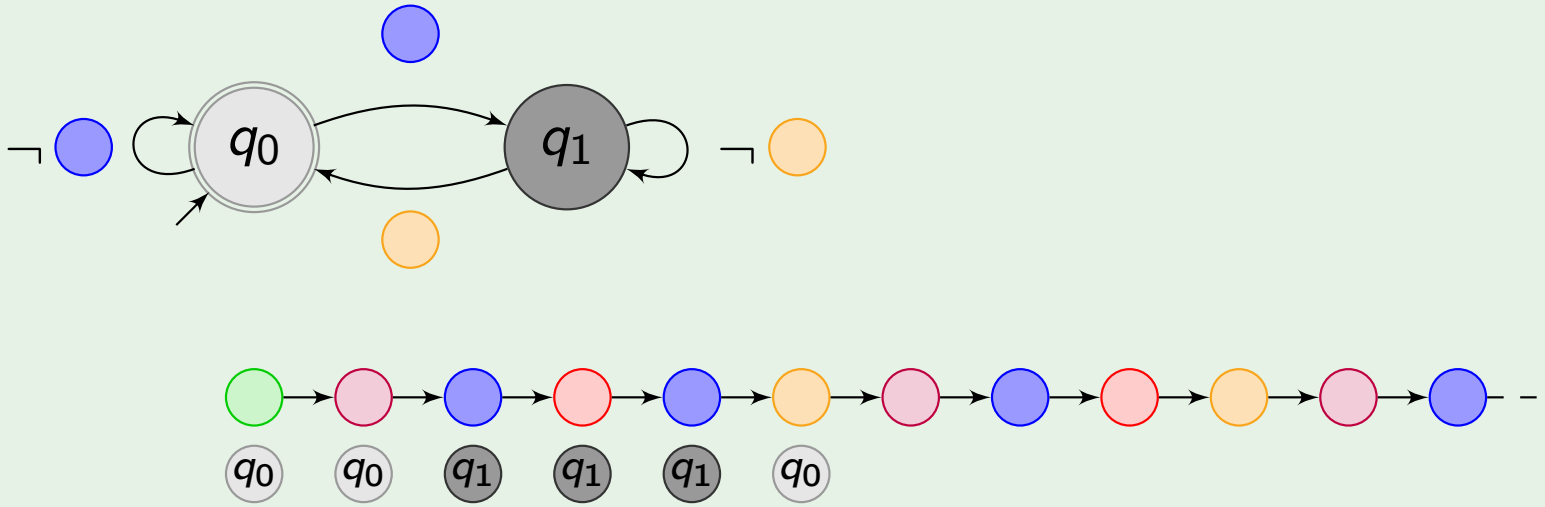
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

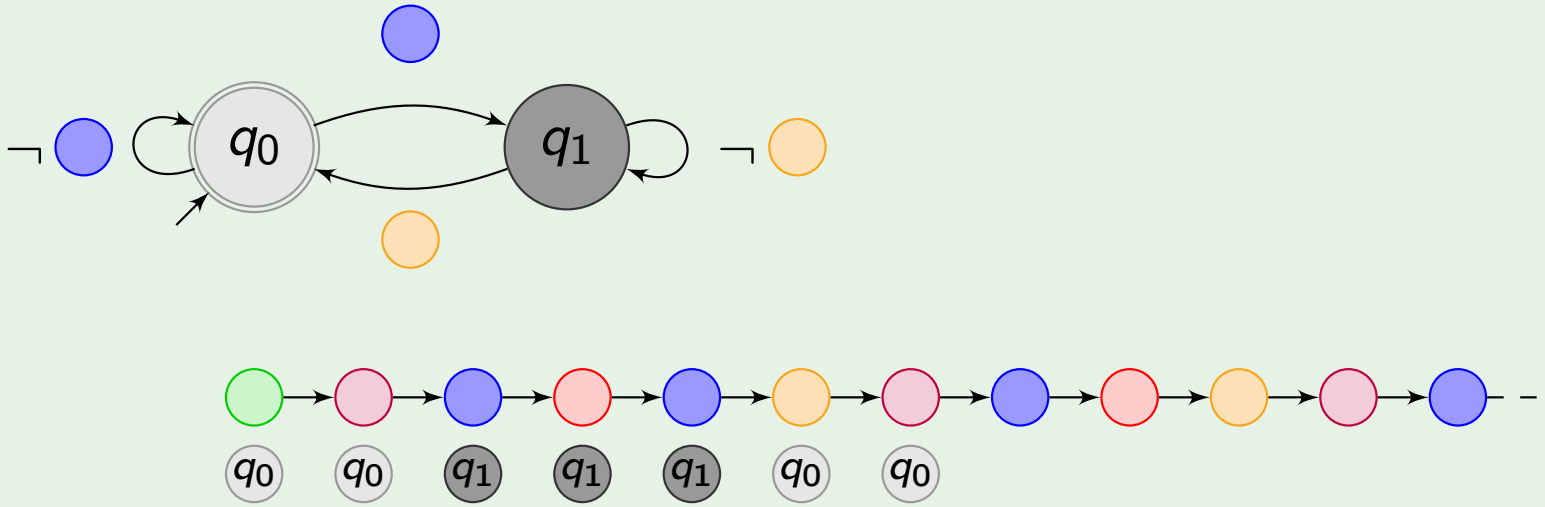
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

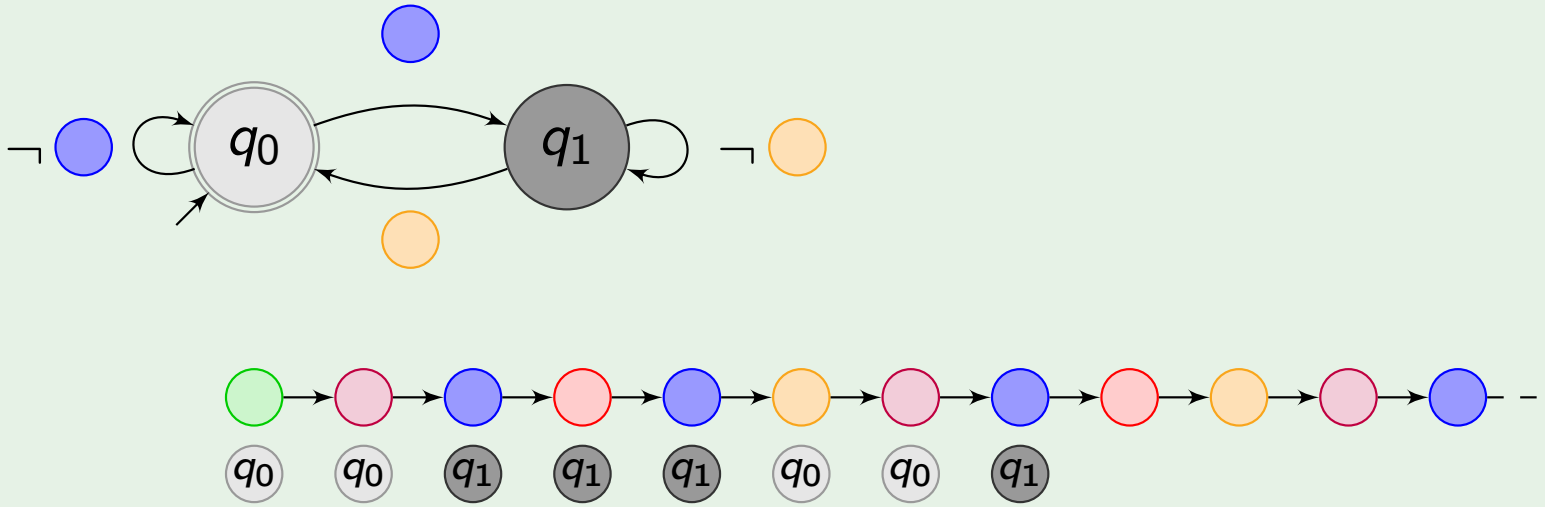
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

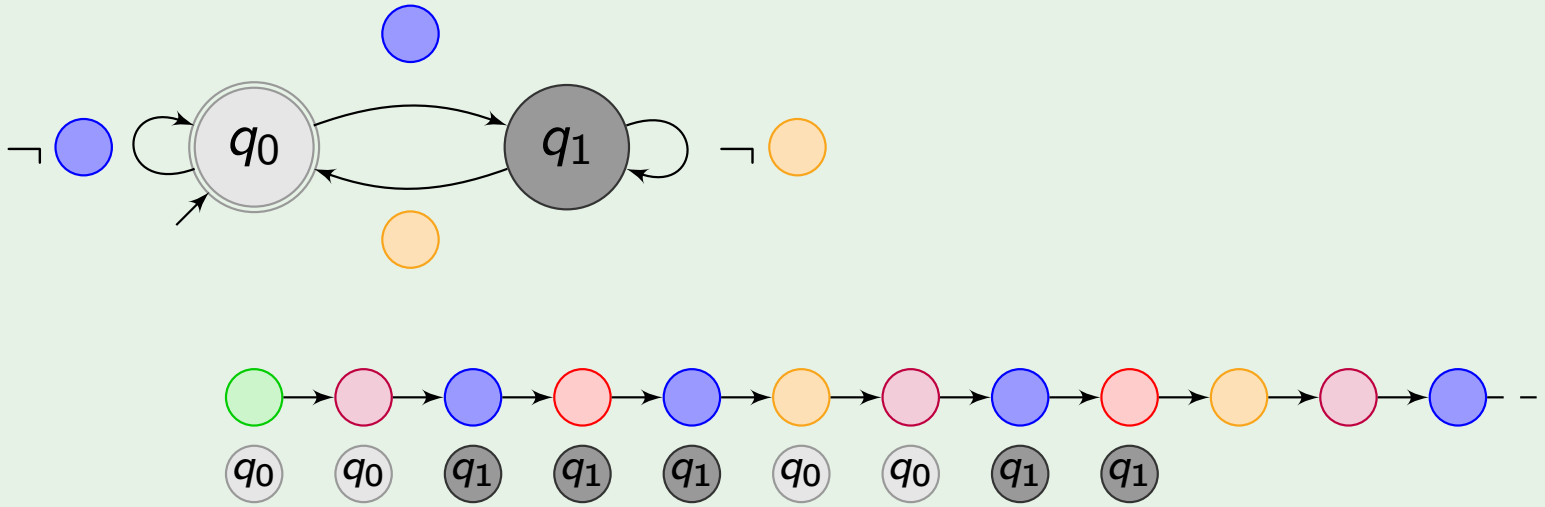
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

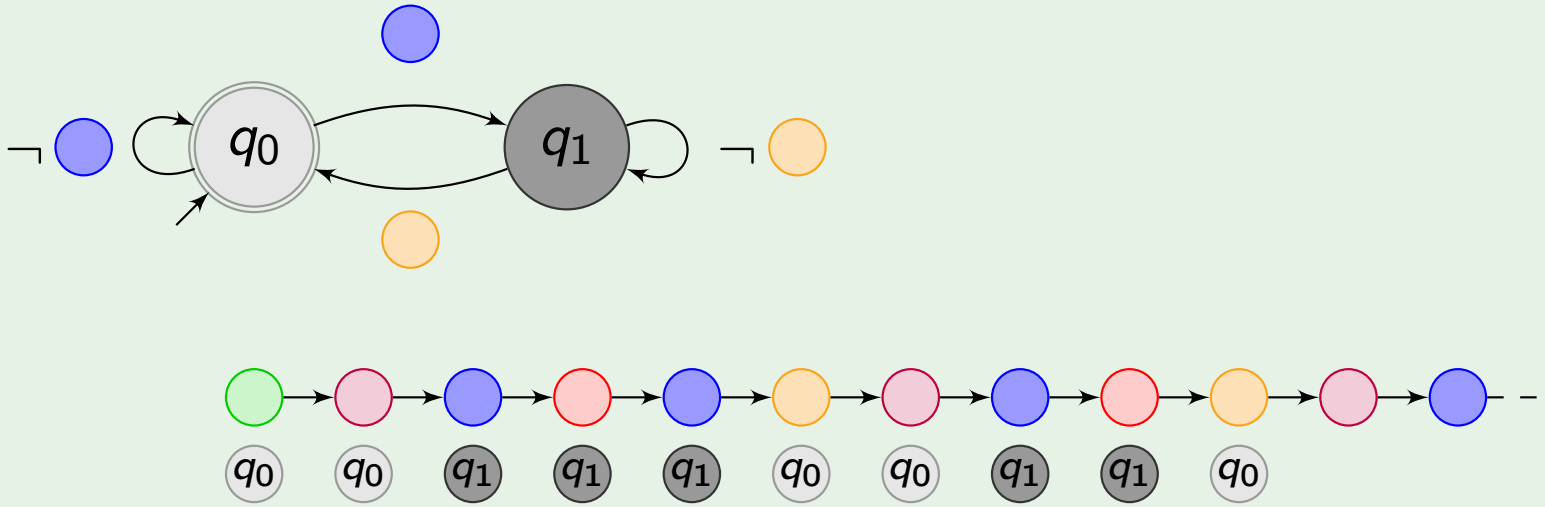
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

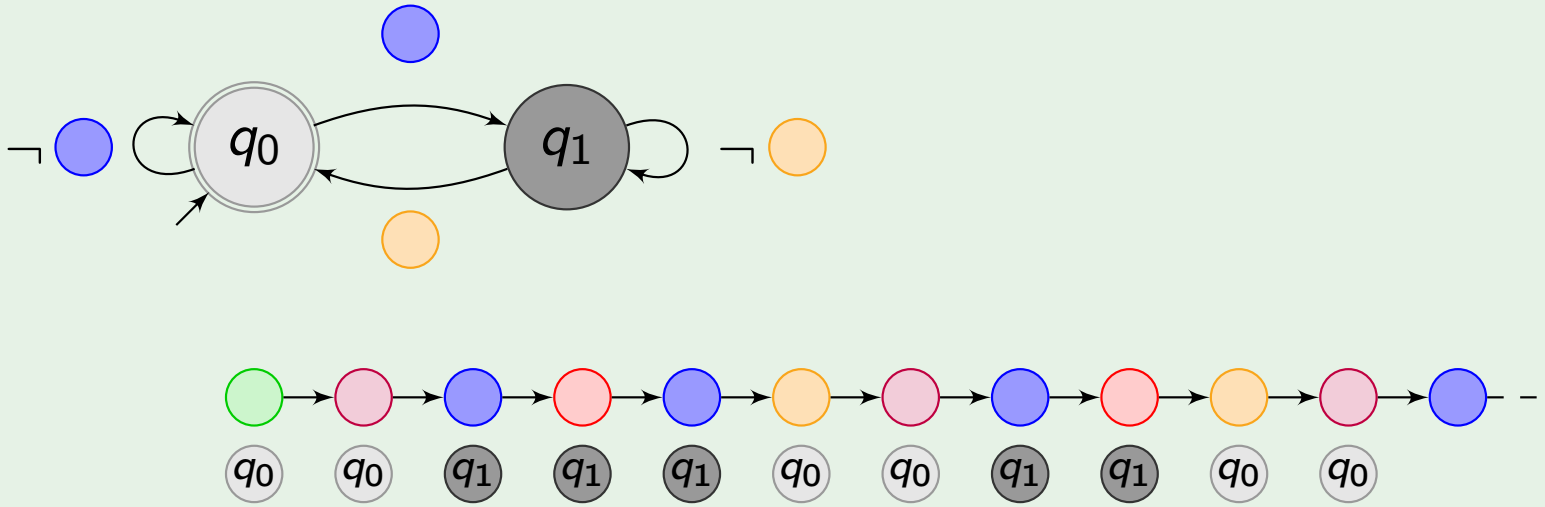
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

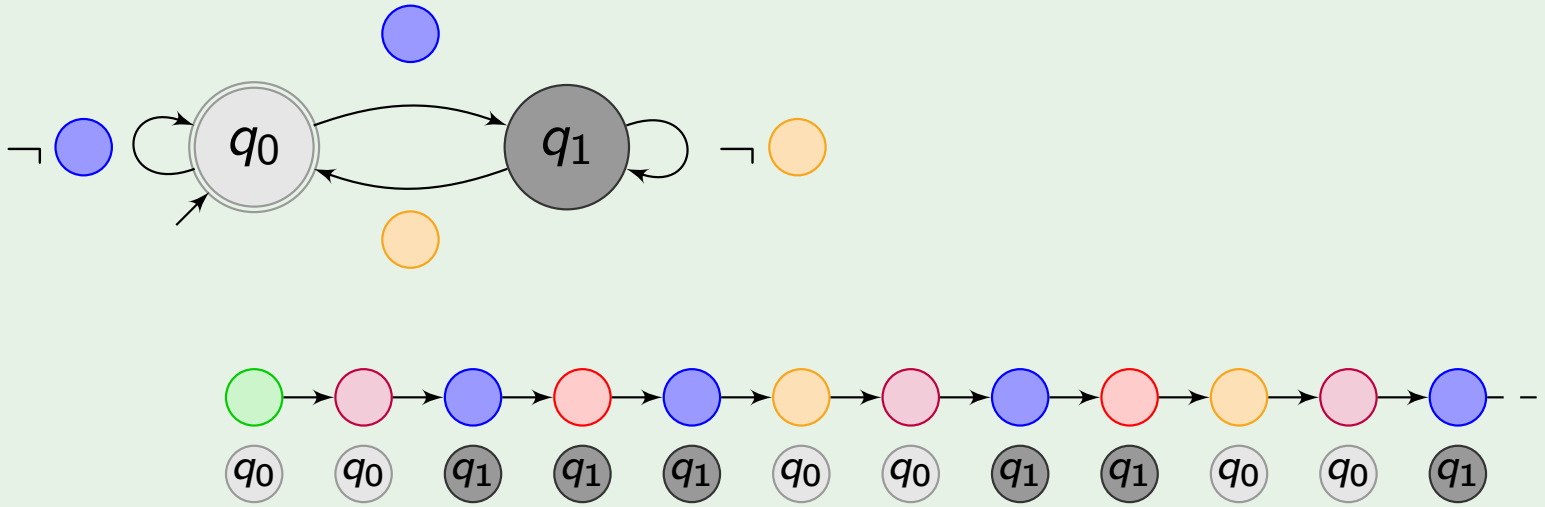
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

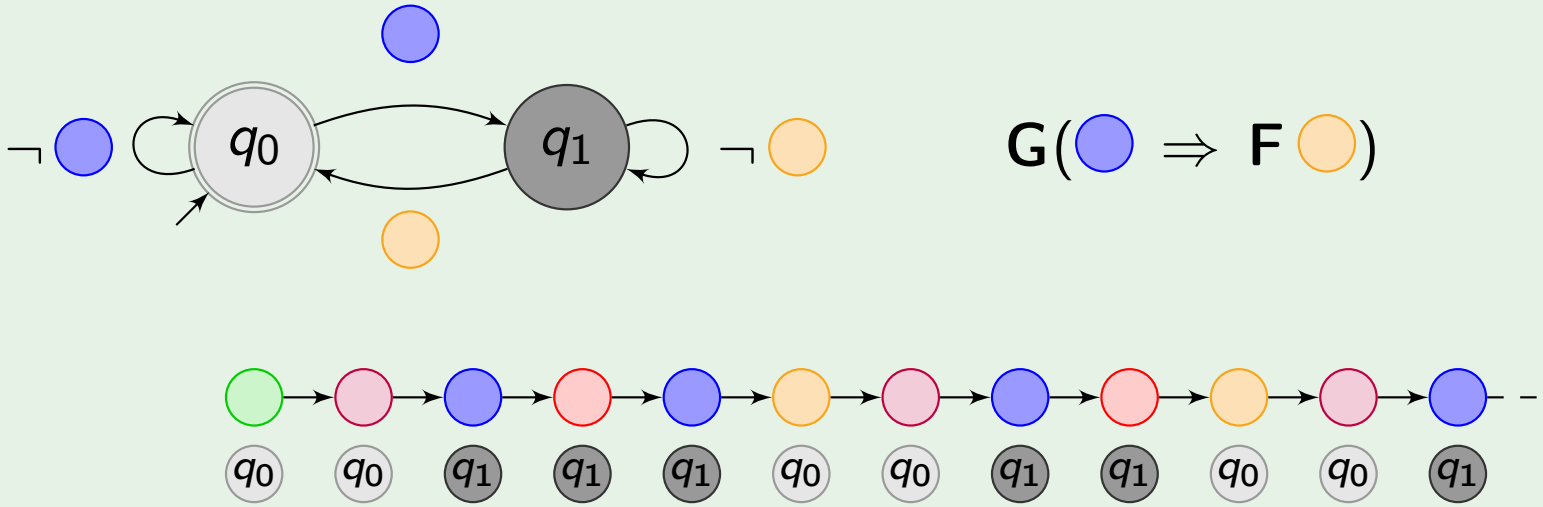
Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Proof (Sketch)

[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Proof (Sketch)

- use a slightly extended notion of subformula:

$$\bigcirc \ \mathbf{U} \ \bigcirc \quad \rightsquigarrow \quad \bigcirc \vee (\bigcirc \wedge \mathbf{X}(\bigcirc \ \mathbf{U} \ \bigcirc)).$$

[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Proof (Sketch)

- use a slightly extended notion of subformula:

$$\bigcirc\ \mathbf{U}\ \bigcirc \quad \rightsquigarrow \quad \bigcirc \vee (\bigcirc \wedge \mathbf{X}(\bigcirc\ \mathbf{U}\ \bigcirc)).$$

- states of $\mathcal{A}_\varphi$ = maximal consistent sets of subformulas;
- transition $S \to S'$ whenever for all $\mathbf{X}\,\psi \in S$, it holds $\psi \in S'$;

[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Proof (Sketch)

- use a slightly extended notion of subformula:

$$\bigcirc\ \mathbf{U}\ \bigcirc \quad \rightsquigarrow \quad \bigcirc \vee (\bigcirc \wedge \mathbf{X}(\bigcirc\ \mathbf{U}\ \bigcirc)).$$

- states of $\mathcal{A}_\varphi$ = maximal consistent sets of subformulas;
- transition $S \to S'$ 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)

[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

Verifying LTL properties – automata-theoretic approach

Theorem ([VW86])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Theorem ([VW86])

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

[VW86] Vardi, Wolper. An automata-theoretic approach to automatic program verification. LICS, 1986.

Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

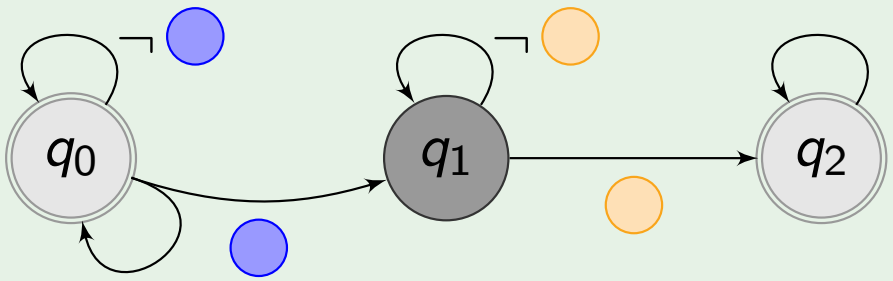
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

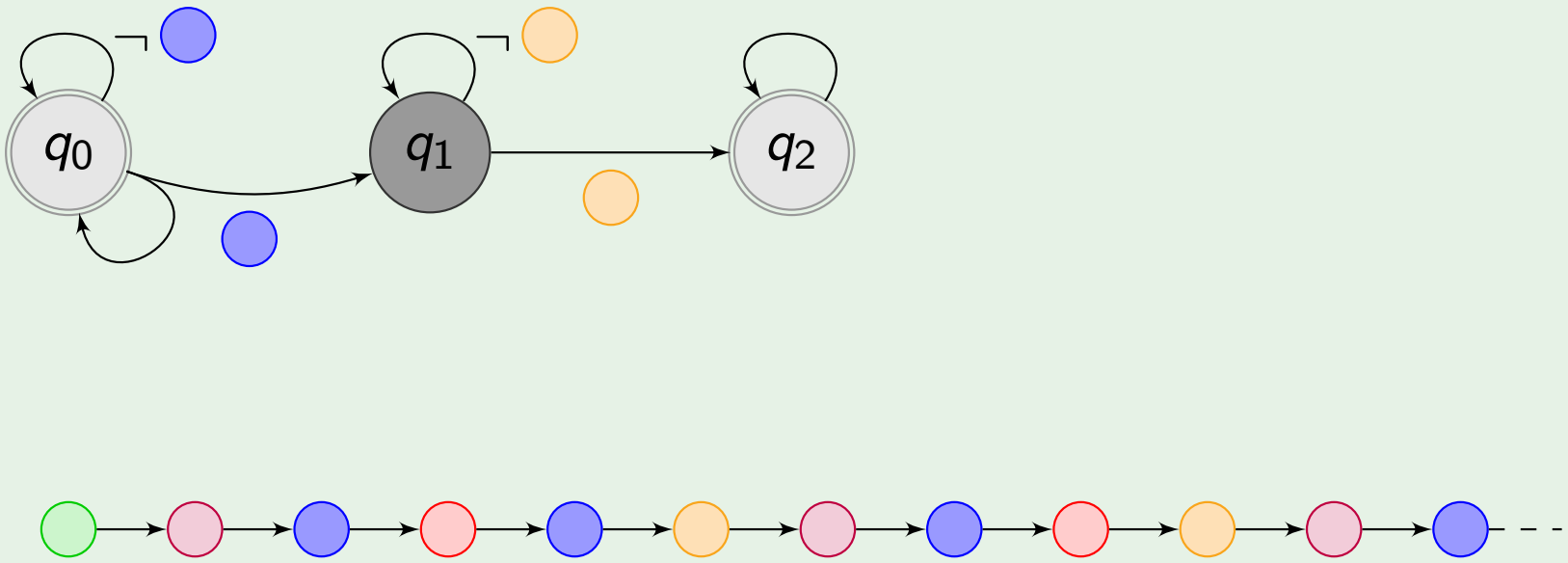
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

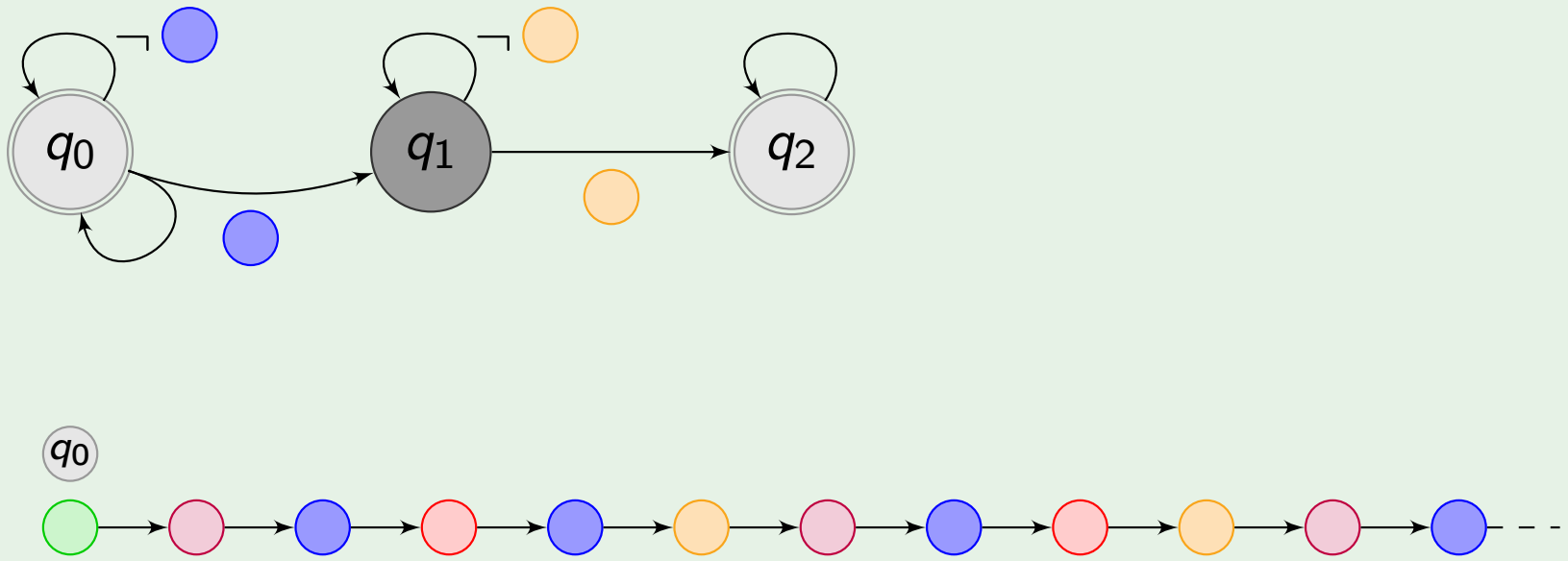
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

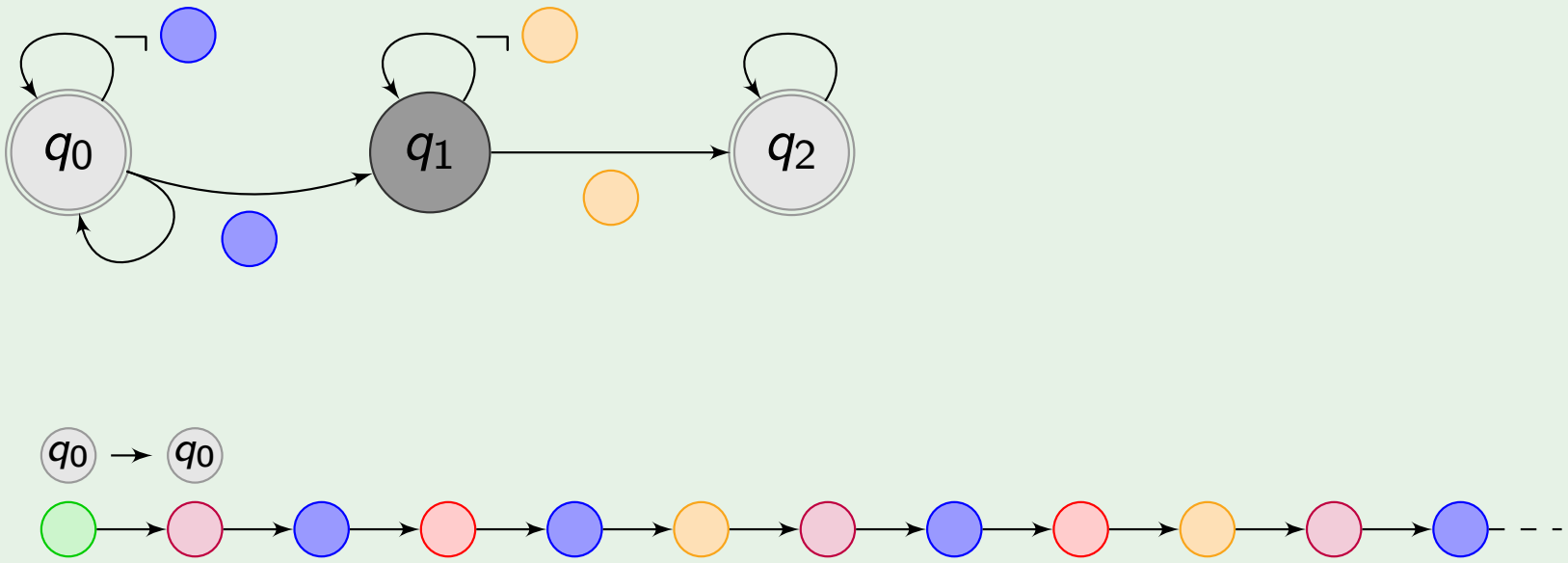
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

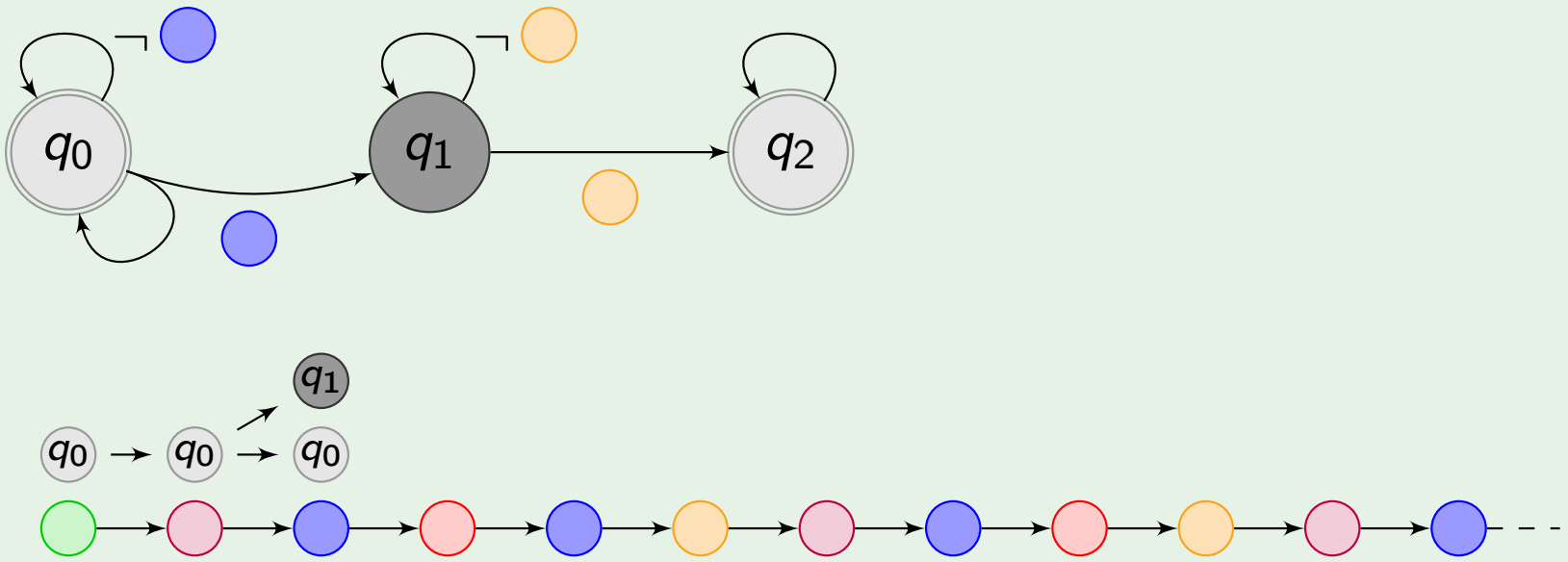
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

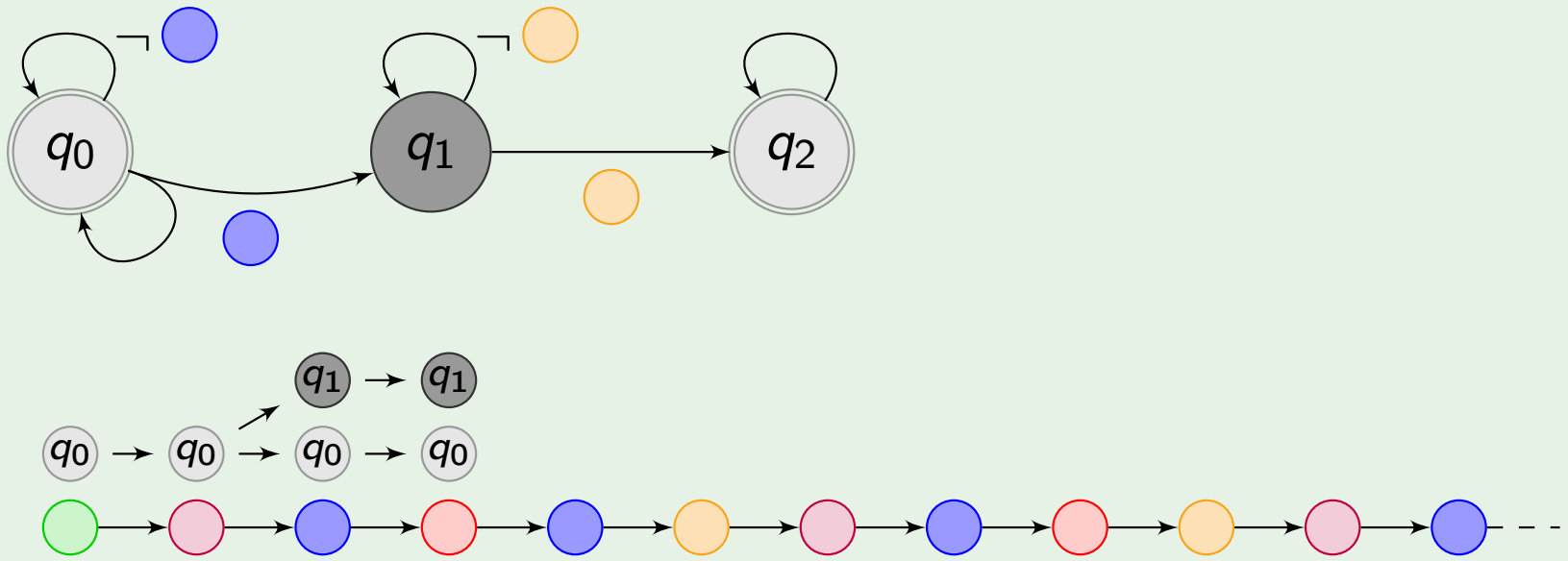
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

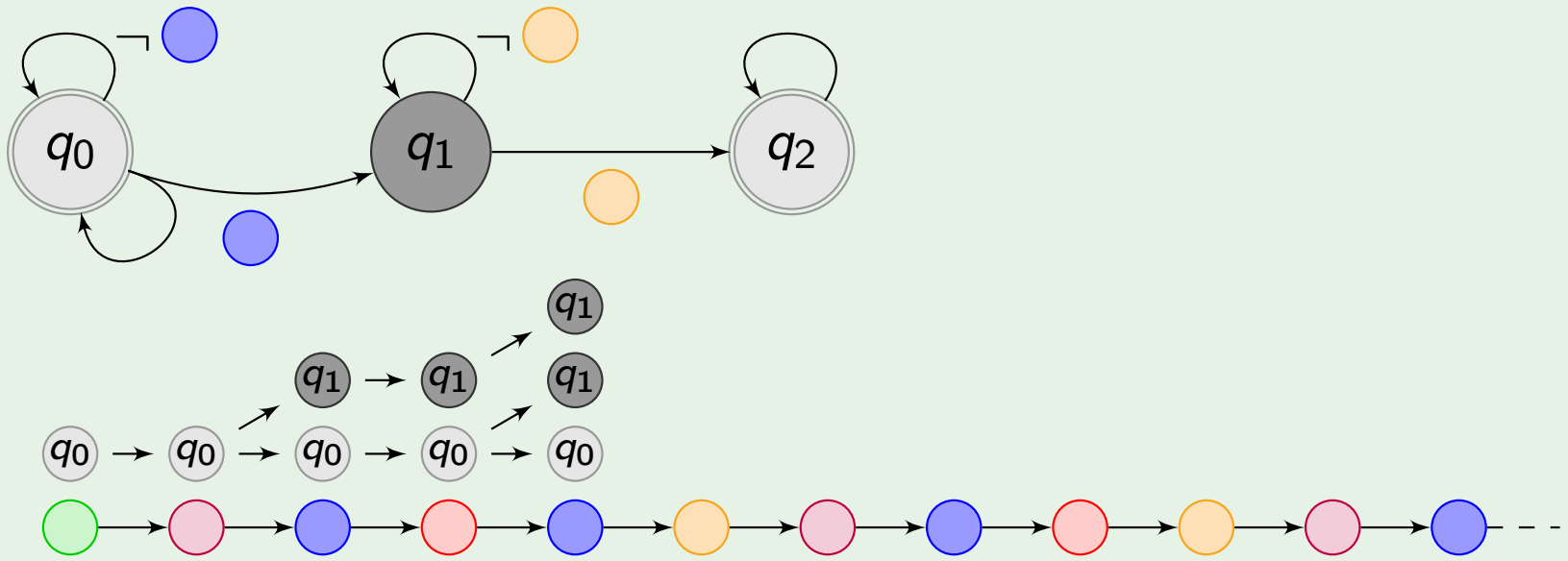
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

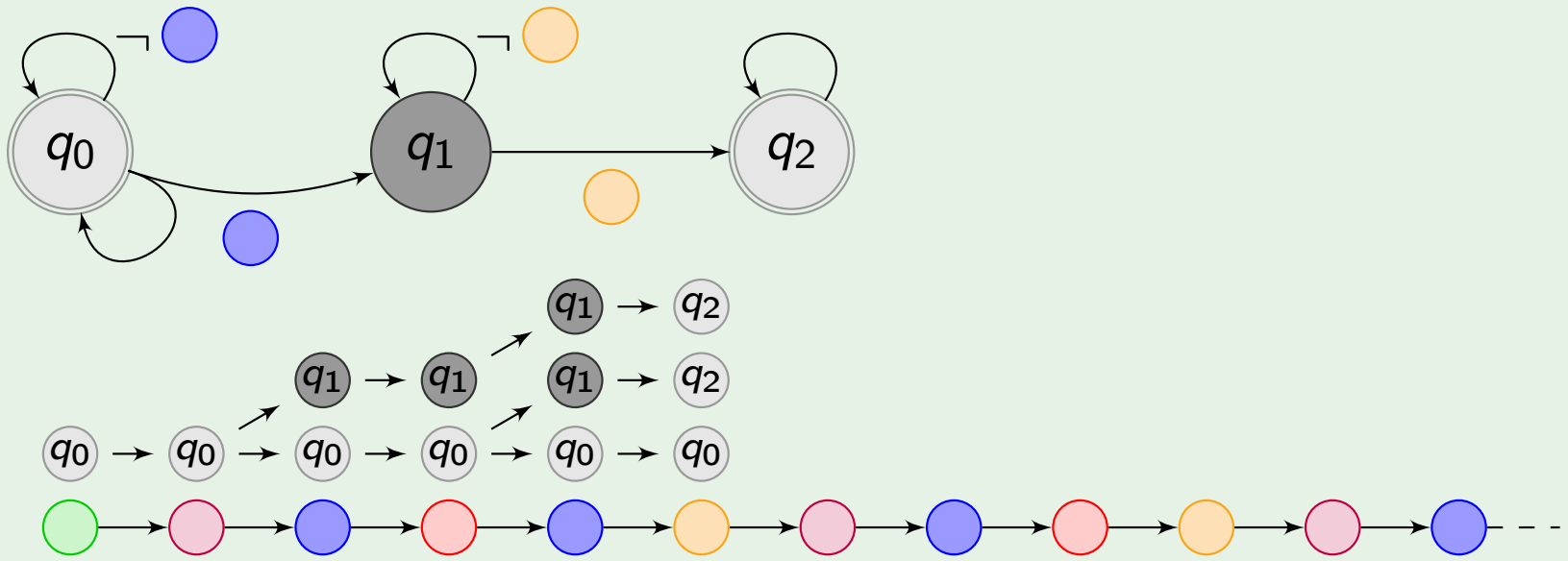
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

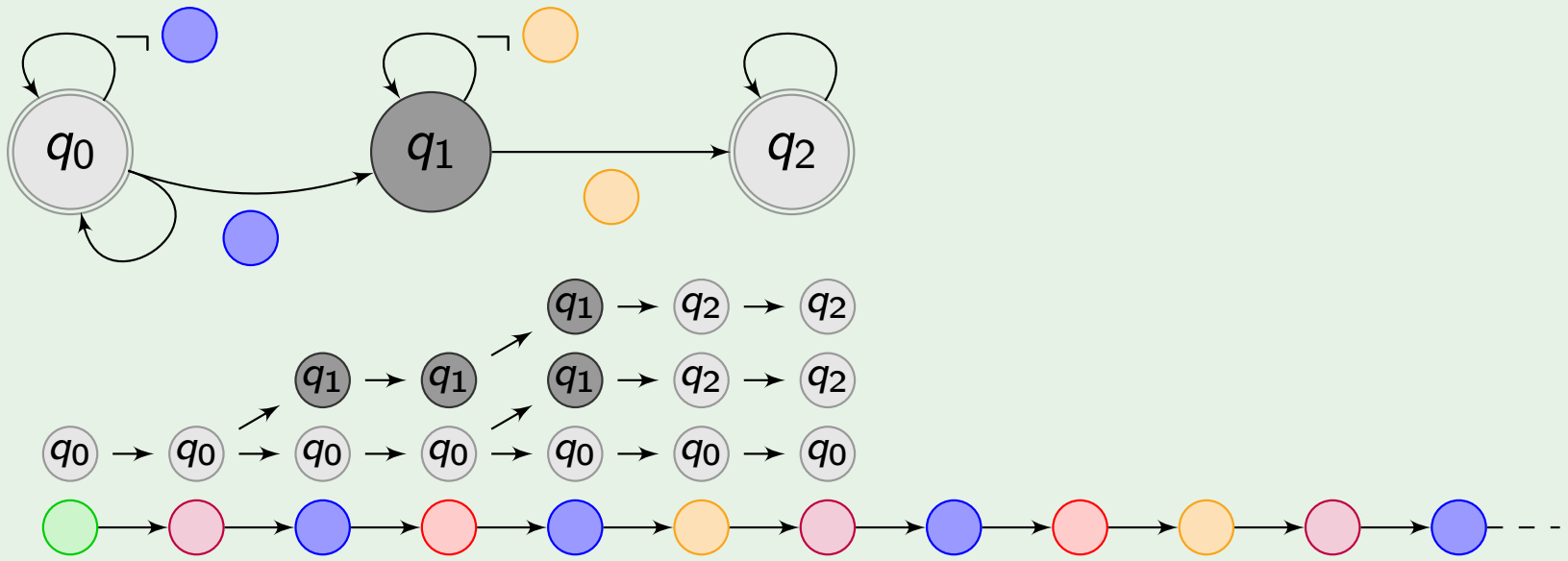
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

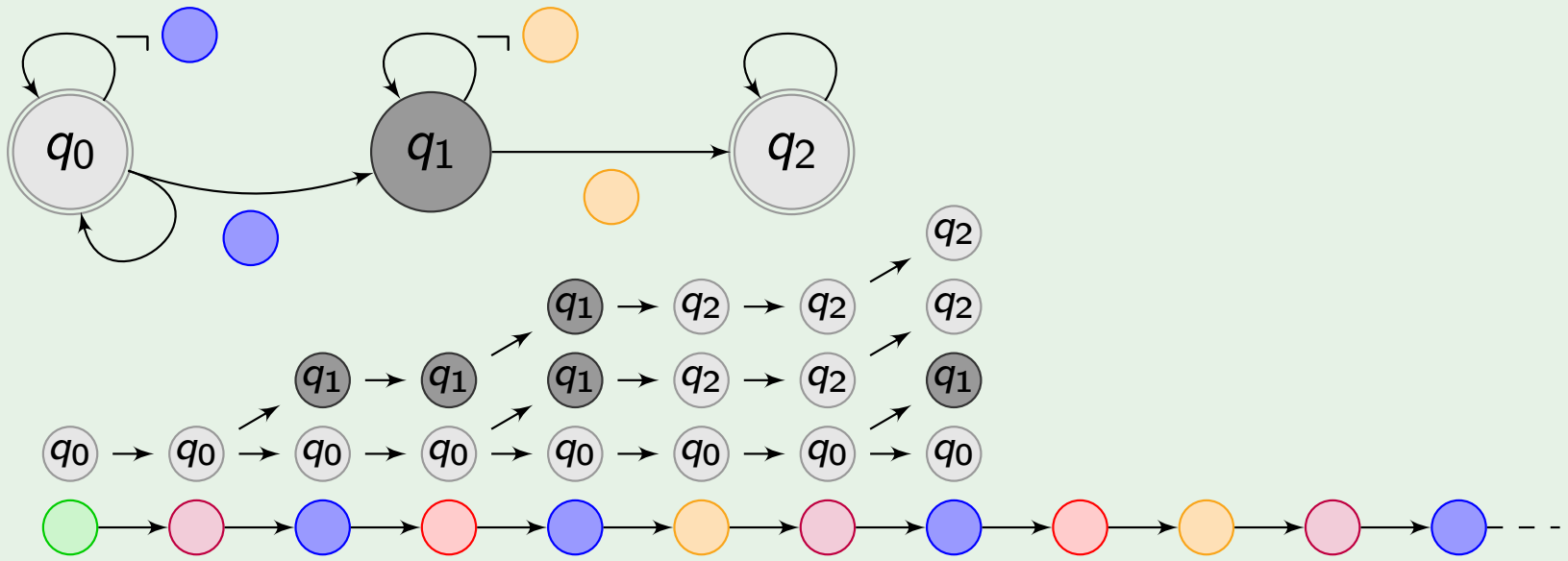
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

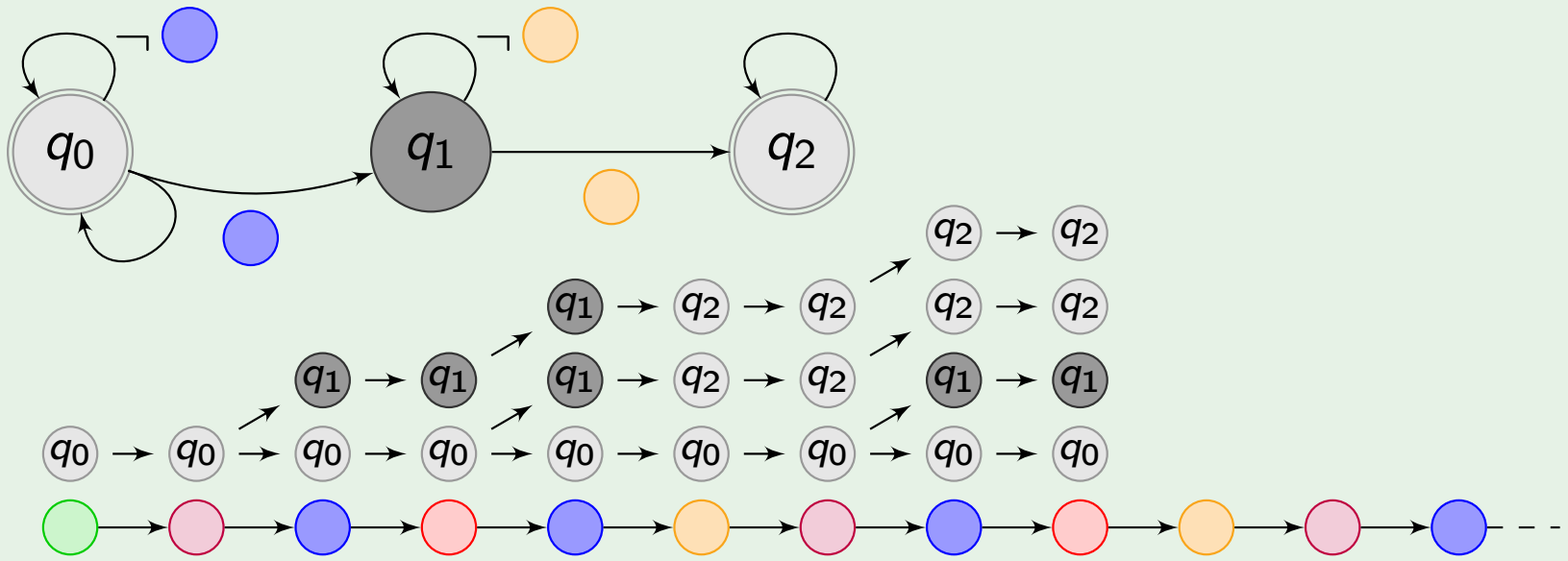
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

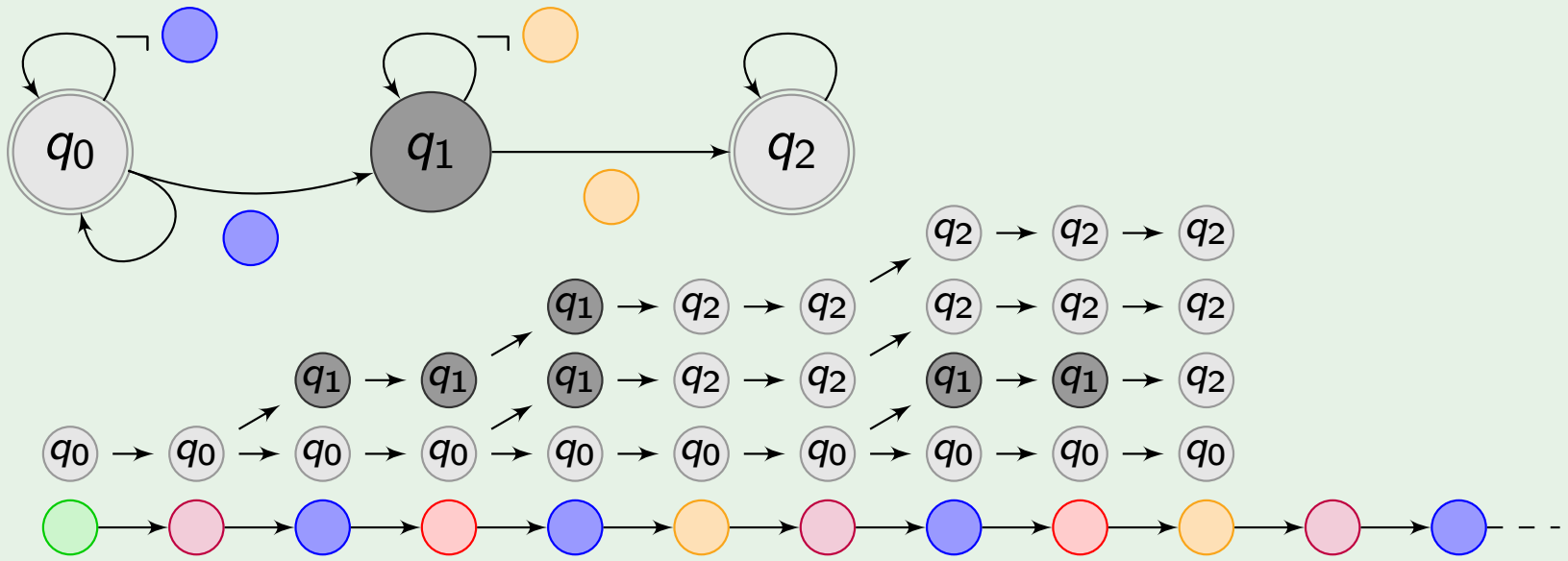
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

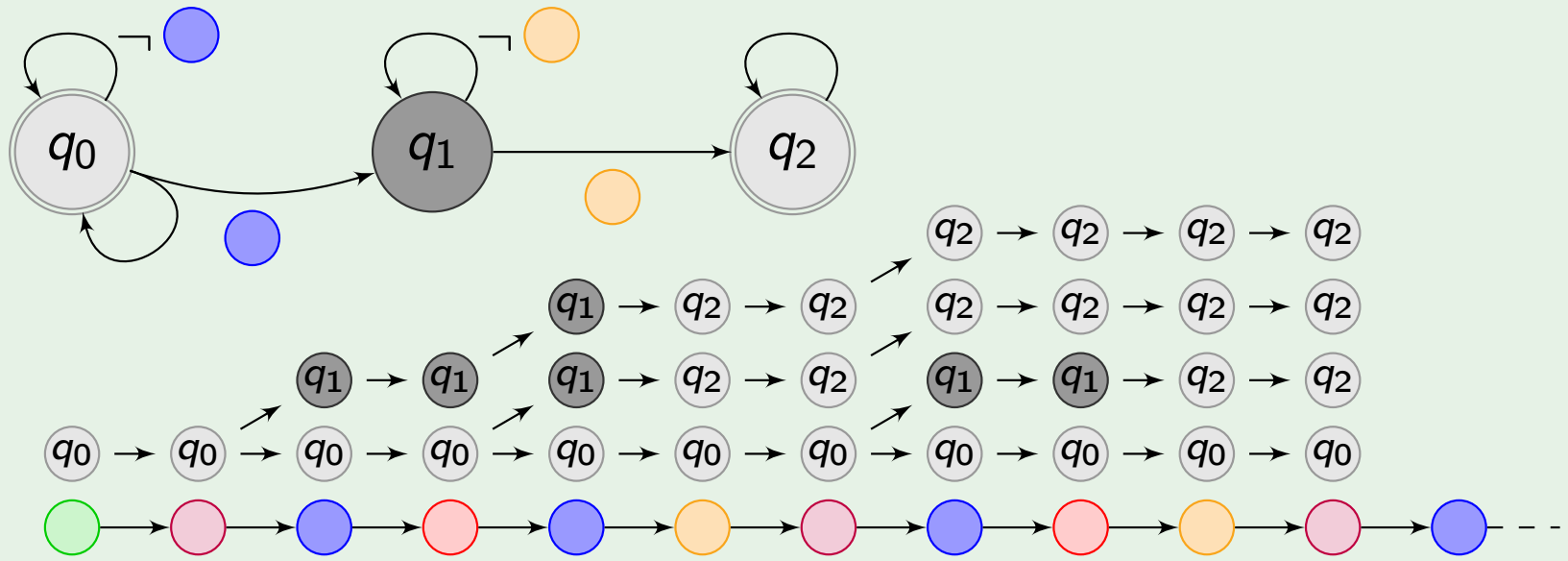
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

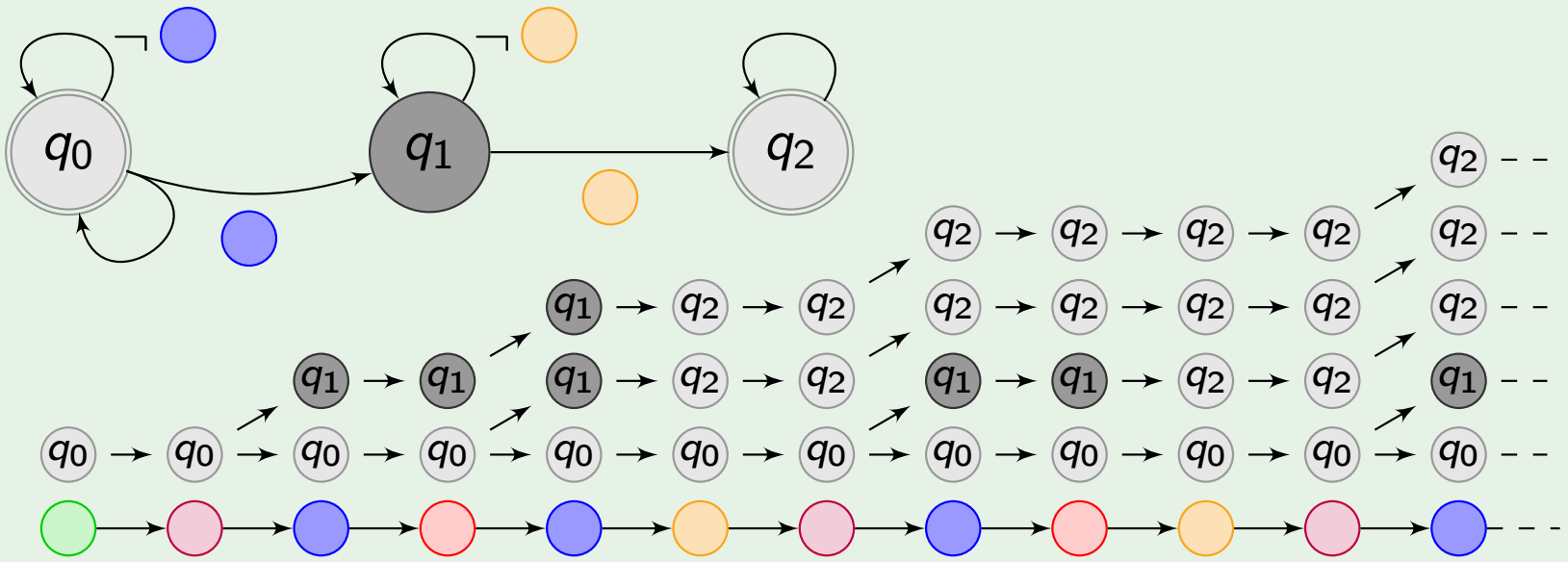
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

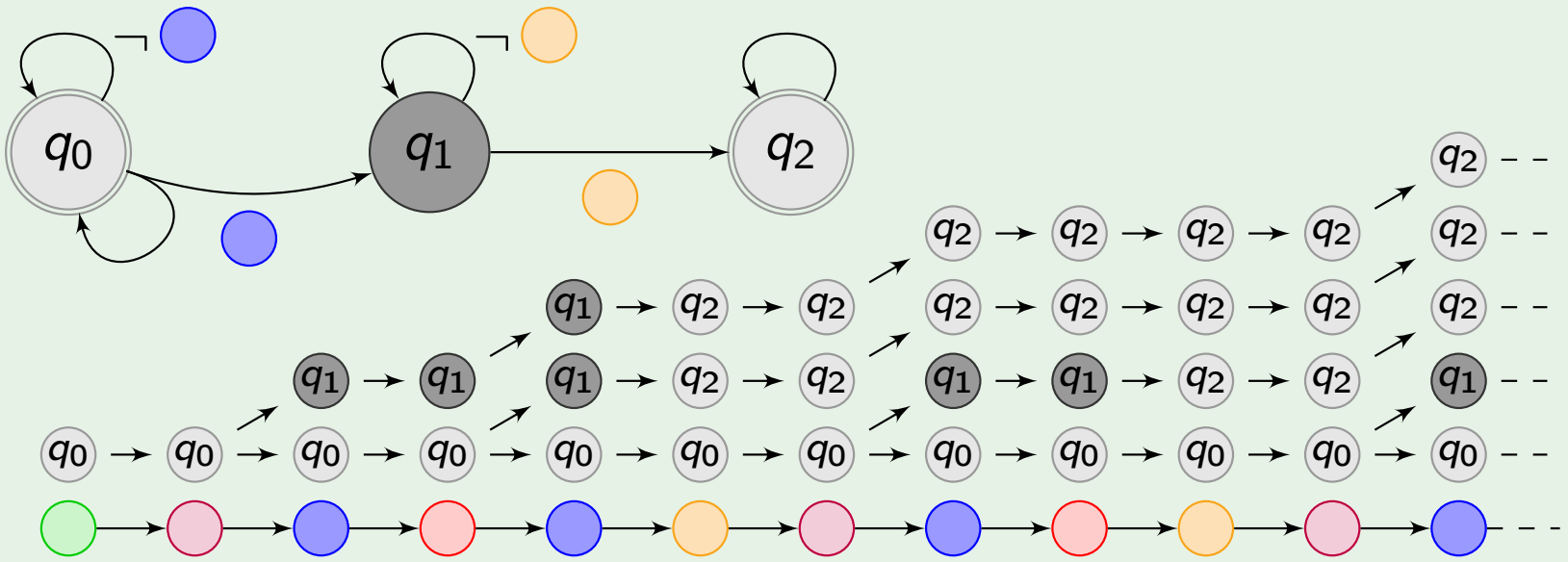
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

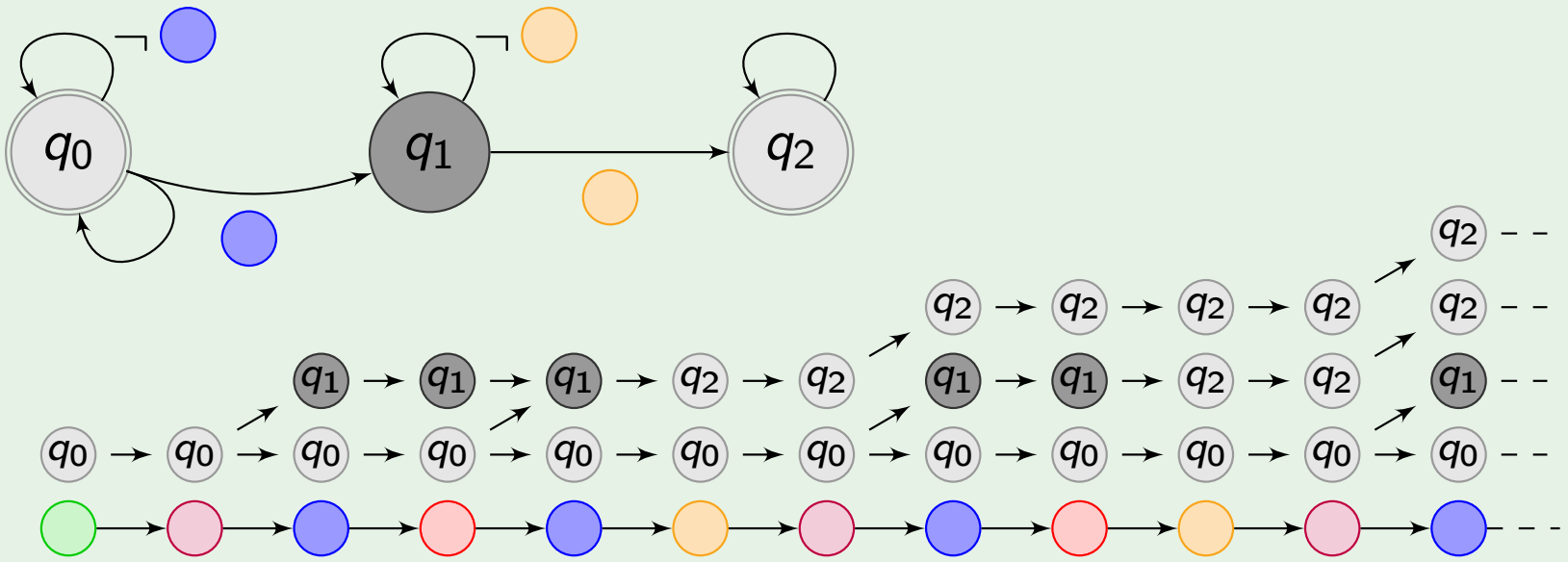
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

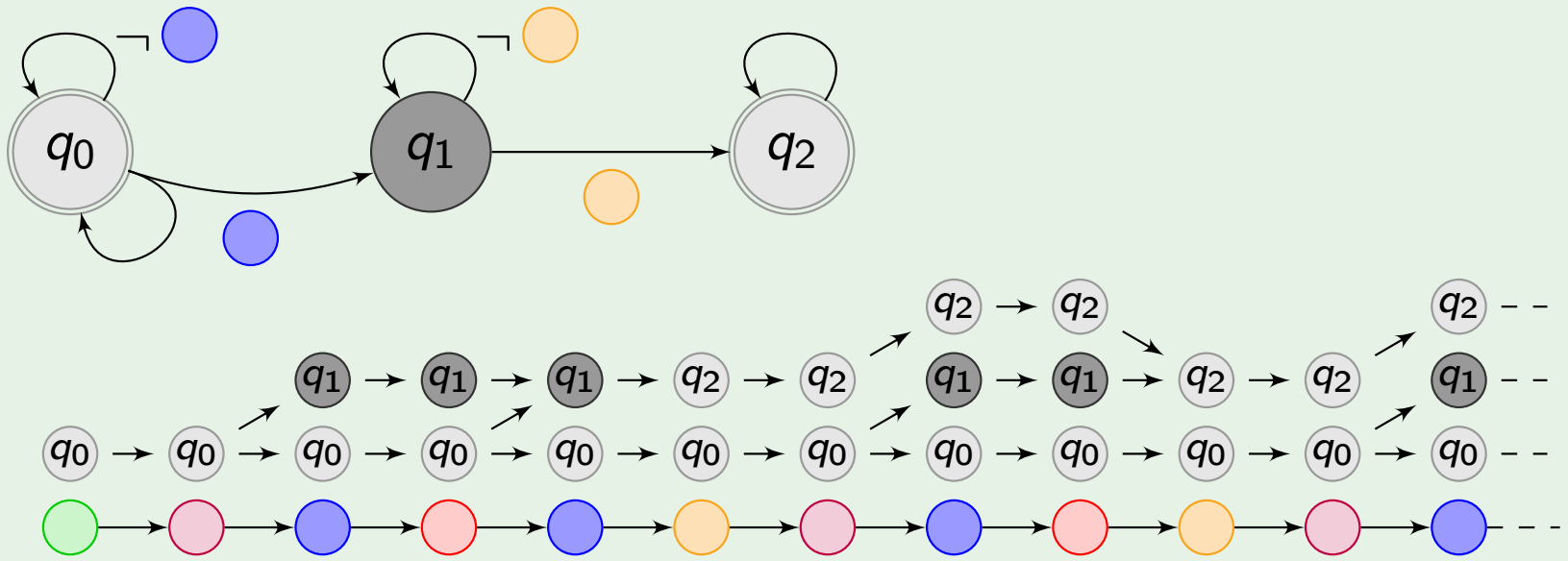
Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Example



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

[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

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.
[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Proof (Sketch)

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

$$\delta(\psi_1 \ \mathbf{U} \ \psi_2, \bigcirc) = \delta(\psi_2, \bigcirc) \vee \left[\delta(\psi_1, \bigcirc) \wedge \psi_1 \ \mathbf{U} \ \psi_2\right].$$

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

Verifying LTL properties – using alternating automata

Theorem ([MSS88,Var94])

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

$$\forall w. \qquad w \models \varphi \iff w \in Lang(\mathcal{A}_\varphi).$$

Proof (Sketch)

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

$$\delta(\psi_1 \ \mathbf{U} \ \psi_2, \bigcirc) = \delta(\psi_2, \bigcirc) \vee \left[\delta(\psi_1, \bigcirc) \wedge \psi_1 \ \mathbf{U} \ \psi_2\right].$$

- acceptance condition: Büchi condition over non-until states.

[MSS88] Muller, Saoudi, Schupp. Weak alternating automata give a simple explanation ... LICS, 1988.
[Var94] Vardi. Nontraditional applications of automata theory. TACS, 1994.

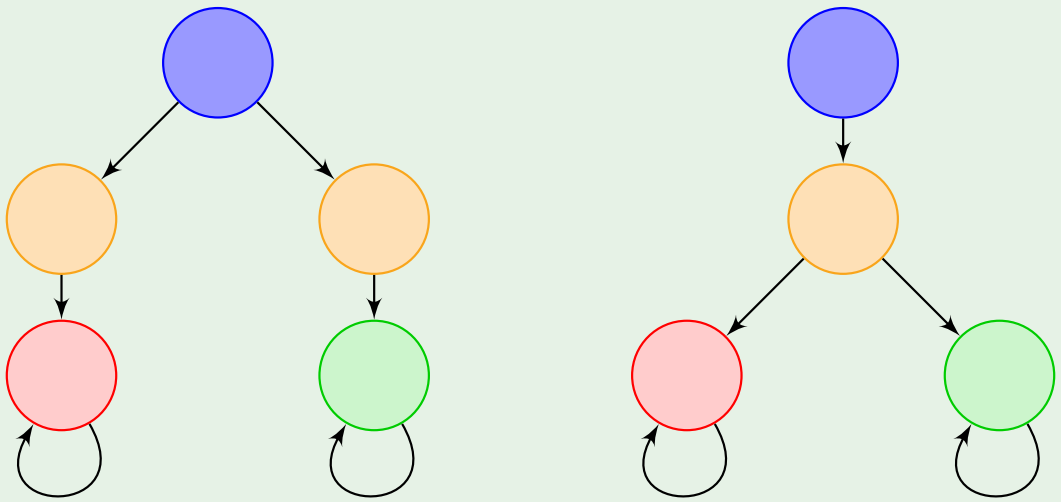
Behavioural equivalence for LTL

When are two Kripke structures indistinguishable by LTL?

Behavioural equivalence for LTL

When are two Kripke structures indistinguishable by LTL?

Example



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.

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.

Proof (Sketch)

Lemma

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

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

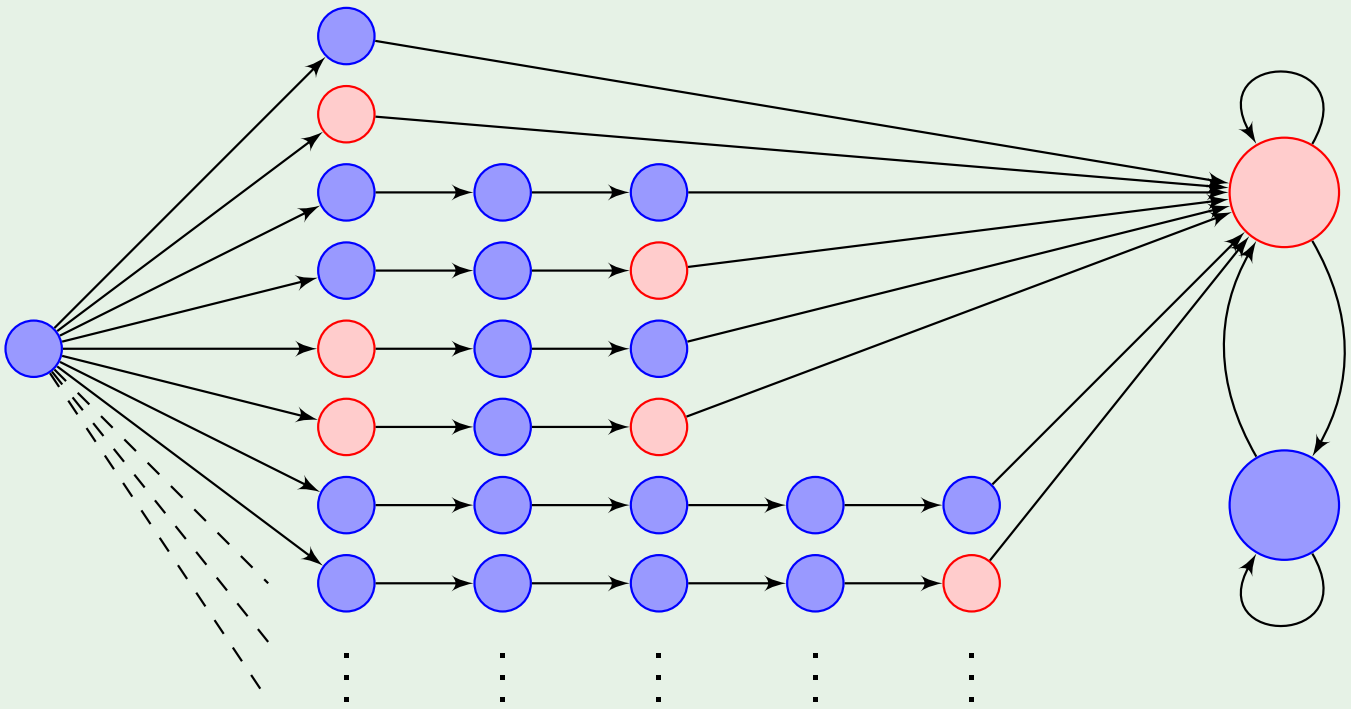
(for finitely branching Kripke structures $\mathcal{M}$ and $\mathcal{N}$).

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

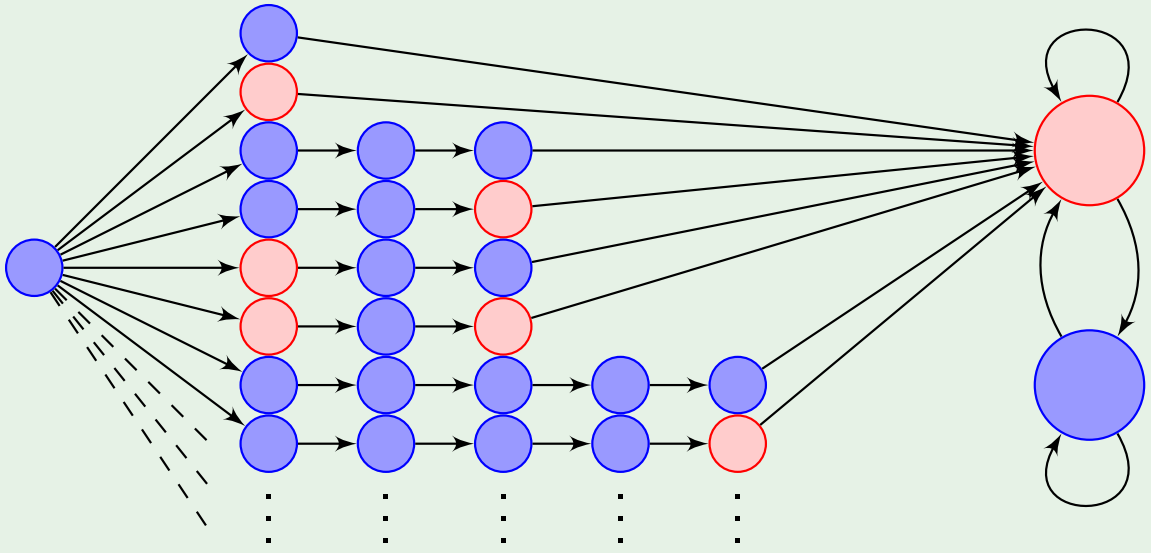


Behavioural equivalence for LTL

Theorem

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

Counter-example

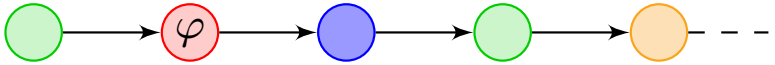


$$w \in \mathsf{Traces}(\mathcal{M}) \iff \exists i > 0.\ w(2i) =$$

Computation-Tree Logic (CTL*)

Computation-Tree Logic (CTL*)

- **atomic propositions:** ●, ●, ...
- **boolean combinators:** $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, ...
- **temporal modalities:**

$\mathbf{X}\,\varphi$ 

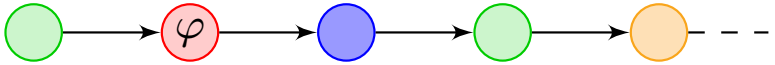
"next φ"

$\varphi\ \mathbf{U}\ \psi$ 

"φ until ψ"

Computation-Tree Logic (CTL*)

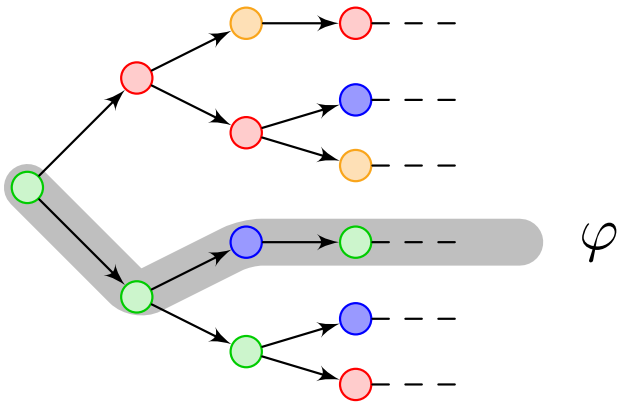
- **atomic propositions:** ◯, ◯, ...
- **boolean combinators:** $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, ...
- **temporal modalities:**

$\mathbf{X}\,\varphi$  "next φ"

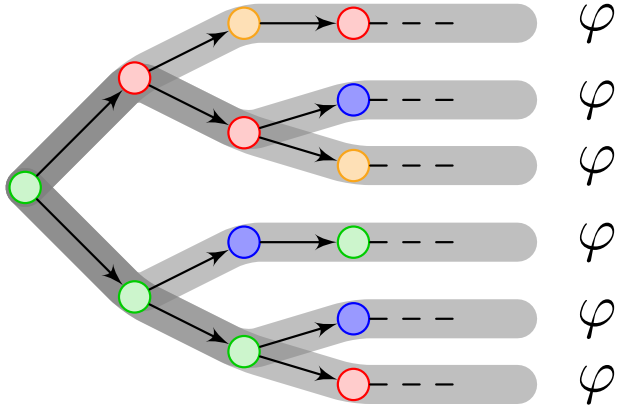
$\varphi\ \mathbf{U}\ \psi$  "φ until ψ"

- **path quantifiers:**

$\mathbf{E}\varphi$



$\mathbf{A}\varphi$



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.

A F floor_1 *the cabin eventually reaches the first floor*

Examples of CTL formulas

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

$\mathbf{A}\,\mathbf{F}\,\mathrm{floor}_1$	*the cabin eventually reaches the first floor*
$\mathbf{A}\,\mathbf{G}(\mathrm{op}_1 \Rightarrow \mathrm{serv}_1)$	*if the door is open, the cabin is present*

Examples of CTL formulas

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

$\mathbf{A}\,\mathbf{F}\,\mathrm{floor}_1$	*the cabin eventually reaches the first floor*
$\mathbf{A}\,\mathbf{G}(\mathrm{op}_1 \Rightarrow \mathrm{serv}_1)$	*if the door is open, the cabin is present*
$\mathbf{A}\,\mathbf{G}(\mathrm{req}_0 \Rightarrow \mathbf{A}\,\mathbf{F}\,\mathrm{serv}_0)$	*any request is eventually served*

Examples of CTL formulas

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

$\mathbf{A}\,\mathbf{F}\,\mathrm{floor}_1$	*the cabin eventually reaches the first floor*
$\mathbf{A}\,\mathbf{G}(\mathrm{op}_1 \Rightarrow \mathrm{serv}_1)$	*if the door is open, the cabin is present*
$\mathbf{A}\,\mathbf{G}(\mathrm{req}_0 \Rightarrow \mathbf{A}\,\mathbf{F}\,\mathrm{serv}_0)$	*any request is eventually served*
$\mathbf{A}\,\mathbf{G}(\mathrm{req}_0 \Rightarrow \mathbf{E}\,\mathbf{F}\,\mathrm{serv}_0)$	*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.

$\mathbf{A}\,\mathbf{F}\,\mathrm{floor}_1$	*the cabin eventually reaches the first floor*
$\mathbf{A}\,\mathbf{G}(\mathrm{op}_1 \Rightarrow \mathrm{serv}_1)$	*if the door is open, the cabin is present*
$\mathbf{A}\,\mathbf{G}(\mathrm{req}_0 \Rightarrow \mathbf{A}\,\mathbf{F}\,\mathrm{serv}_0)$	*any request is eventually served*
$\mathbf{A}\,\mathbf{G}(\mathrm{req}_0 \Rightarrow \mathbf{E}\,\mathbf{F}\,\mathrm{serv}_0)$	*any request can eventually be served*
$\mathbf{A}\,\mathbf{G}\,\mathbf{A}\,\mathbf{F}\,\mathrm{idle}_0)$	*the button is off infinitely often*

Expressiveness of CTL

Theorem ([EH86])

CTL cannot express the following property:

there is a path visiting ● infinitely many times.

[EH86] Emerson, Halpern. "Sometimes" and "Not Never" Revisited: ... J.ACM, 1986

Expressiveness of CTL

Theorem ([EH86])

CTL cannot express the following property:

there is a path visiting ● infinitely many times.

Example

A G A F ● ● *visited infinitely many times along all branches*

[EH86] Emerson, Halpern. "Sometimes" and "Not Never" Revisited: ... J.ACM, 1986

Expressiveness of CTL

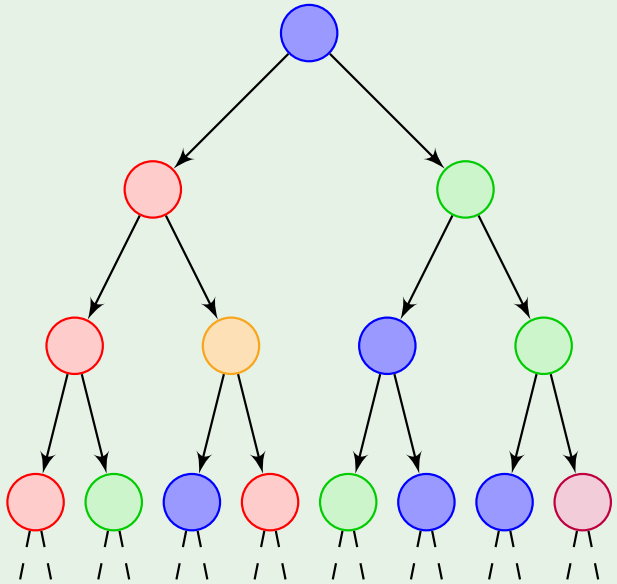
Theorem ([EH86])

CTL cannot express the following property:

there is a path visiting ● infinitely many times.

Example

A G A F ● ● *visited infinitely many times along all branches*



[EH86] Emerson, Halpern. "Sometimes" and "Not Never" Revisited: ... J.ACM, 1986

Expressiveness of CTL

Theorem ([EH86])

CTL cannot express the following property:

there is a path visiting ● infinitely many times.

Example

A G A F ● ● *visited infinitely many times along all branches*

E G E F ●

[EH86] Emerson, Halpern. "Sometimes" and "Not Never" Revisited: ... J.ACM, 1986

Expressiveness of CTL

Theorem ([EH86])

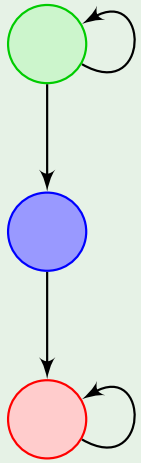
CTL cannot express the following property:

there is a path visiting ● infinitely many times.

Example

A G A F ● ● *visited infinitely many times along all branches*

E G E F ● *there is a path along which* ● *can always be visited*



[EH86] Emerson, Halpern. "Sometimes" and "Not Never" Revisited: ... J.ACM, 1986

Expressiveness of CTL

Theorem ([EH86])

CTL cannot express the following property:

there is a path visiting ● infinitely many times.

Example

A G A F ● ● *visited infinitely many times along all branches*

E G E F ● *there is a path along which* ● *can always be visited*

E G A F ●

[EH86] Emerson, Halpern. "Sometimes" and "Not Never" Revisited: ... J.ACM, 1986

Expressiveness of CTL

Theorem ([EH86])

CTL cannot express the following property:

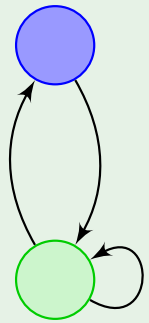
there is a path visiting ● infinitely many times.

Example

A G A F ● ● *visited infinitely many times along all branches*

E G E F ● *there is a path along which* ● *can always be visited*

E G A F ● *there is a path along which* ● *cannot be avoided*



[EH86] Emerson, Halpern. "Sometimes" and "Not Never" Revisited: ... J.ACM, 1986

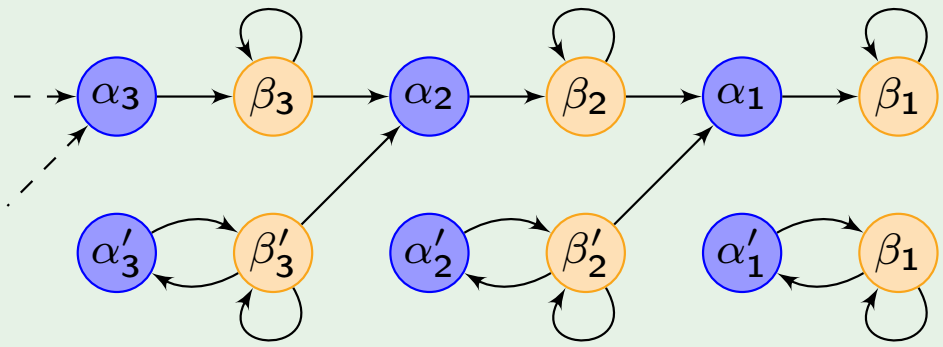
Expressiveness of CTL

Theorem ([EH86])

CTL cannot express the following property:

there is a path visiting ● infinitely many times.

Proof



$\alpha_i \not\models \mathbf{E}\,\mathbf{G}\,\mathbf{F}$ ●

$\alpha'_i \models \mathbf{E}\,\mathbf{G}\,\mathbf{F}$ ●

[EH86] Emerson, Halpern. "Sometimes" and "Not Never" Revisited: ... J.ACM, 1986

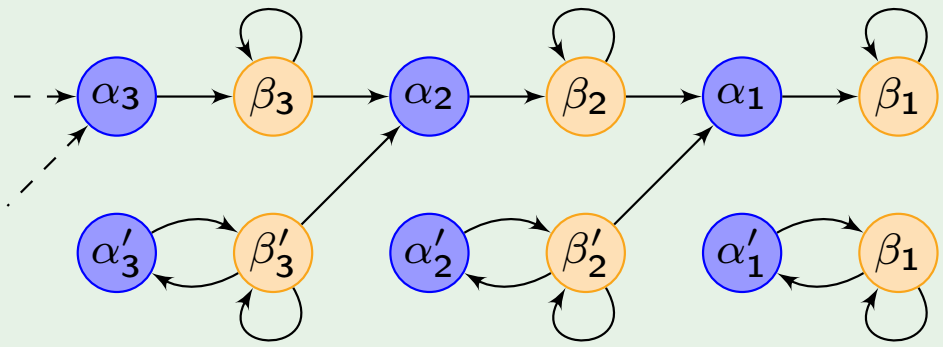
Expressiveness of CTL

Theorem ([EH86])

CTL cannot express the following property:

there is a path visiting ● infinitely many times.

Proof



$\alpha_i \not\models \mathbf{E\,G\,F}$ ●

$\alpha'_i \models \mathbf{E\,G\,F}$ ●

Lemma

α_i and α'_i satisfy the same CTL formulas of length at most i.

[EH86] Emerson, Halpern. "Sometimes" and "Not Never" Revisited: ... J.ACM, 1986

Expressiveness of CTL

Theorem

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

Expressiveness of CTL

Theorem

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

Proof

$\mathbf{A}\,\mathbf{X}$  $\equiv$

Expressiveness of CTL

Theorem

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

Proof

$$\mathbf{A}\,\mathbf{X}\,\bigcirc \equiv \neg\,\mathbf{E}\,\mathbf{X}\,\neg\,\bigcirc$$

Expressiveness of CTL

Theorem

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

Proof

$$\mathbf{A}\,\mathbf{X}\,\bigcirc \equiv \neg\,\mathbf{E}\,\mathbf{X}\,\neg\,\bigcirc$$

$$\mathbf{A}\,\bigcirc\;\mathbf{U}\;\bigcirc \equiv$$

Expressiveness of CTL

Theorem

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

Proof

$$\mathbf{A}\,\mathbf{X}\,\bigcirc \equiv \neg\,\mathbf{E}\,\mathbf{X}\,\neg\,\bigcirc$$

$$\mathbf{A}\,\bigcirc\ \mathbf{U}\ \bigcirc \equiv$$

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

Expressiveness of CTL

Theorem

Modalities **E X**, **E U** *and* **E G** *are sufficient to express any CTL formula.*

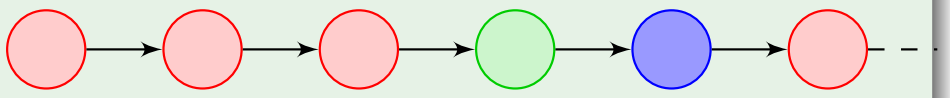
Proof

$$\mathbf{A}\,\mathbf{X}\,\bigcirc \equiv \neg\,\mathbf{E}\,\mathbf{X}\,\neg\,\bigcirc$$

$$\mathbf{A}\,\bigcirc\;\mathbf{U}\;\bigcirc \equiv$$

How to falsify $\bigcirc$ **U** $\bigcirc$:

- never $\bigcirc$;



Expressiveness of CTL

Theorem

Modalities **E X**, **E U** *and* **E G** *are sufficient to express any CTL formula.*

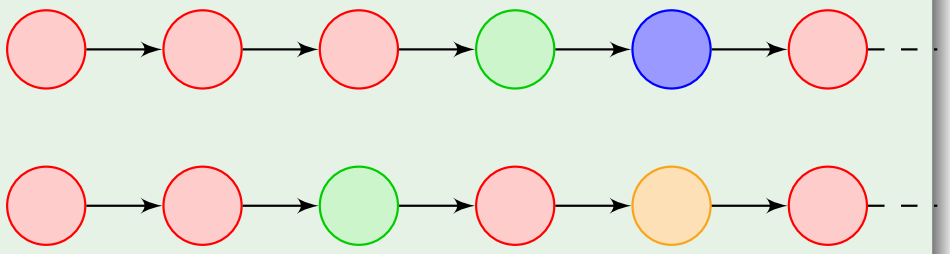
Proof

$$\mathbf{A}\,\mathbf{X}\,\bigcirc \equiv \neg\,\mathbf{E}\,\mathbf{X}\,\neg\,\bigcirc$$

$$\mathbf{A}\,\bigcirc\;\mathbf{U}\;\bigcirc \equiv$$

How to falsify $\bigcirc$ **U** $\bigcirc$:

- never $\bigcirc$;
- otherwise:
  - pick earliest $\bigcirc$;



Expressiveness of CTL

Theorem

Modalities **E X**, **E U** *and* **E G** *are sufficient to express any CTL formula.*

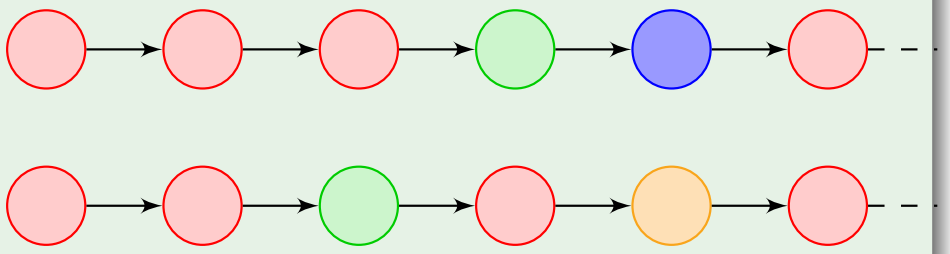
Proof

$$\mathbf{A}\,\mathbf{X}\,\bigcirc \equiv \neg\,\mathbf{E}\,\mathbf{X}\,\neg\,\bigcirc$$

$$\mathbf{A}\,\bigcirc\;\mathbf{U}\;\bigcirc \equiv$$

How to falsify $\bigcirc$ **U** $\bigcirc$:

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



Expressiveness of CTL

Theorem

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

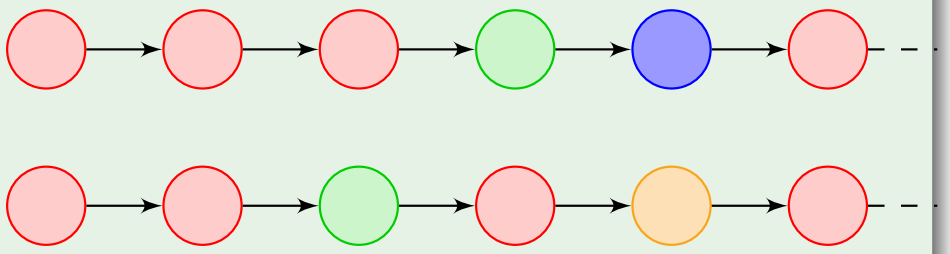
Proof

$$\mathbf{A}\,\mathbf{X}\,\bigcirc \equiv \neg\,\mathbf{E}\,\mathbf{X}\,\neg\,\bigcirc$$

$$\mathbf{A}\,\bigcirc\ \mathbf{U}\ \bigcirc \equiv \neg\,\mathbf{E}(\neg\,\bigcirc)\ \mathbf{U}\,(\neg\,\bigcirc\ \wedge\ \neg\,\bigcirc)\ \wedge\ \neg\,\mathbf{E}\,\mathbf{G}\,\neg\,\bigcirc$$

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

- never $\bigcirc$;
- otherwise:
  - pick earliest $\bigcirc$;
  - there must be a $\neg\,\bigcirc$ before.



Verifying CTL properties

Theorem ([CE81,QS82])

CTL model checking is PTIME*-complete.*

[CE81] Clarke, Emerson. Design and Synthesis ... using Branching-Time Temporal Logic. LOP, 1981.
[QS82] Queille, Sifakis. Specification and verification of concurrent systems in CESAR. SOP, 1982.

Verifying CTL properties

Theorem ([CE81,QS82])

CTL model checking is PTIME-*complete.*

Proof

label states with the subformulas they satisfy.

[CE81] Clarke, Emerson. Design and Synthesis ... using Branching-Time Temporal Logic. LOP, 1981.
[QS82] Queille, Sifakis. Specification and verification of concurrent systems in CESAR. SOP, 1982.

Verifying CTL properties

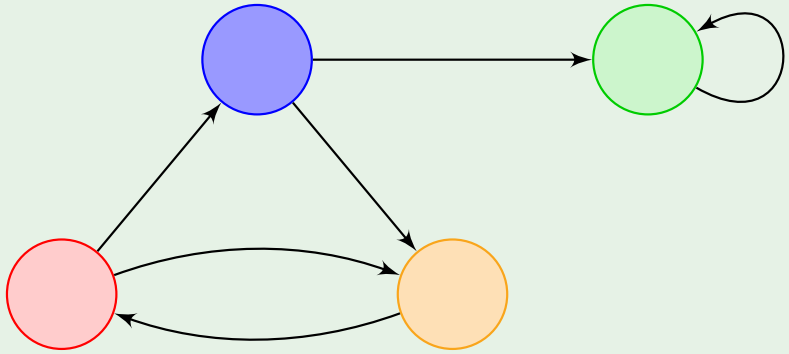
Theorem ([CE81,QS82])

CTL model checking is PTIME-*complete.*

Proof

label states with the subformulas they satisfy.

E F ● ● is reachable



[CE81] Clarke, Emerson. Design and Synthesis ... using Branching-Time Temporal Logic. LOP, 1981.

[QS82] Queille, Sifakis. Specification and verification of concurrent systems in CESAR. SOP, 1982.

Verifying CTL properties

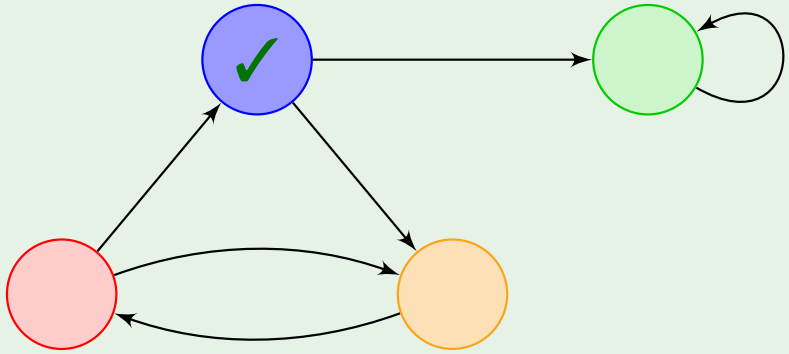
Theorem ([CE81,QS82])

CTL model checking is PTIME-*complete.*

Proof

label states with the subformulas they satisfy.

E F ● ● is reachable



[CE81] Clarke, Emerson. Design and Synthesis ... using Branching-Time Temporal Logic. LOP, 1981.

[QS82] Queille, Sifakis. Specification and verification of concurrent systems in CESAR. SOP, 1982.

Verifying CTL properties

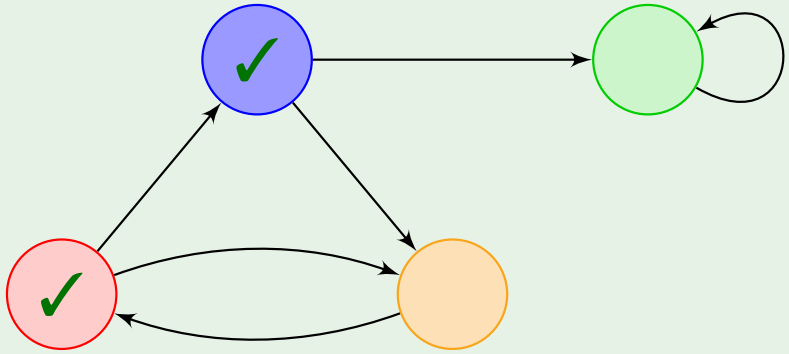
Theorem ([CE81,QS82])

CTL model checking is PTIME-*complete.*

Proof

label states with the subformulas they satisfy.

E F ● ● is reachable



[CE81] Clarke, Emerson. Design and Synthesis ... using Branching-Time Temporal Logic. LOP, 1981.

[QS82] Queille, Sifakis. Specification and verification of concurrent systems in CESAR. SOP, 1982.

Verifying CTL properties

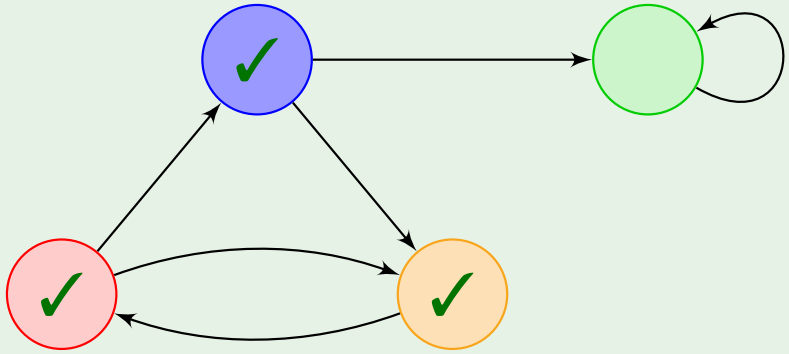
Theorem ([CE81,QS82])

CTL model checking is PTIME-*complete.*

Proof

label states with the subformulas they satisfy.

E F ● ● is reachable



[CE81] Clarke, Emerson. Design and Synthesis ... using Branching-Time Temporal Logic. LOP, 1981.

[QS82] Queille, Sifakis. Specification and verification of concurrent systems in CESAR. SOP, 1982.

Verifying CTL properties

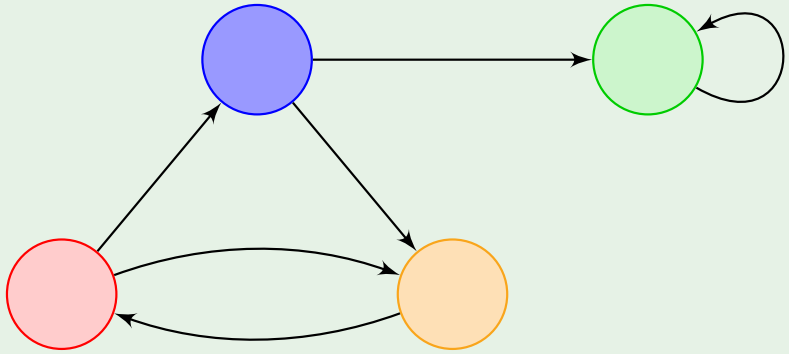
Theorem ([CE81,QS82])

CTL model checking is PTIME-*complete.*

Proof

label states with the subformulas they satisfy.

$\mathbf{E}\,\mathbf{G}(\neg$ ● $\wedge\ \mathbf{E}\,\mathbf{F}$ ●$)$ there is a path along which ● is always reachable, but never reached



[CE81] Clarke, Emerson. Design and Synthesis ... using Branching-Time Temporal Logic. LOP, 1981.

[QS82] Queille, Sifakis. Specification and verification of concurrent systems in CESAR. SOP, 1982.

Verifying CTL properties

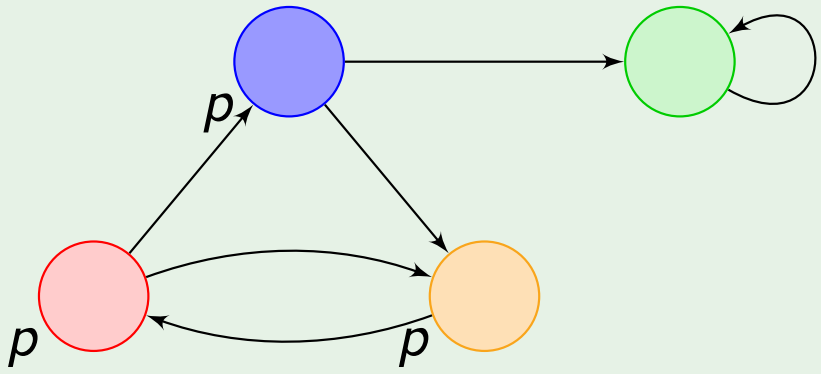
Theorem ([CE81,QS82])

CTL model checking is PTIME-*complete.*

Proof

label states with the subformulas they satisfy.

$\mathbf{E}\,\mathbf{G}(\neg \bigcirc \wedge \underbrace{\mathbf{E}\,\mathbf{F}\,\bigcirc}_{p})$ there is a path along which $\bigcirc$ is always reachable, but never reached



[CE81] Clarke, Emerson. Design and Synthesis ... using Branching-Time Temporal Logic. LOP, 1981.
[QS82] Queille, Sifakis. Specification and verification of concurrent systems in CESAR. SOP, 1982.

Verifying CTL properties

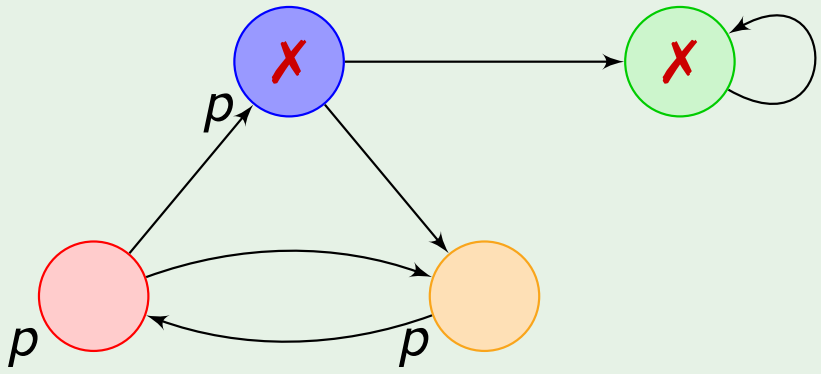
Theorem ([CE81,QS82])

CTL model checking is PTIME-*complete.*

Proof

label states with the subformulas they satisfy.

$\mathbf{E}\,\mathbf{G}(\neg \bigcirc \wedge \underbrace{\mathbf{E}\,\mathbf{F}\,\bigcirc}_{p})$ there is a path along which $\bigcirc$ is always reachable, but never reached



[CE81] Clarke, Emerson. Design and Synthesis ... using Branching-Time Temporal Logic. LOP, 1981.
[QS82] Queille, Sifakis. Specification and verification of concurrent systems in CESAR. SOP, 1982.

Verifying CTL properties

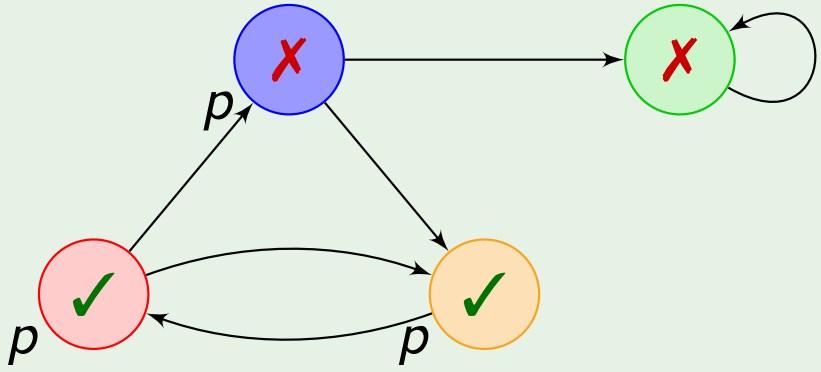
Theorem ([CE81,QS82])

CTL model checking is PTIME-*complete.*

Proof

label states with the subformulas they satisfy.

$\mathbf{E}\,\mathbf{G}(\neg \bigcirc \wedge \underbrace{\mathbf{E}\,\mathbf{F}\,\bigcirc}_{p})$ there is a path along which $\bigcirc$ is always reachable, but never reached



[CE81] Clarke, Emerson. Design and Synthesis ... using Branching-Time Temporal Logic. LOP, 1981.
[QS82] Queille, Sifakis. Specification and verification of concurrent systems in CESAR. SOP, 1982.

Verifying CTL properties

Theorem ([CE81,QS82])

CTL model checking is PTIME*-complete.*

Theorem ([KVW94])

CTL model checking on product structures is PSPACE-complete.

[CE81] Clarke, Emerson. Design and Synthesis ... using Branching-Time Temporal Logic. LOP, 1981.
[QS82] Queille, Sifakis. Specification and verification of concurrent systems in CESAR. SOP, 1982.
[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

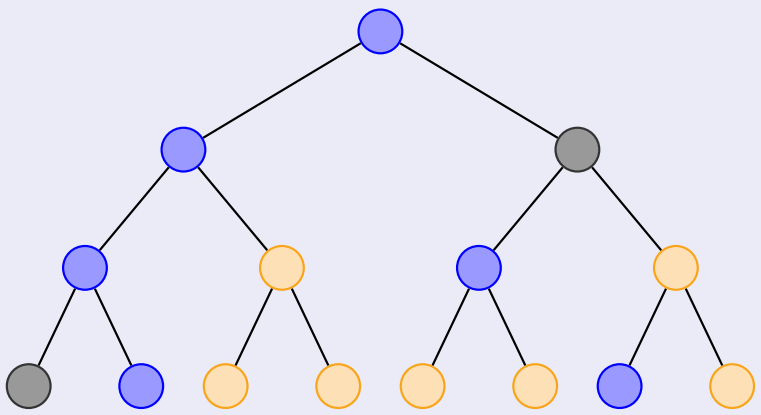
Verifying CTL properties – automata-theoretic approach

Tree automata

[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Tree automata



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Tree automata

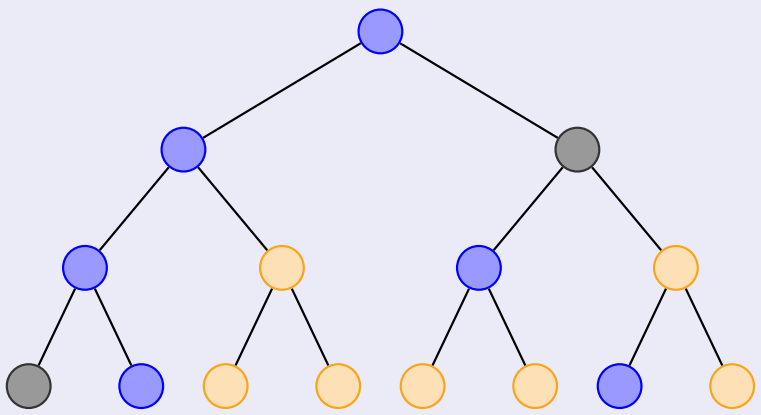
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

$\delta(q_0, \bullet) = (q_1, q_1)$

$\delta(q_0, \bullet) = (q_2, q_2)$

$\delta(q_1, \circledast) = (q_1, q_1)$

$\delta(q_2, \circledast) = (q_2, q_2)$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Tree automata

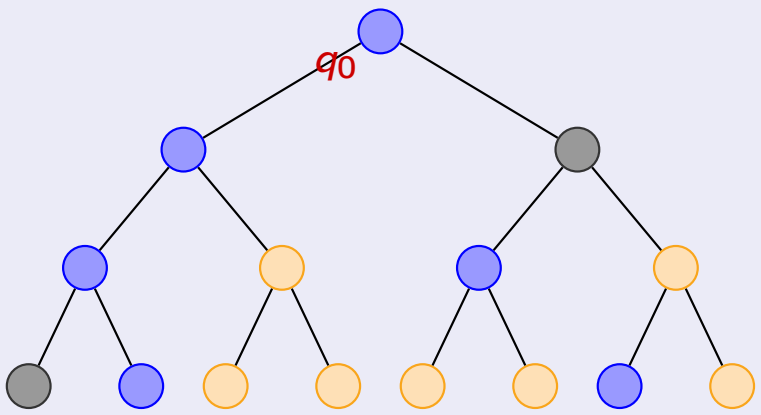
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

$\delta(q_0, \bullet) = (q_1, q_1)$

$\delta(q_0, \bullet) = (q_2, q_2)$

$\delta(q_1, \circledast) = (q_1, q_1)$

$\delta(q_2, \circledast) = (q_2, q_2)$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Tree automata

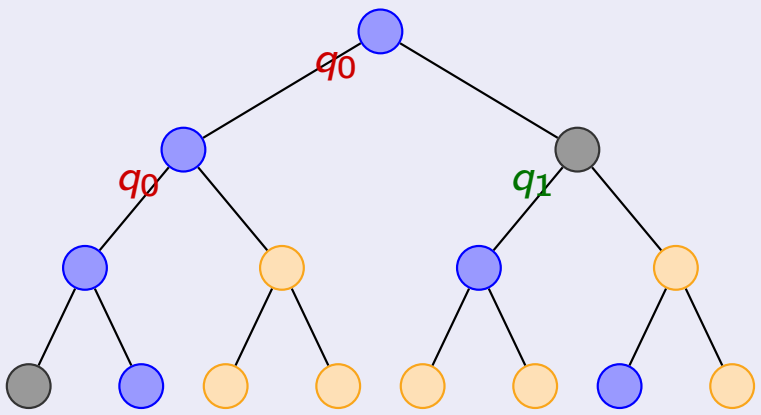
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

$\delta(q_0, \bullet) = (q_1, q_1)$

$\delta(q_0, \bullet) = (q_2, q_2)$

$\delta(q_1, \circledast) = (q_1, q_1)$

$\delta(q_2, \circledast) = (q_2, q_2)$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Tree automata

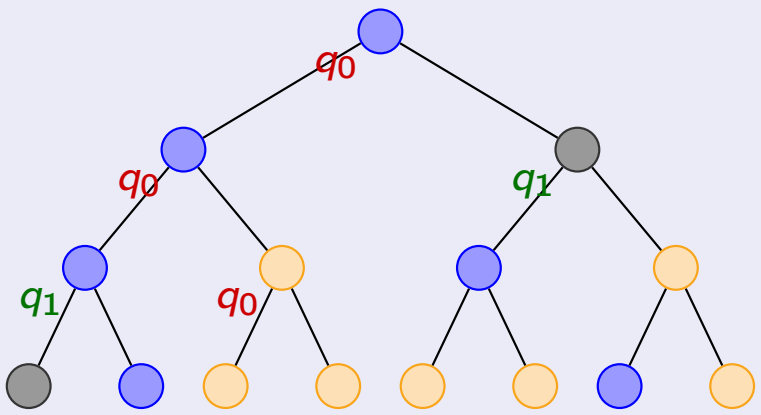
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

$\delta(q_0, \bullet) = (q_1, q_1)$

$\delta(q_0, \bullet) = (q_2, q_2)$

$\delta(q_1, \circledast) = (q_1, q_1)$

$\delta(q_2, \circledast) = (q_2, q_2)$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Tree automata

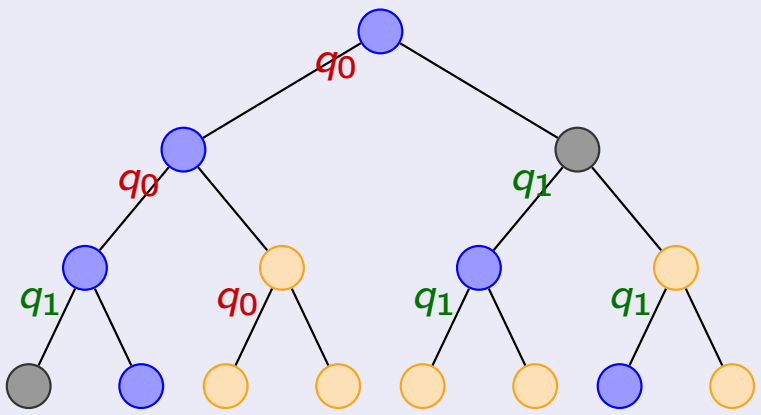
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

$\delta(q_0, \bullet) = (q_1, q_1)$

$\delta(q_0, \bullet) = (q_2, q_2)$

$\delta(q_1, \circledast) = (q_1, q_1)$

$\delta(q_2, \circledast) = (q_2, q_2)$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Tree automata

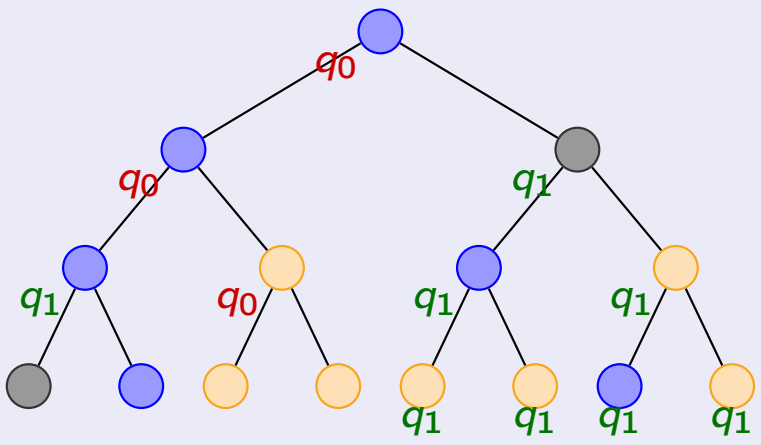
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

$\delta(q_0, \bullet) = (q_1, q_1)$

$\delta(q_0, \bullet) = (q_2, q_2)$

$\delta(q_1, \star) = (q_1, q_1)$

$\delta(q_2, \star) = (q_2, q_2)$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Tree automata

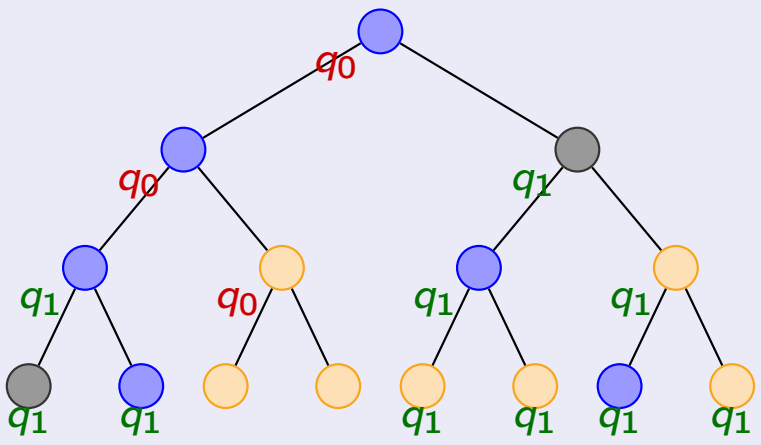
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

$\delta(q_0, \bullet) = (q_1, q_1)$

$\delta(q_0, \bullet) = (q_2, q_2)$

$\delta(q_1, \circledast) = (q_1, q_1)$

$\delta(q_2, \circledast) = (q_2, q_2)$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Tree automata

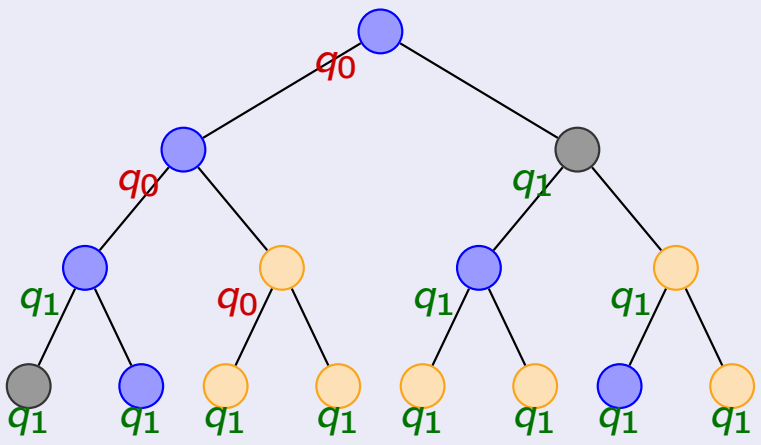
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

$\delta(q_0, \bullet) = (q_1, q_1)$

$\delta(q_0, \bullet) = (q_2, q_2)$

$\delta(q_1, \circledast) = (q_1, q_1)$

$\delta(q_2, \circledast) = (q_2, q_2)$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Tree automata

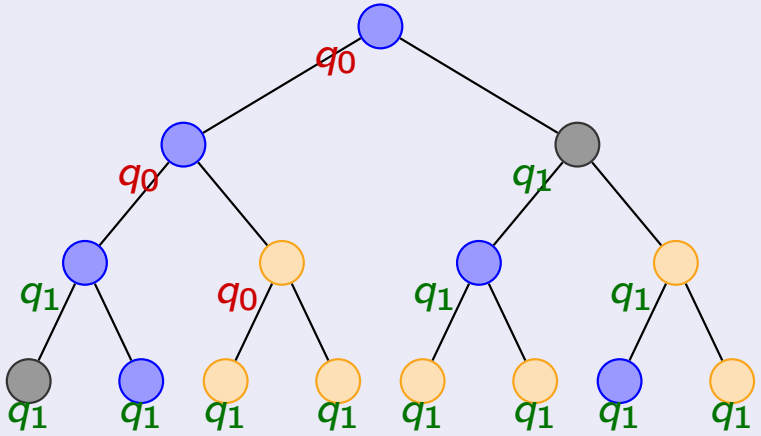
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

$\delta(q_0, \bullet) = (q_1, q_1)$

$\delta(q_0, \bullet) = (q_2, q_2)$

$\delta(q_1, \circledast) = (q_1, q_1)$

$\delta(q_2, \circledast) = (q_2, q_2)$



This automaton corresponds to **E** ● **U** ●

[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

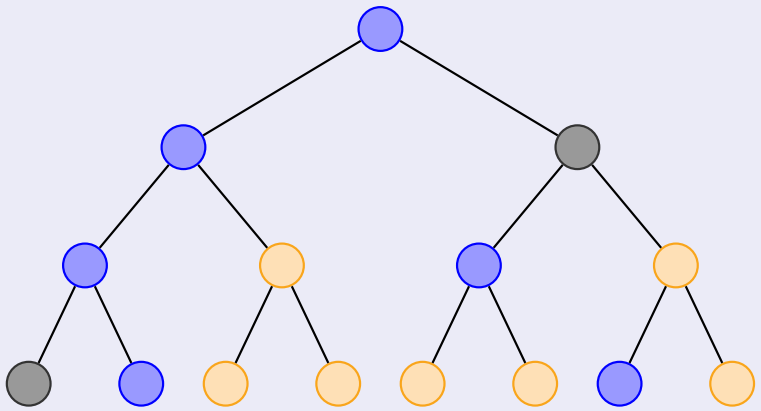
Verifying CTL properties – automata-theoretic approach

Alternating tree automata

[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Alternating tree automata



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

$$\delta(q_0, \bullet) = [(q_1, q_2) \vee (q_2, q_1)] \wedge (q_3, q_3)$$

$$\delta(q_0, \neg\bullet) = \bot$$

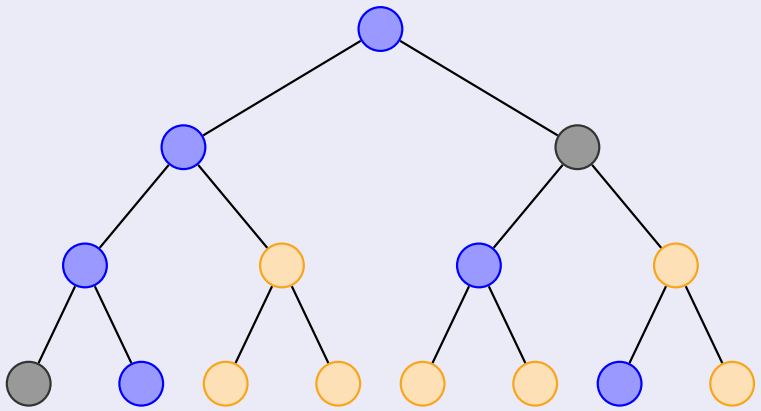
$$\delta(q_1, \bullet) = (q_1, q_2) \vee (q_2, q_1)$$

$$\delta(q_1, \neg\bullet) = \bot$$

$$\delta(q_2, \circledast) = (q_2, q_2)$$

$$\delta(q_3, \bullet) = (q_2, q_2)$$

$$\delta(q_3, \neg\bullet) = (q_3, q_3)$$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

$$\delta(q_0, \bullet) = [(q_1, q_2) \vee (q_2, q_1)] \wedge (q_3, q_3)$$

$$\delta(q_0, \neg\bullet) = \bot$$

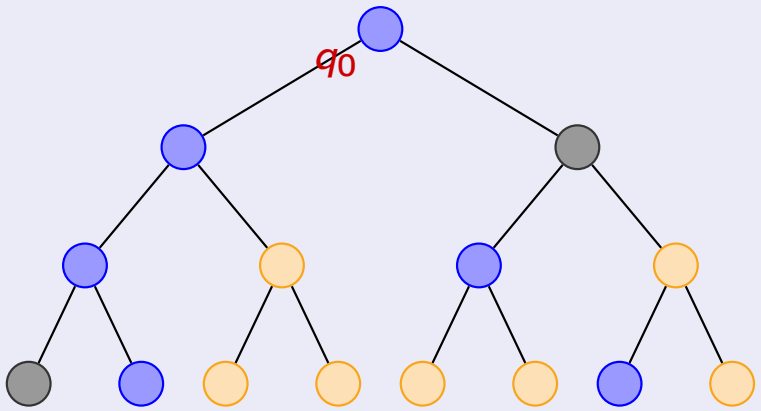
$$\delta(q_1, \bullet) = (q_1, q_2) \vee (q_2, q_1)$$

$$\delta(q_1, \neg\bullet) = \bot$$

$$\delta(q_2, \star) = (q_2, q_2)$$

$$\delta(q_3, \bullet) = (q_2, q_2)$$

$$\delta(q_3, \neg\bullet) = (q_3, q_3)$$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

$$\delta(q_0, \bigcirc) = [(q_1, q_2) \vee (q_2, q_1)] \wedge (q_3, q_3)$$

$$\delta(q_0, \neg\bigcirc) = \bot$$

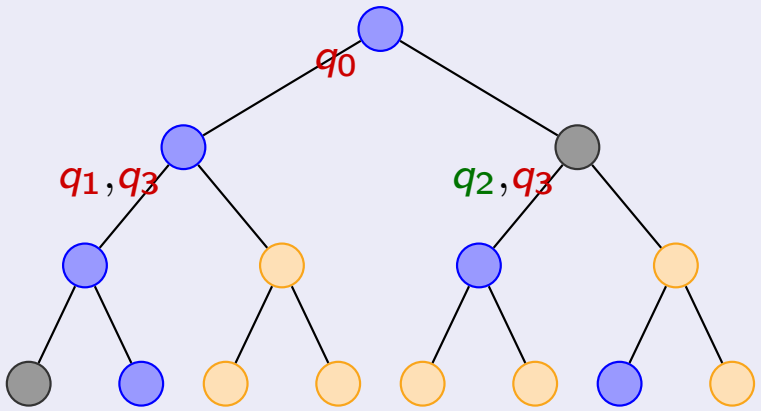
$$\delta(q_1, \bigcirc) = (q_1, q_2) \vee (q_2, q_1)$$

$$\delta(q_1, \neg\bigcirc) = \bot$$

$$\delta(q_2, \circledast) = (q_2, q_2)$$

$$\delta(q_3, \bigcirc) = (q_2, q_2)$$

$$\delta(q_3, \neg\bigcirc) = (q_3, q_3)$$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

$$\delta(q_0, \bigcirc) = [(q_1, q_2) \vee (q_2, q_1)] \wedge (q_3, q_3)$$

$$\delta(q_0, \neg\bigcirc) = \bot$$

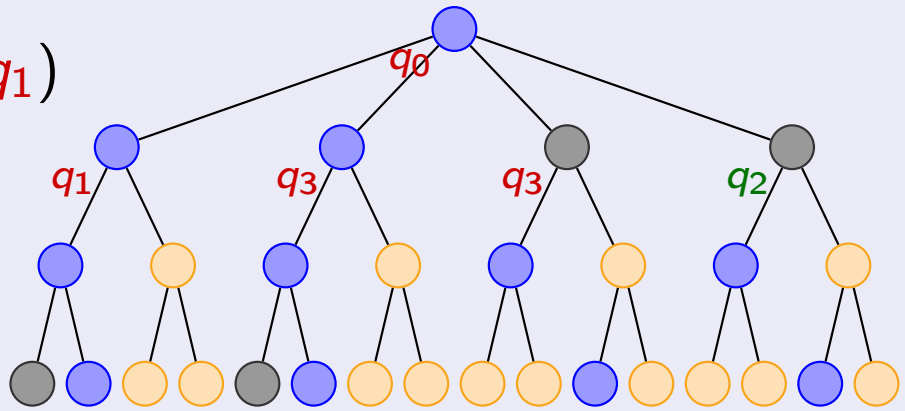
$$\delta(q_1, \bigcirc) = (q_1, q_2) \vee (q_2, q_1)$$

$$\delta(q_1, \neg\bigcirc) = \bot$$

$$\delta(q_2, \circledast) = (q_2, q_2)$$

$$\delta(q_3, \bigcirc) = (q_2, q_2)$$

$$\delta(q_3, \neg\bigcirc) = (q_3, q_3)$$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

$$\delta(q_0, \bullet) = [(q_1, q_2) \vee (q_2, q_1)] \wedge (q_3, q_3)$$

$$\delta(q_0, \neg\bullet) = \bot$$

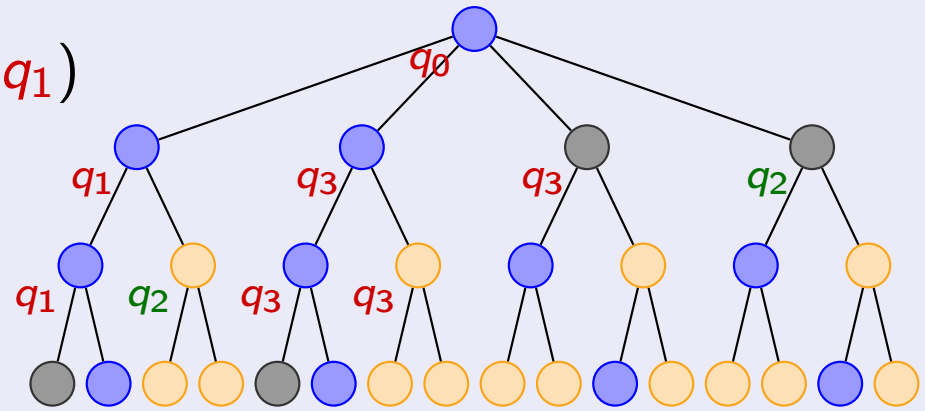
$$\delta(q_1, \bullet) = (q_1, q_2) \vee (q_2, q_1)$$

$$\delta(q_1, \neg\bullet) = \bot$$

$$\delta(q_2, \circledast) = (q_2, q_2)$$

$$\delta(q_3, \bullet) = (q_2, q_2)$$

$$\delta(q_3, \neg\bullet) = (q_3, q_3)$$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

$$\delta(q_0, \bullet) = [(q_1, q_2) \vee (q_2, q_1)] \wedge (q_3, q_3)$$

$$\delta(q_0, \neg\bullet) = \bot$$

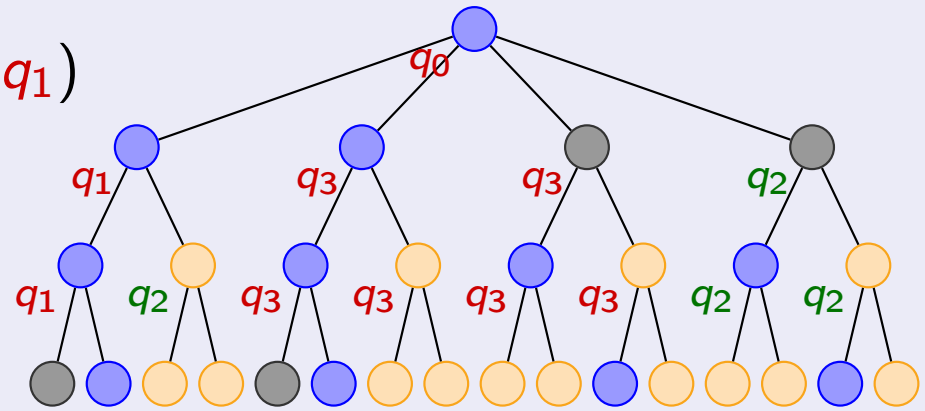
$$\delta(q_1, \bullet) = (q_1, q_2) \vee (q_2, q_1)$$

$$\delta(q_1, \neg\bullet) = \bot$$

$$\delta(q_2, \star) = (q_2, q_2)$$

$$\delta(q_3, \bullet) = (q_2, q_2)$$

$$\delta(q_3, \neg\bullet) = (q_3, q_3)$$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

$$\delta(q_0, \bullet) = [(q_1, q_2) \vee (q_2, q_1)] \wedge (q_3, q_3)$$

$$\delta(q_0, \neg\bullet) = \bot$$

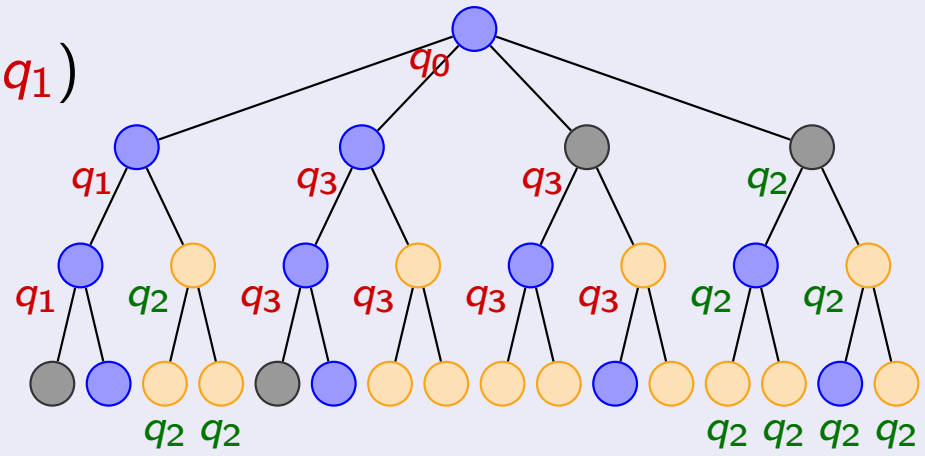
$$\delta(q_1, \bullet) = (q_1, q_2) \vee (q_2, q_1)$$

$$\delta(q_1, \neg\bullet) = \bot$$

$$\delta(q_2, \star) = (q_2, q_2)$$

$$\delta(q_3, \bullet) = (q_2, q_2)$$

$$\delta(q_3, \neg\bullet) = (q_3, q_3)$$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

$$\delta(q_0, \bigcirc) = [(q_1, q_2) \vee (q_2, q_1)] \wedge (q_3, q_3)$$

$$\delta(q_0, \neg\bigcirc) = \bot$$

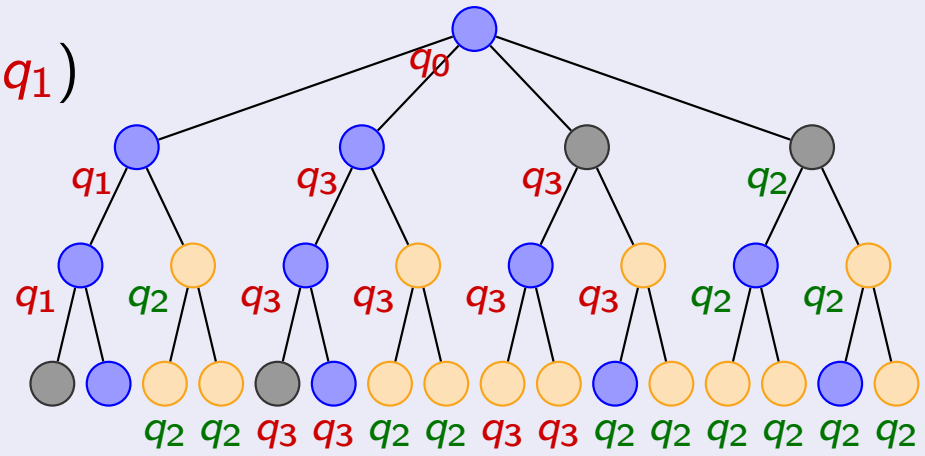
$$\delta(q_1, \bigcirc) = (q_1, q_2) \vee (q_2, q_1)$$

$$\delta(q_1, \neg\bigcirc) = \bot$$

$$\delta(q_2, \circledast) = (q_2, q_2)$$

$$\delta(q_3, \bigcirc) = (q_2, q_2)$$

$$\delta(q_3, \neg\bigcirc) = (q_3, q_3)$$



[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

$$\delta(q_0, \bigcirc) = [(q_1, q_2) \vee (q_2, q_1)] \wedge (q_3, q_3)$$

$$\delta(q_0, \neg\bigcirc) = \bot$$

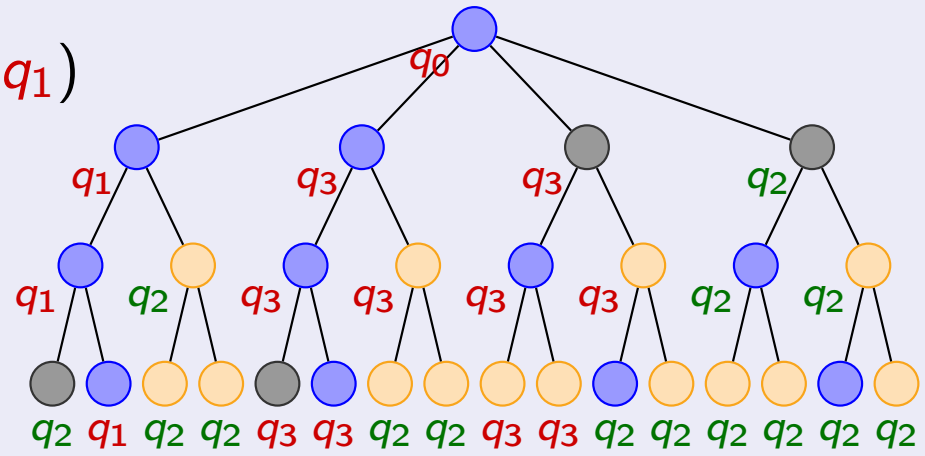
$$\delta(q_1, \bigcirc) = (q_1, q_2) \vee (q_2, q_1)$$

$$\delta(q_1, \neg\bigcirc) = \bot$$

$$\delta(q_2, \circledast) = (q_2, q_2)$$

$$\delta(q_3, \bigcirc) = (q_2, q_2)$$

$$\delta(q_3, \neg\bigcirc) = (q_3, q_3)$$

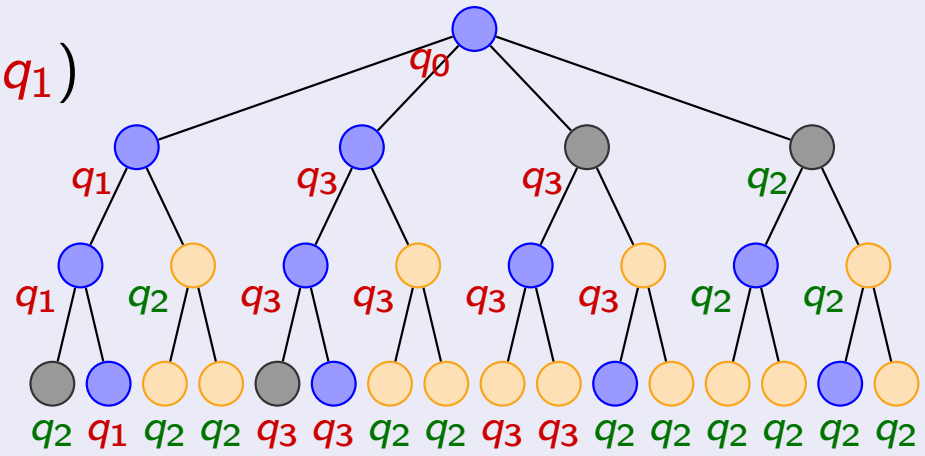


[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

Verifying CTL properties – automata-theoretic approach

Alternating tree automata

$$\delta(q_0, \bullet) = [(q_1, q_2) \vee (q_2, q_1)] \wedge (q_3, q_3)$$
$$\delta(q_0, \neg\bullet) = \bot$$
$$\delta(q_1, \bullet) = (q_1, q_2) \vee (q_2, q_1)$$
$$\delta(q_1, \neg\bullet) = \bot$$
$$\delta(q_2, \star) = (q_2, q_2)$$
$$\delta(q_3, \bullet) = (q_2, q_2)$$
$$\delta(q_3, \neg\bullet) = (q_3, q_3)$$



This automaton corresponds to $\mathbf{E}\,\mathbf{G}\,\bullet \wedge \mathbf{A}\,\mathbf{F}\,\bullet$

Detailled explanations during François' lectures.

[KVW94] Kupferman, Vardi, Wolper. An Automata-Theoretic Approach to ... CAV, 1994.

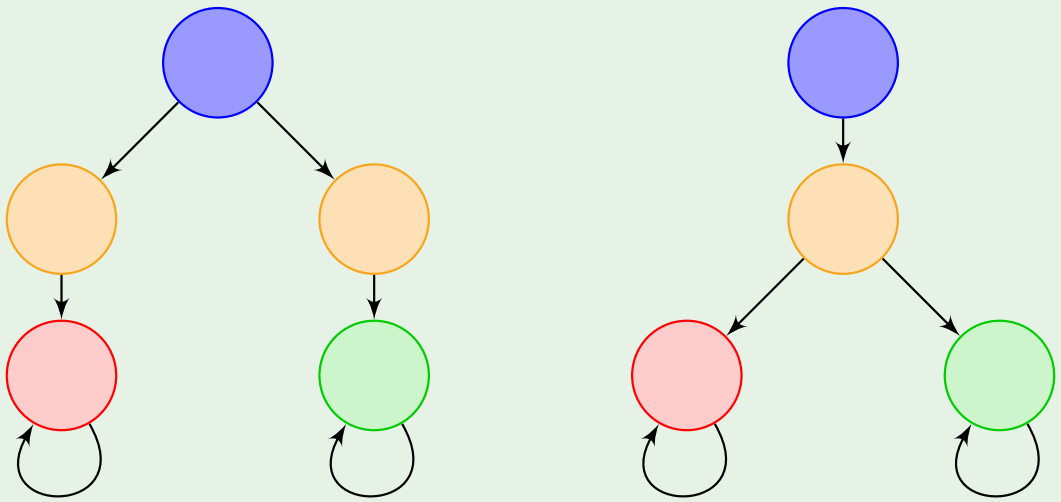
Behavioural equivalence for CTL

When are two Kripke structures indistinguishable by CTL?

Behavioural equivalence for CTL

When are two Kripke structures indistinguishable by CTL?

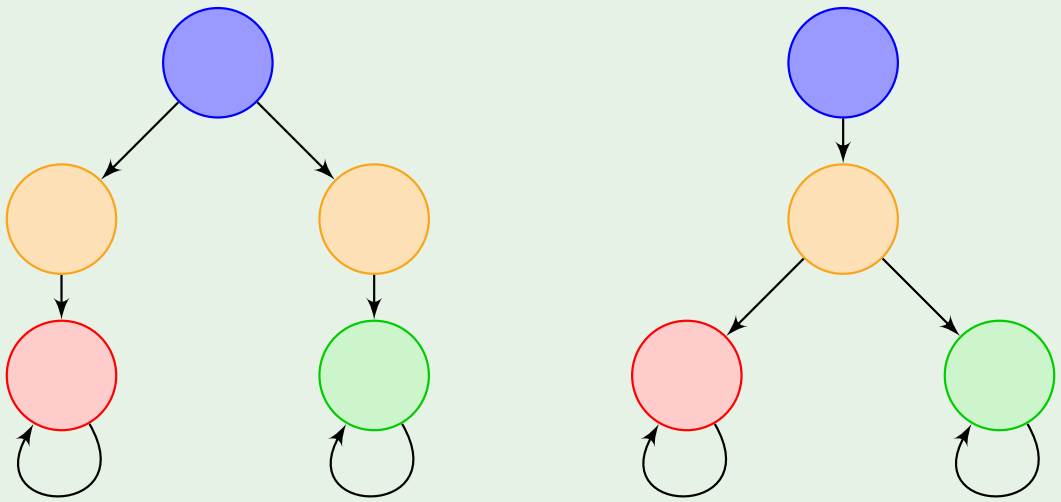
Example



Behavioural equivalence for CTL

When are two Kripke structures indistinguishable by CTL?

Example



$\mathbf{EX}(\mathbf{EX}\,\bigcirc \wedge \mathbf{EX}\,\bigcirc)$

Behavioural equivalence for CTL

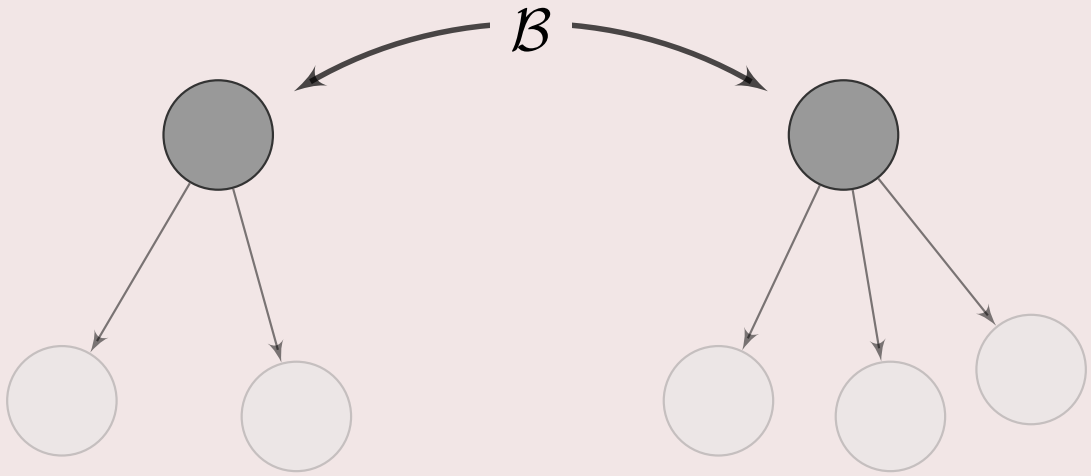
Definition

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

Behavioural equivalence for CTL

Definition

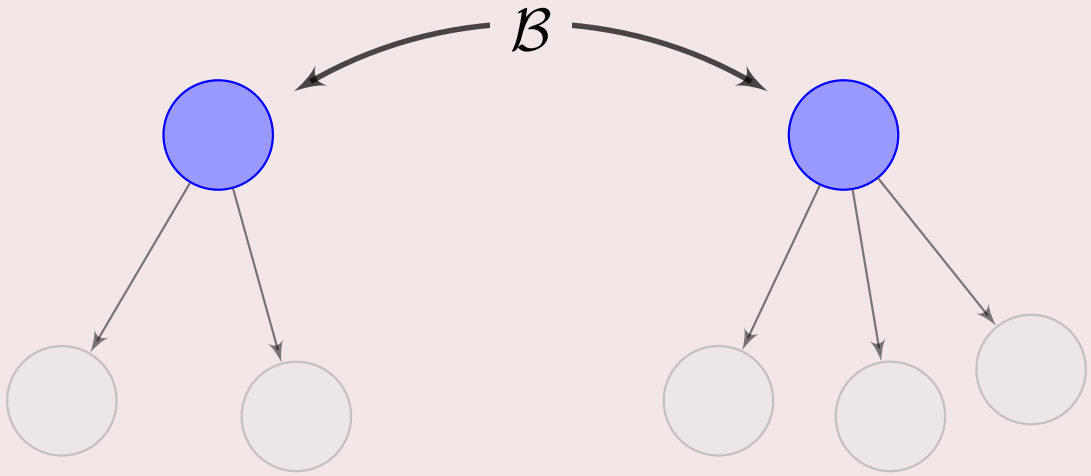
A relation $\mathcal{B} \subseteq Q \times Q$ is a bisimulation if



Behavioural equivalence for CTL

Definition

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

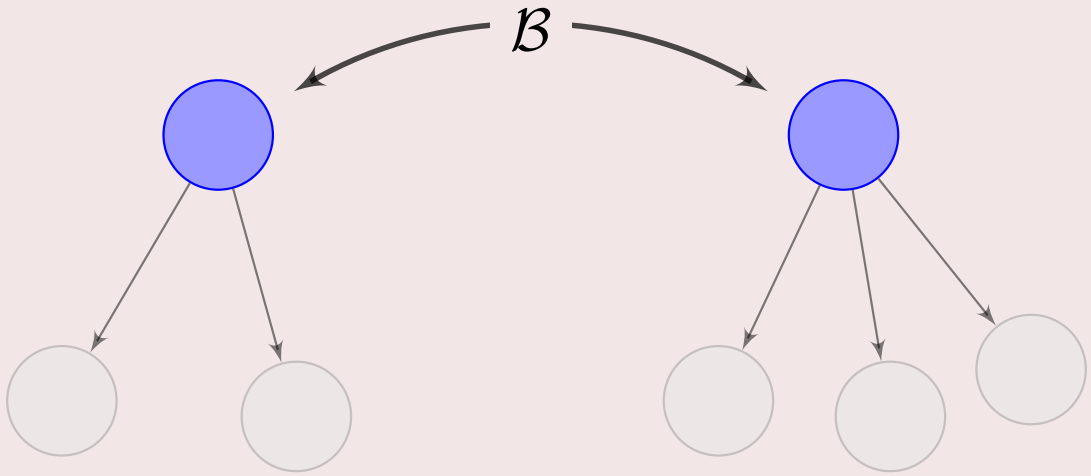


- same label;

Behavioural equivalence for CTL

Definition

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

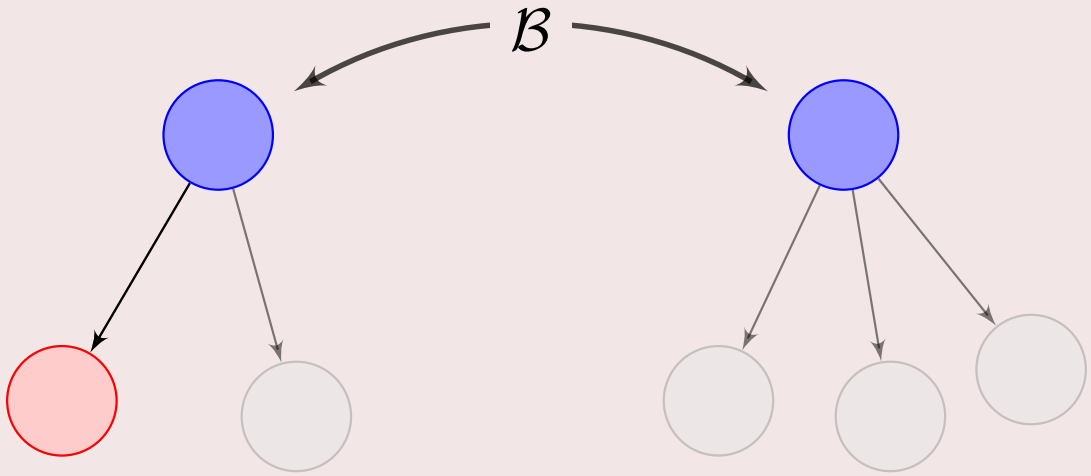


- same label;
- successors on one side have corresponding successors on other side.

Behavioural equivalence for CTL

Definition

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

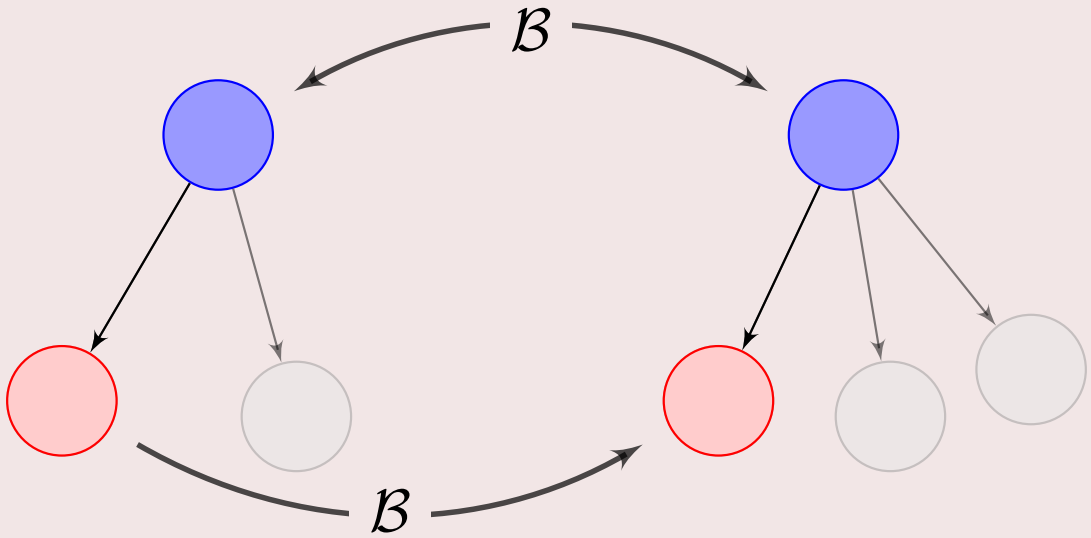


- same label;
- successors on one side have corresponding successors on other side.

Behavioural equivalence for CTL

Definition

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

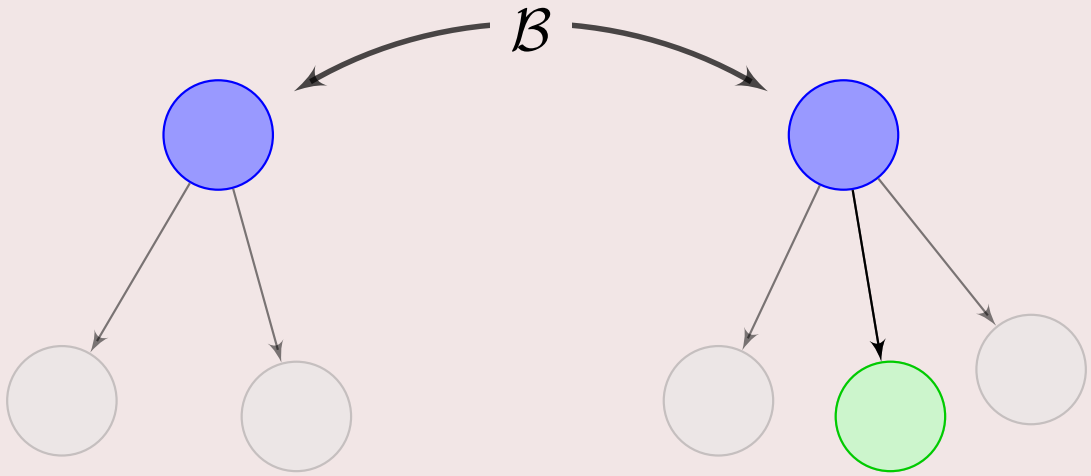


- same label;
- successors on one side have corresponding successors on other side.

Behavioural equivalence for CTL

Definition

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

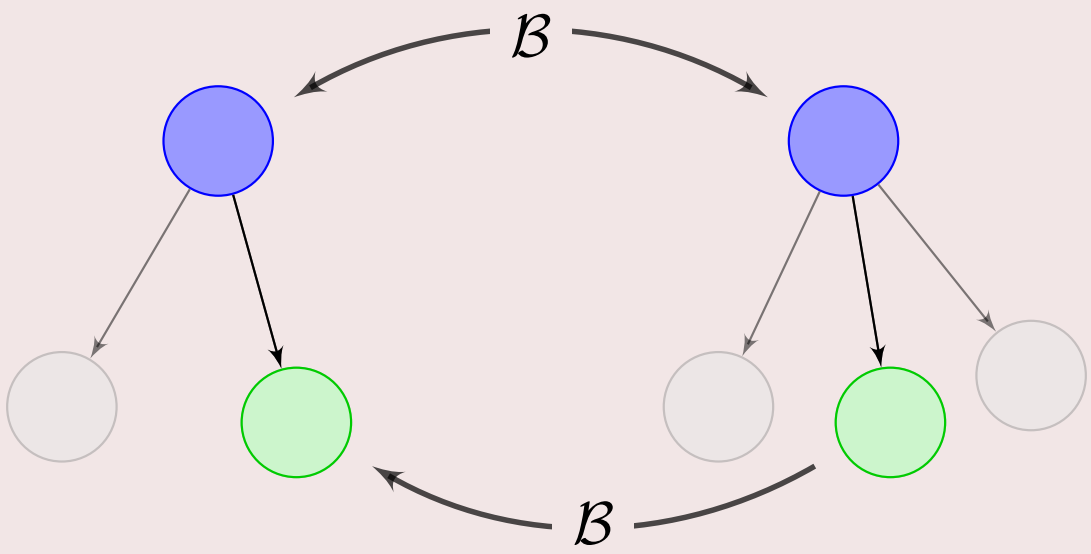


- same label;
- successors on one side have corresponding successors on other side.

Behavioural equivalence for CTL

Definition

A relation $\mathcal{B} \subseteq Q \times Q$ 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.

[BCG88] Browne, Clarke, Grumberg. Characterizing finite Kripke structures in PTL. TCS, 1988.

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

[BCG88] Browne, Clarke, Grumberg. Characterizing finite Kripke structures in PTL. TCS, 1988.

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

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

[BCG88] Browne, Clarke, Grumberg. Characterizing finite Kripke structures in PTL. TCS, 1988.

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

- 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 $\mathbf{EX}$ and $\mathbf{AX}$).

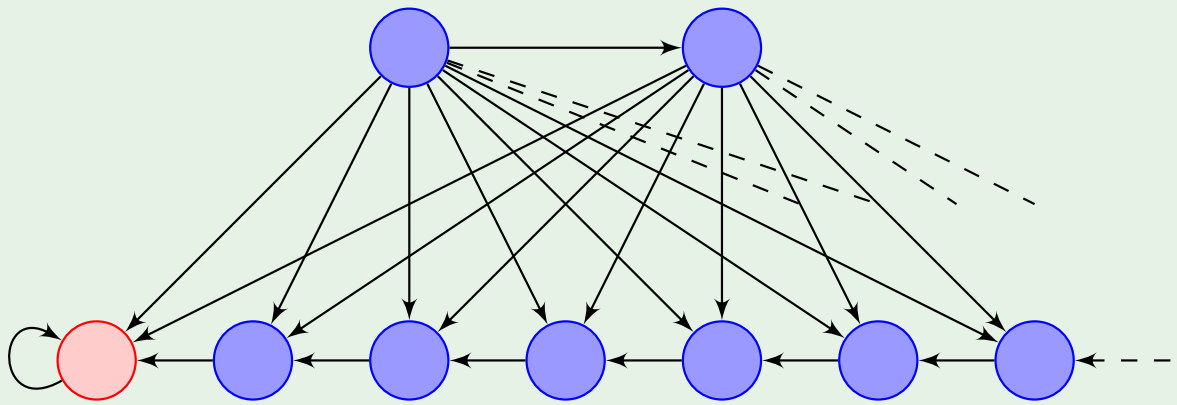
[BCG88] Browne, Clarke, Grumberg. Characterizing finite Kripke structures in PTL. TCS, 1988.

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

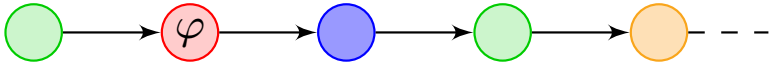


[BCG88] Browne, Clarke, Grumberg. Characterizing finite Kripke structures in PTL. TCS, 1988.

Computation-Tree Logic (CTL*)

Computation-Tree Logic (CTL*)

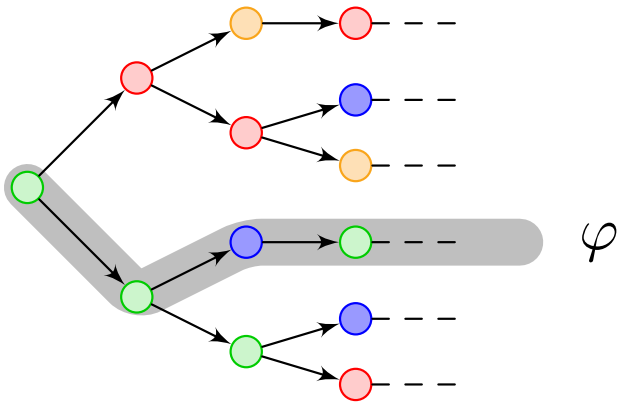
- **atomic propositions:** ◯, ◯, ...
- **boolean combinators:** $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, ...
- **temporal modalities:**

$\mathbf{X}\,\varphi$  "next φ"

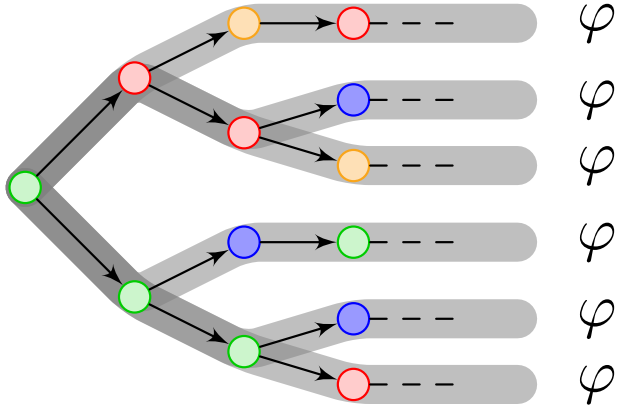
$\varphi\ \mathbf{U}\ \psi$  "φ until ψ"

- **path quantifiers:**

$\mathbf{E}\varphi$



$\mathbf{A}\varphi$



Computation-Tree Logic (CTL*)

$\mathbf{A}\,\mathbf{G}(\mathsf{req}_2 \Rightarrow \mathbf{A}(\mathbf{G}\,\mathbf{F}\,\mathsf{serv}_i \Rightarrow \mathbf{F}\,\mathsf{serv}_2))$ *any request is eventually served along any fair execution*

Computation-Tree Logic (CTL*)

$\mathbf{A}\,\mathbf{G}(\mathrm{req}_2 \Rightarrow \mathbf{A}(\mathbf{G}\,\mathbf{F}\,\mathrm{serv}_i \Rightarrow \mathbf{F}\,\mathrm{serv}_2))$ *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.

Computation-Tree Logic (CTL*)

$\mathbf{A}\,\mathbf{G}(\mathrm{req}_2 \Rightarrow \mathbf{A}(\mathbf{G}\,\mathbf{F}\,\mathrm{serv}_i \Rightarrow \mathbf{F}\,\mathrm{serv}_2))$ *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.*

[ES84] Emerson, Sistla. Deciding full branching-time logic. STOC, 1984.

Outline of the presentation

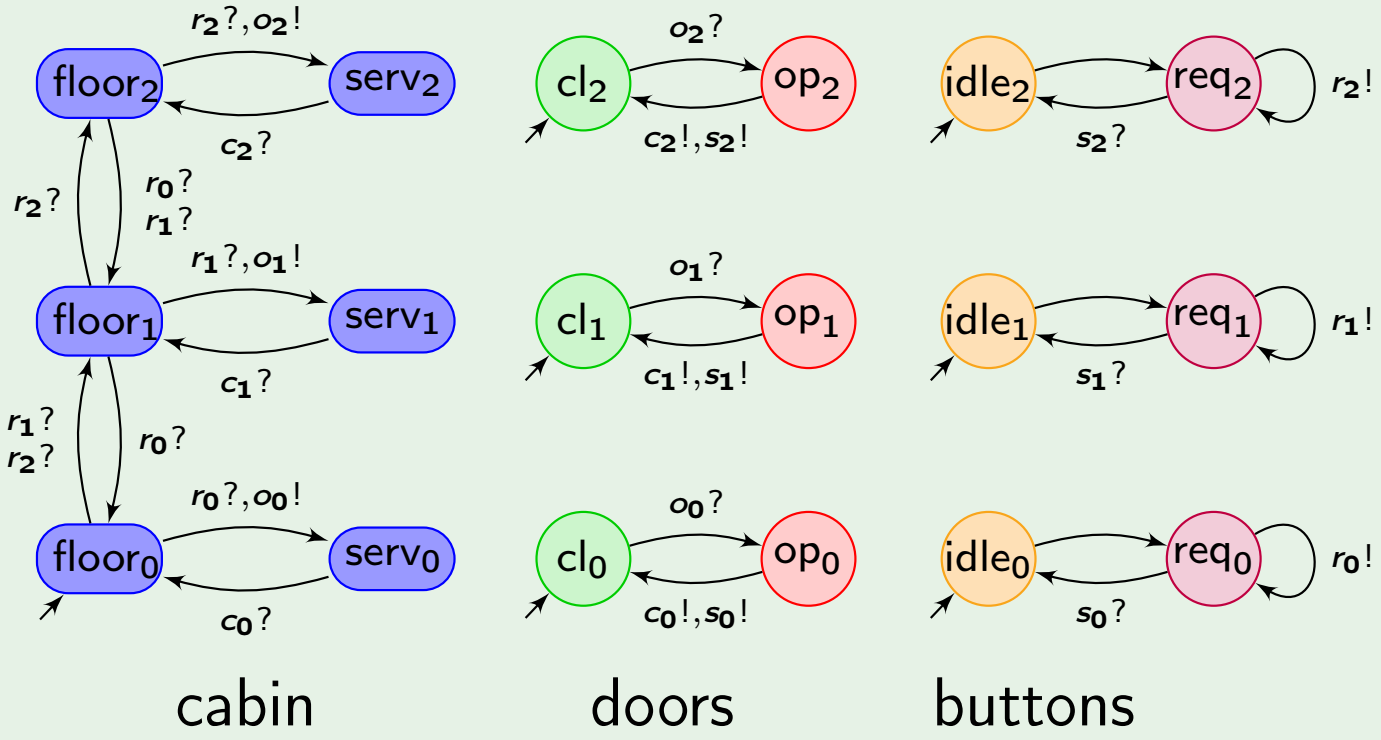
1. Introduction

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

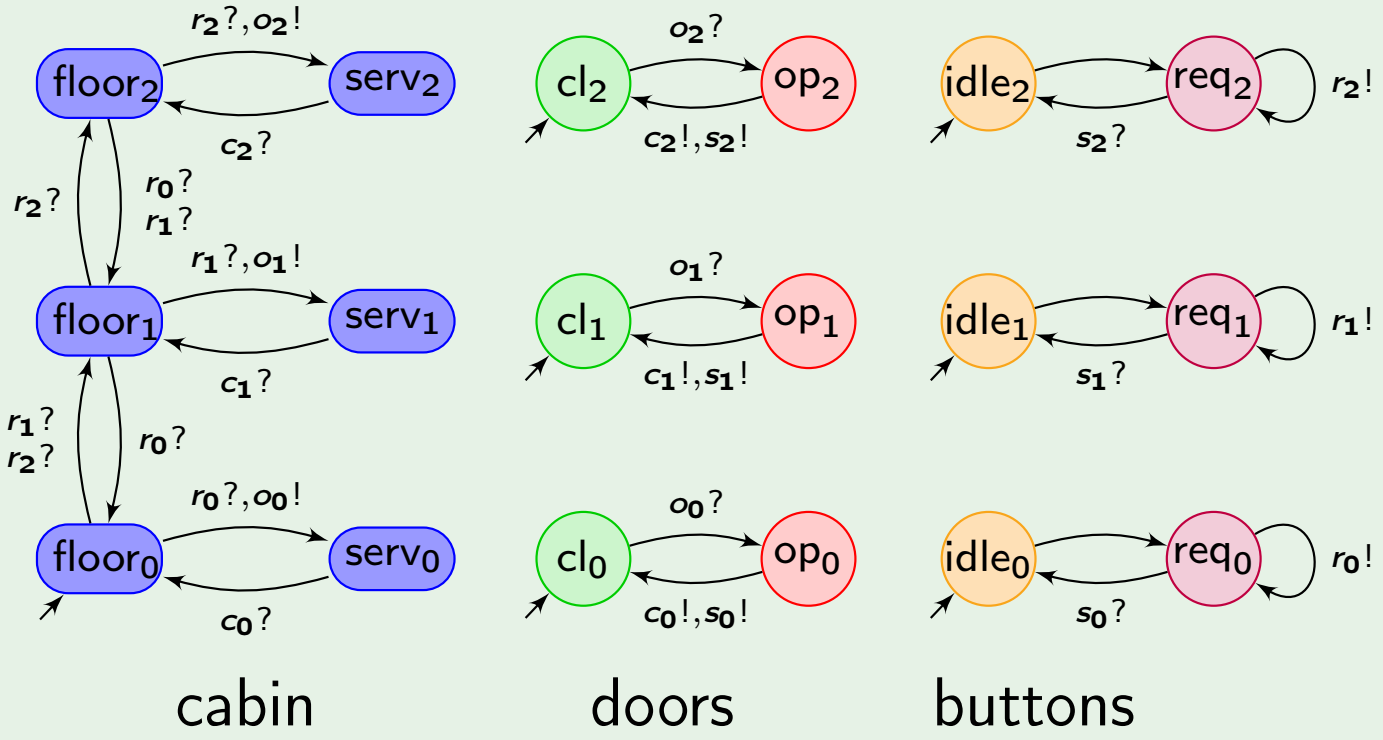
Reasoning about open systems

Example (An elevator)



Reasoning about open systems

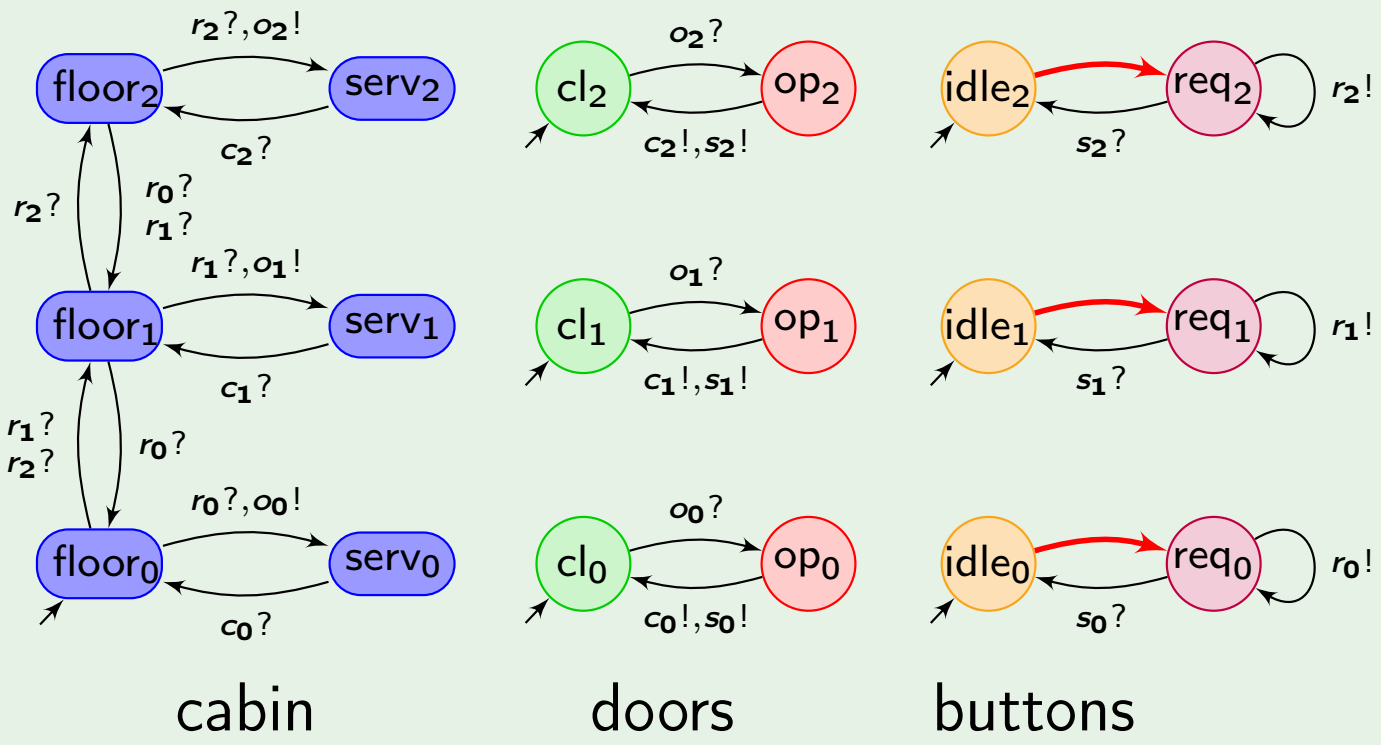
Example (An elevator)



Non-determinism due to

Reasoning about open systems

Example (An elevator)

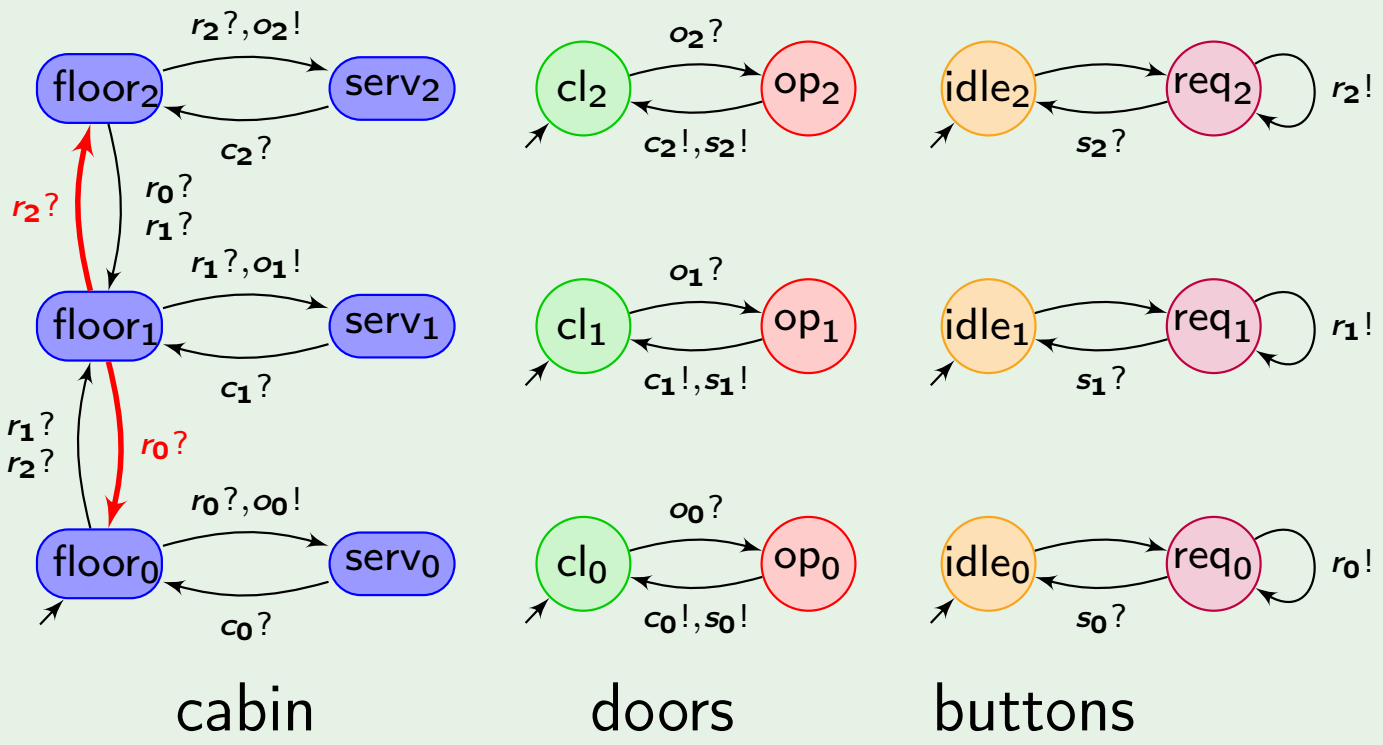


Non-determinism due to

- interactions with users;

Reasoning about open systems

Example (An elevator)

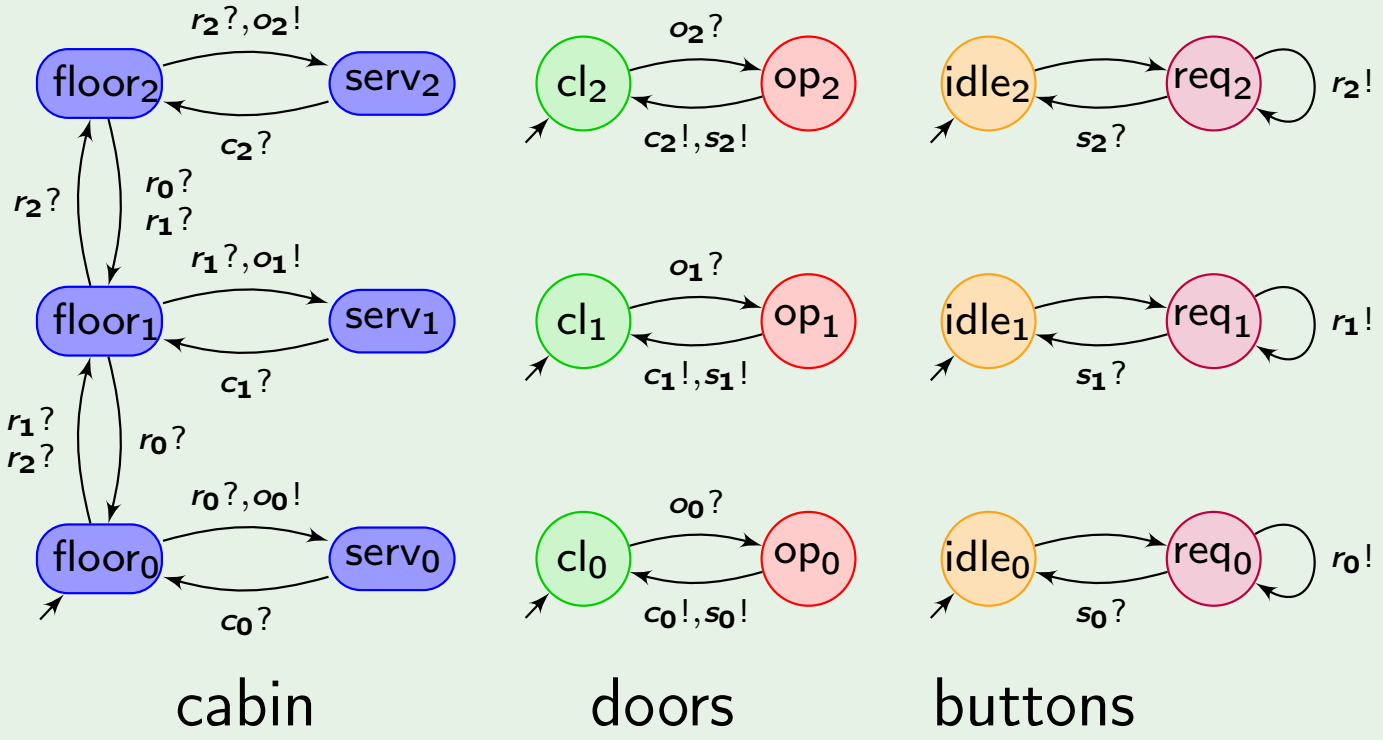


Non-determinism due to

- 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?

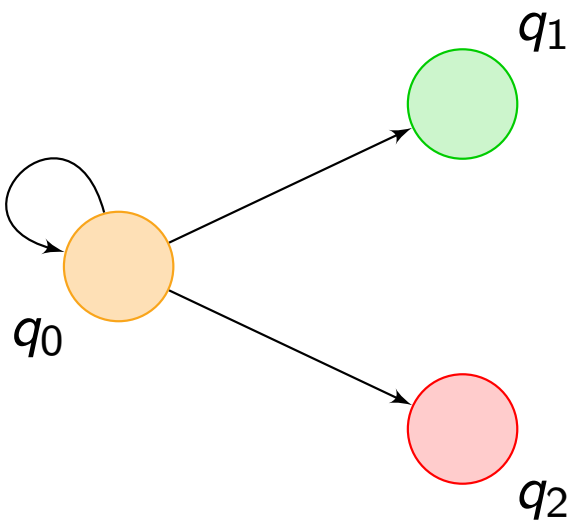
Reasoning about open systems

Reasoning about open systems

Games on graphs

A concurrent game structure is made of

- a Kripke structure

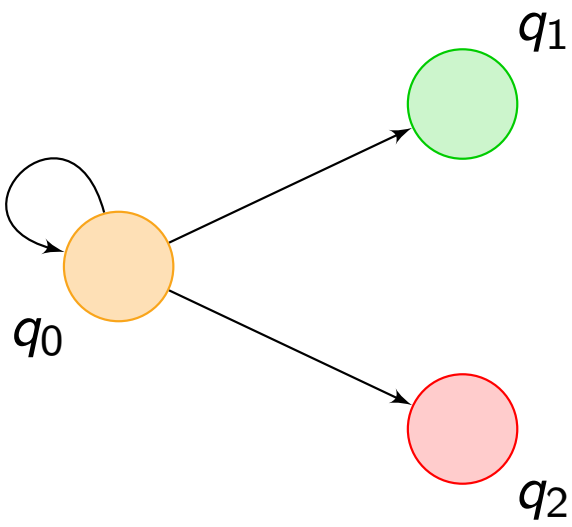


Reasoning about open systems

Games on graphs

A concurrent game structure is made of

- a Kripke structure
- a set of agents (or players);

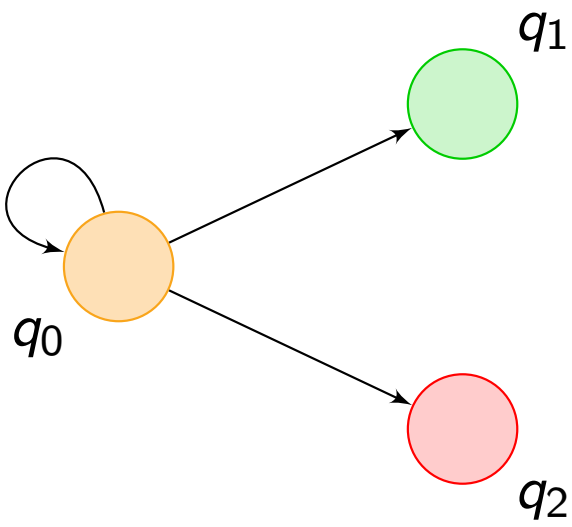


Reasoning about open systems

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.



		player 1		
				
player 2		q_0	q_2	q_1
		q_1	q_0	q_2
		q_2	q_1	q_0

Reasoning about open systems

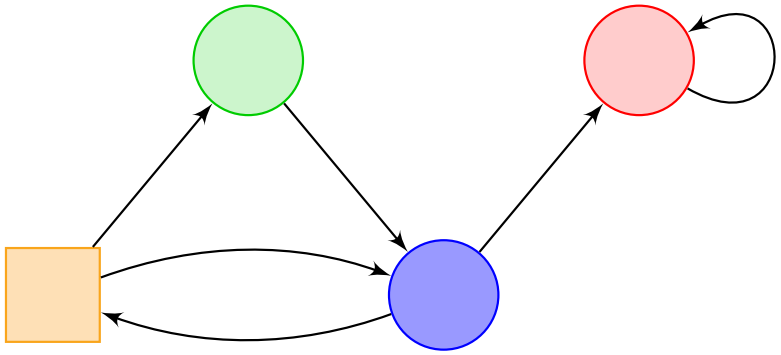
Games on graphs

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.

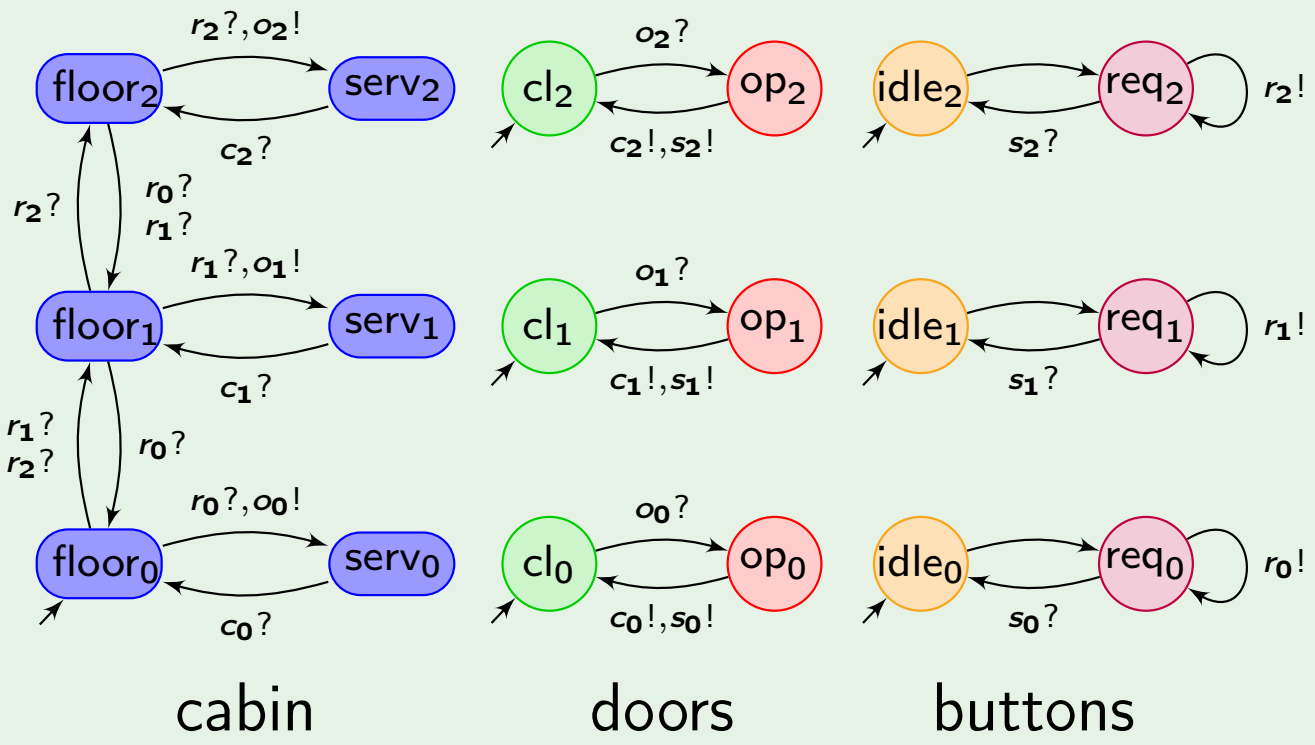
Turn-based games

A turn-based game structure is a game where each state is controlled by a single player.



Reasoning about open systems

Example (An elevator)

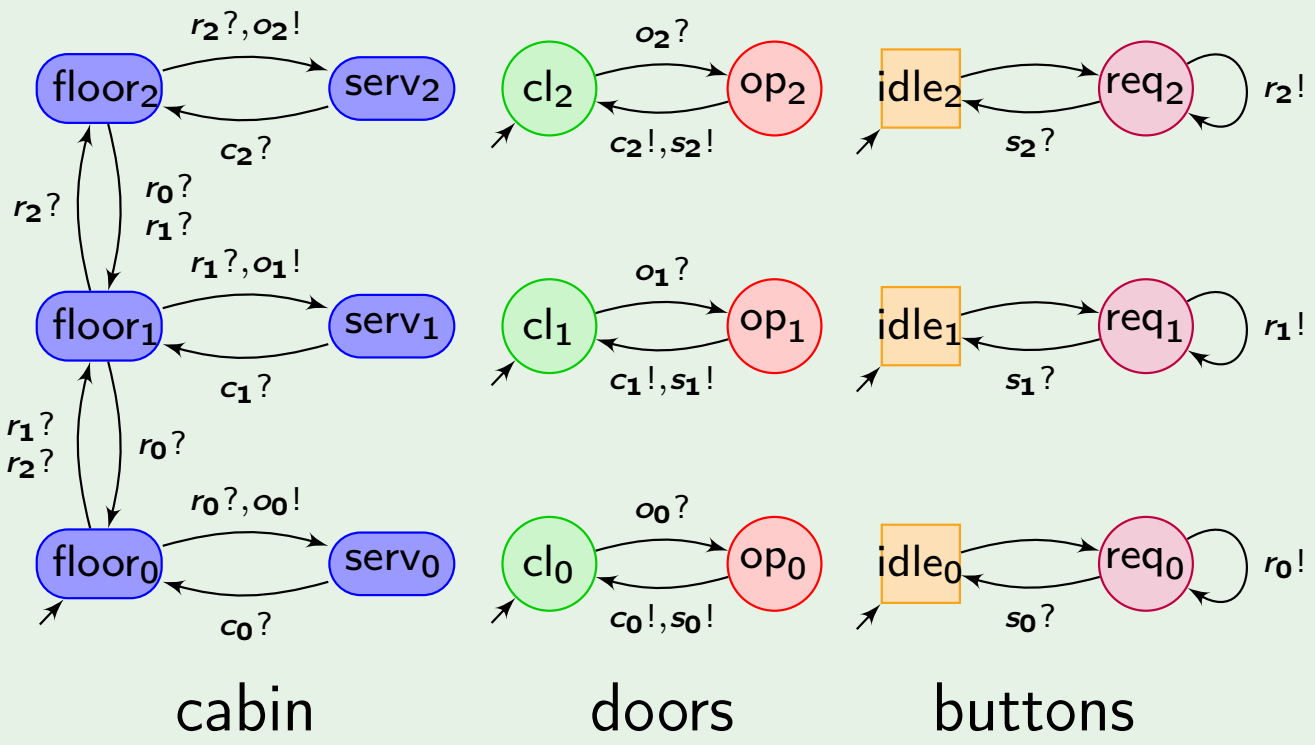


Controllability properties:

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

Reasoning about open systems

Example (An elevator)

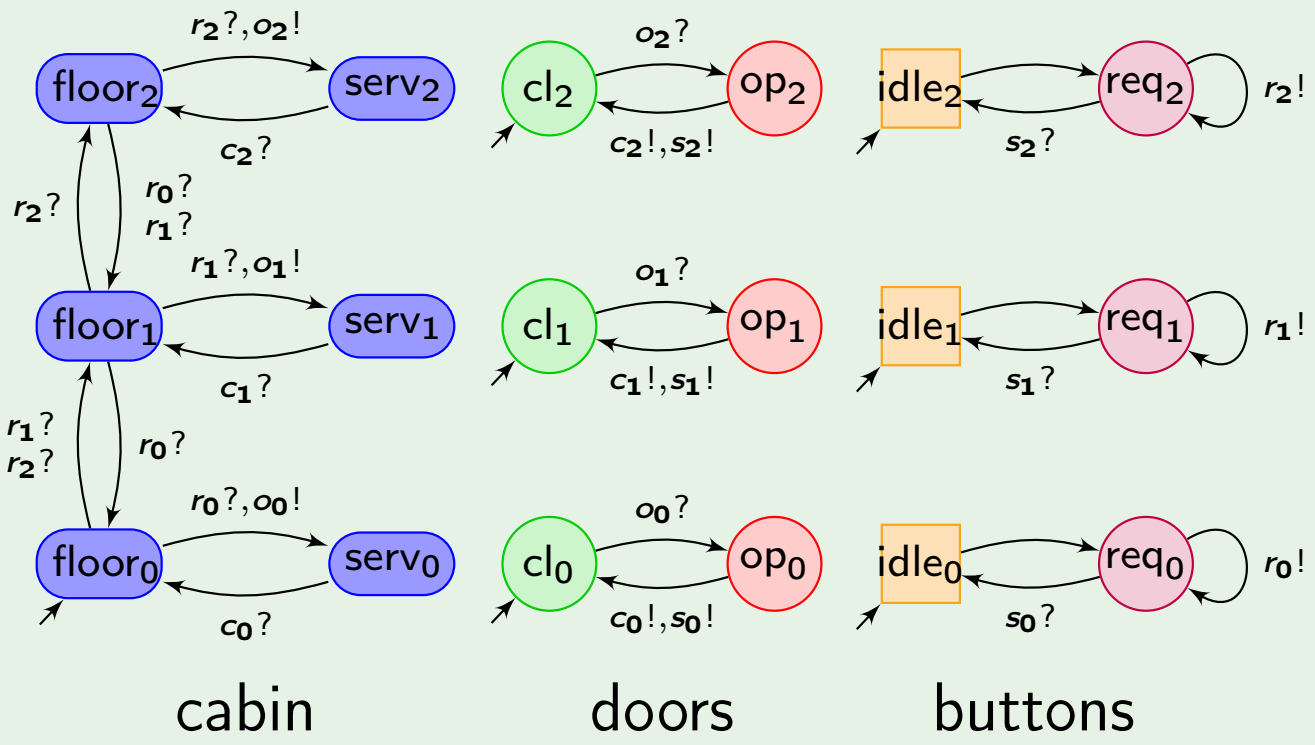


Controllability properties:

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

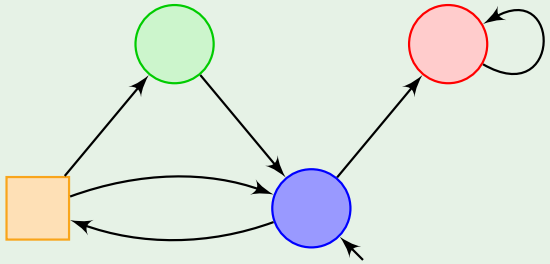
A strategy for a given player is a function telling what to play depending on what has happened previously.

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



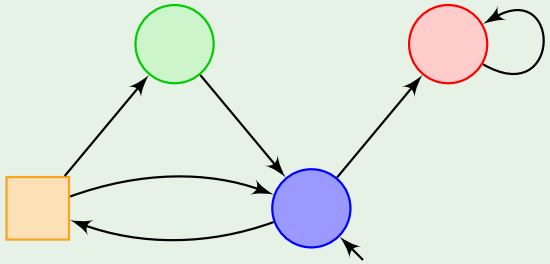
Reasoning about open systems

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

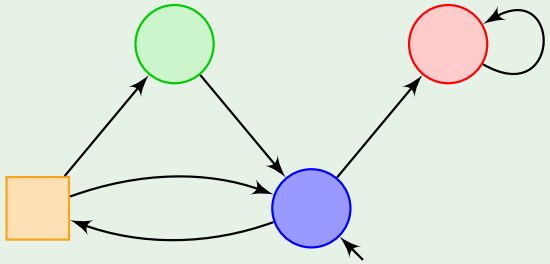
Strategies

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 ● and ○ (starting with ●).



Reasoning about open systems

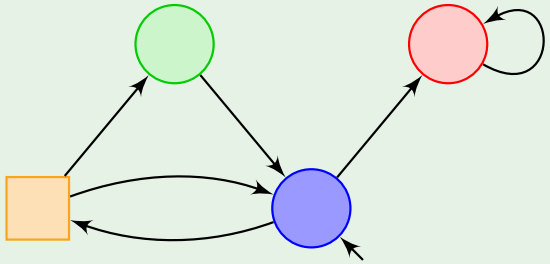
Strategies

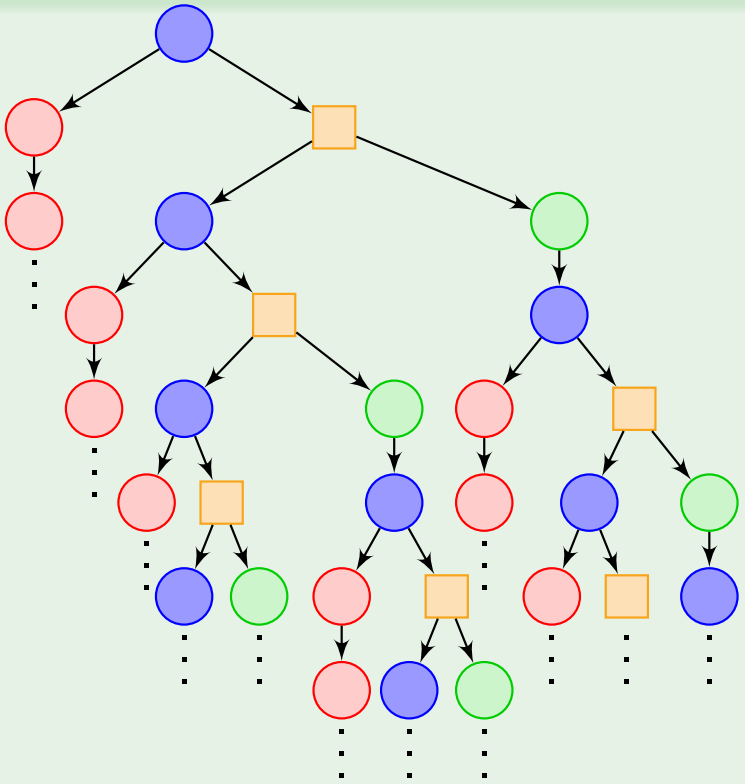
A strategy for a given player is a function telling what to play depending on what has happened previously.

Example

Strategy for player □

alternately go to ● and ● (starting with ●).





Reasoning about open systems

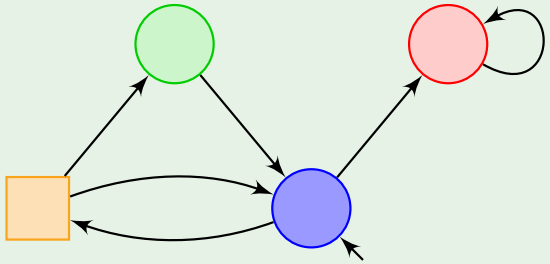
Strategies

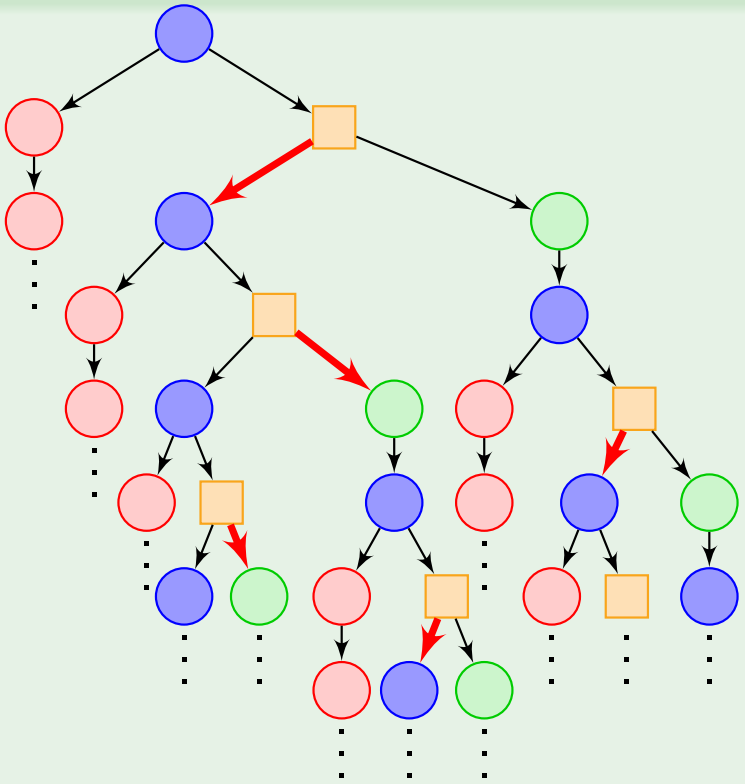
A strategy for a given player is a function telling what to play depending on what has happened previously.

Example

Strategy for player □

alternately go to ● and ● (starting with ●).





Reasoning about open systems

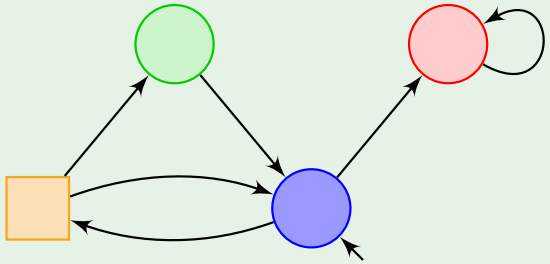
Strategies

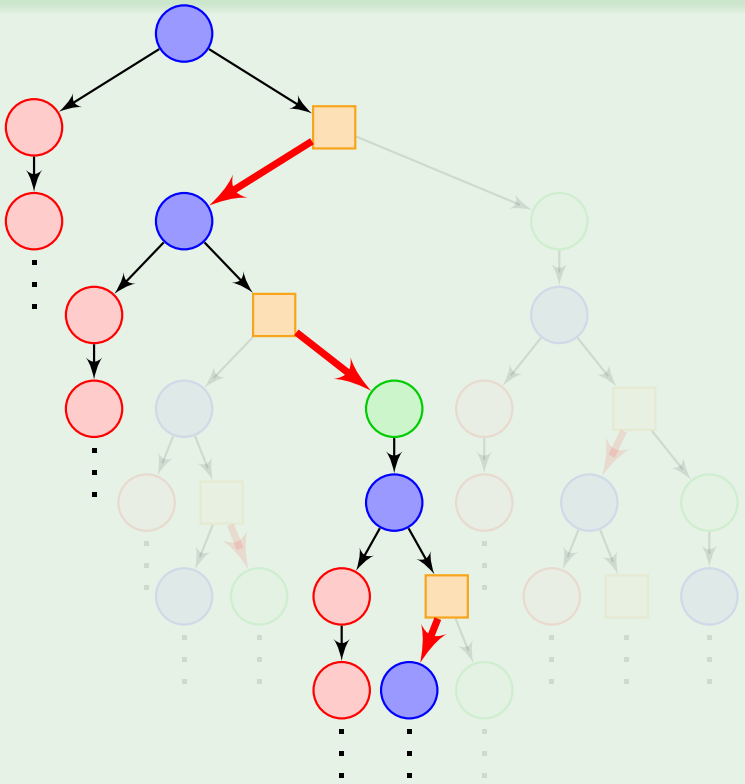
A strategy for a given player is a function telling what to play depending on what has happened previously.

Example

Strategy for player □

alternately go to ● and ● (starting with ●).





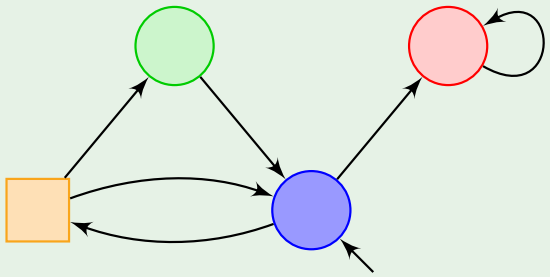
Reasoning about open systems

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 □



Reasoning about open systems

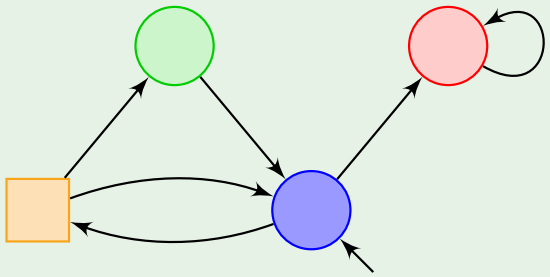
Strategies

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

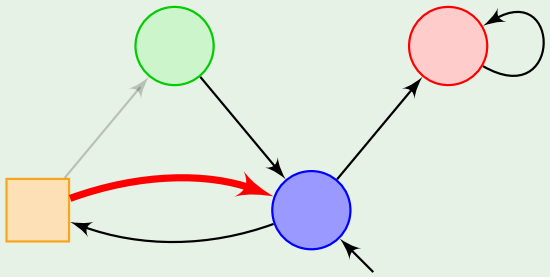
Strategies

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

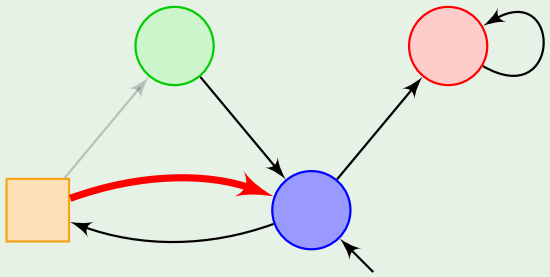
Strategies

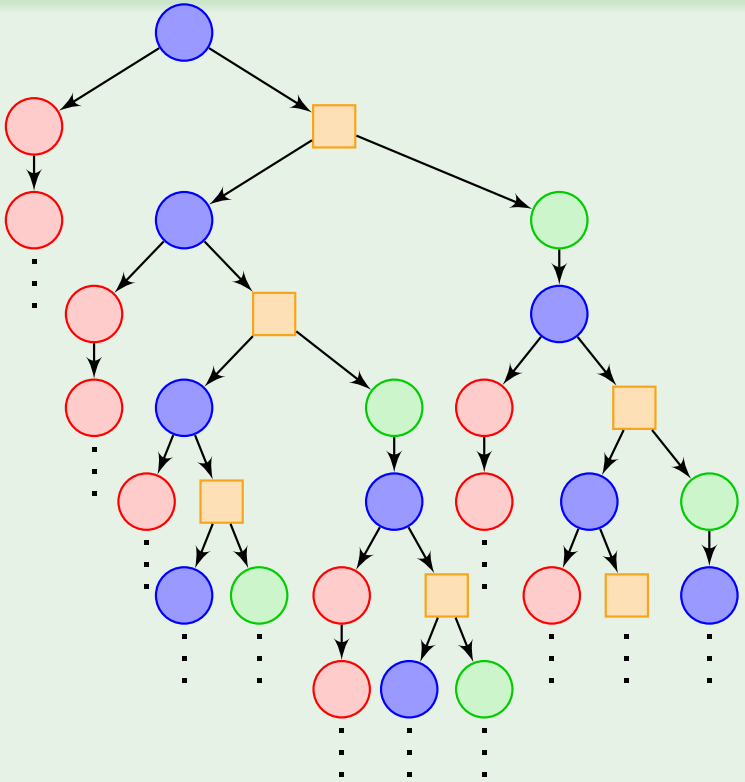
A strategy for a given player is a function telling what to play depending on what has happened previously.

Example

Memoryless strategy for player □

always go to ●.





Reasoning about open systems

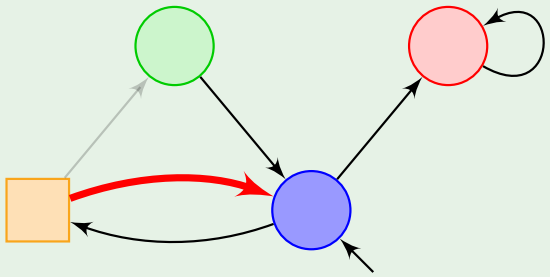
Strategies

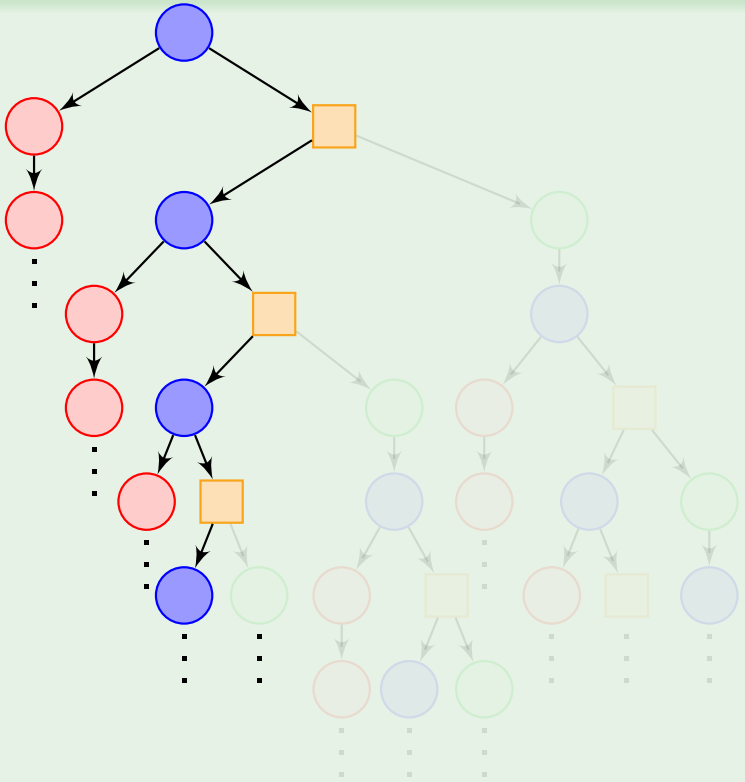
A strategy for a given player is a function telling what to play depending on what has happened previously.

Example

Memoryless strategy for player □

always go to ●.





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 φ.

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

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 φ.

$\langle\langle \bigcirc \rangle\rangle\, \mathbf{F}\, \boxed{\text{floor}_1}$

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

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

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 φ.

$\langle\langle \bigcirc \rangle\rangle\, \mathbf{F}\, \mathrm{floor}_1$

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

$\langle\langle \bigcirc \rangle\rangle\, \mathbf{G}(\mathrm{op}_1 \Rightarrow \mathrm{serv}_1)$

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

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

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 φ.

$\langle\langle \bigcirc \rangle\rangle\, \mathbf{F}\, \mathrm{floor}_1$ — *there is a controller with which the cabin eventually reaches the first floor (whatever Player $\square$ does).*

$\langle\langle \bigcirc \rangle\rangle\, \mathbf{G}(\mathrm{op}_1 \Rightarrow \mathrm{serv}_1)$ — *there is a controller for which, if the door is open, the cabin is present.*

$\mathbf{A}\, \mathbf{G}(\mathrm{req}_0 \Rightarrow \langle\langle \bigcirc \rangle\rangle\, \mathbf{F}\, \mathrm{serv}_0)$ — *any request can be granted by the controller*

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Expressiveness of ATL

Semantics of $\langle\langle A \rangle\rangle\, \varphi$

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

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

Expressiveness of ATL

Semantics of $\langle\langle A \rangle\rangle\, \varphi$

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

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

Proposition

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

Expressiveness of ATL

Semantics of $\langle\langle A \rangle\rangle\, \varphi$

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

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

Proposition

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

Proof

$$\mathbf{A}\varphi \equiv \langle\langle \varnothing \rangle\rangle\, \varphi \qquad\qquad \mathbf{E}\varphi \equiv \neg\, \langle\langle \varnothing \rangle\rangle\, \neg\, \varphi$$

Expressiveness of ATL

Semantics of $\langle\langle A \rangle\rangle\, \varphi$

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

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

Proposition

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

Proof

$$\mathbf{A}\varphi \equiv \langle\langle \varnothing \rangle\rangle\, \varphi \qquad\qquad \mathbf{E}\varphi \equiv \neg\, \langle\langle \varnothing \rangle\rangle\, \neg\, \varphi$$

Remark

Writing $\mathbf{E}\varphi \equiv \langle\langle \mathrm{Agt} \rangle\rangle\, \varphi$ would depend on the set of agents...

Expressiveness of ATL

Definition

$$[\![A]\!]\,\varphi \equiv \neg\,\langle\!\langle A \rangle\!\rangle\,\neg\,\varphi$$

Expressiveness of ATL

Definition

$$[\![A]\!]\,\varphi \equiv \neg\,\langle\!\langle A \rangle\!\rangle\,\neg\,\varphi$$

Example

$[\![A]\!]\ \mathbf{F}\,\bigcirc$

Expressiveness of ATL

Definition

$$[\![A]\!]\,\varphi \equiv \neg\,\langle\!\langle A \rangle\!\rangle\,\neg\,\varphi$$

Example

$[\![A]\!]\ \mathbf{F}$ ◯ *for any strategy of A, there is an outcome visiting* ◯

Expressiveness of ATL

Theorem

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

Expressiveness of ATL

Theorem

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

Example

Relaxed version of until in CTL:

$$\mathbf{E}\bigcirc\ \mathbf{W}\ \bigcirc \equiv \mathbf{E}\big[\bigcirc\ \mathbf{U}\ \bigcirc \vee \mathbf{G}\ \bigcirc\big]$$

Expressiveness of ATL

Theorem

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

Example

Relaxed version of until in CTL:

$$\begin{aligned} \mathbf{E}\bigcirc\ \mathbf{W}\ \bigcirc &\equiv \mathbf{E}\big[\bigcirc\ \mathbf{U}\ \bigcirc \vee \mathbf{G}\,\bigcirc\big] \\ &\equiv (\mathbf{E}\bigcirc\ \mathbf{U}\ \bigcirc) \vee (\mathbf{E}\,\mathbf{G}\,\bigcirc) \end{aligned}$$

Expressiveness of ATL

Theorem

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

Example

Relaxed version of until in CTL:

$$\begin{aligned} \mathbf{E}\, \bigcirc\, \mathbf{W}\, \bigcirc &\equiv \mathbf{E}\big[\bigcirc\, \mathbf{U}\, \bigcirc \vee \mathbf{G}\, \bigcirc\big] \\ &\equiv (\mathbf{E}\, \bigcirc\, \mathbf{U}\, \bigcirc) \vee (\mathbf{E}\, \mathbf{G}\, \bigcirc) \end{aligned}$$

In ATL (actually ATL*):

$$\langle\langle A \rangle\rangle\, \bigcirc\, \mathbf{W}\, \bigcirc \equiv \langle\langle A \rangle\rangle\, \big[\bigcirc\, \mathbf{U}\, \bigcirc \vee \mathbf{G}\, \bigcirc\big]$$

Expressiveness of ATL

Theorem

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

Example

Relaxed version of until in CTL:

$$\begin{aligned} \mathbf{E}\,\bigcirc\, \mathbf{W}\, \bigcirc &\equiv \mathbf{E}\big[\bigcirc\, \mathbf{U}\, \bigcirc \vee \mathbf{G}\, \bigcirc\big] \\ &\equiv (\mathbf{E}\,\bigcirc\, \mathbf{U}\, \bigcirc) \vee (\mathbf{E}\,\mathbf{G}\, \bigcirc) \end{aligned}$$

In ATL (actually ATL*):

$$\begin{aligned} \langle\langle A \rangle\rangle\, \bigcirc\, \mathbf{W}\, \bigcirc &\equiv \langle\langle A \rangle\rangle\, \big[\bigcirc\, \mathbf{U}\, \bigcirc \vee \mathbf{G}\, \bigcirc\big] \\ &\not\equiv \langle\langle A \rangle\rangle\, \bigcirc\, \mathbf{U}\, \bigcirc \vee \langle\langle A \rangle\rangle\, \mathbf{G}\, \bigcirc \end{aligned}$$

Expressiveness of ATL

Theorem

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

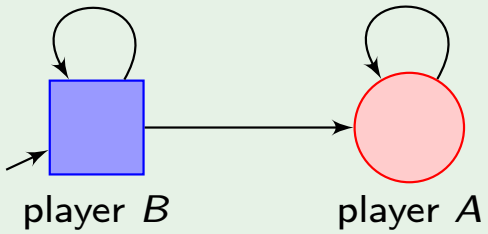
Example

Relaxed version of until in CTL:

$$\mathbf{E}\,\bigcirc\,\mathbf{W}\,\bigcirc \equiv \mathbf{E}\big[\bigcirc\,\mathbf{U}\,\bigcirc \vee \mathbf{G}\,\bigcirc\big]$$
$$\equiv (\mathbf{E}\,\bigcirc\,\mathbf{U}\,\bigcirc) \vee (\mathbf{E}\,\mathbf{G}\,\bigcirc)$$

In ATL (actually ATL*):

$$\langle\langle A\rangle\rangle\,\bigcirc\,\mathbf{W}\,\bigcirc \equiv \langle\langle A\rangle\rangle\,\big[\bigcirc\,\mathbf{U}\,\bigcirc \vee \mathbf{G}\,\bigcirc\big]$$
$$\not\equiv \langle\langle A\rangle\rangle\,\bigcirc\,\mathbf{U}\,\bigcirc \vee \langle\langle A\rangle\rangle\,\mathbf{G}\,\bigcirc$$

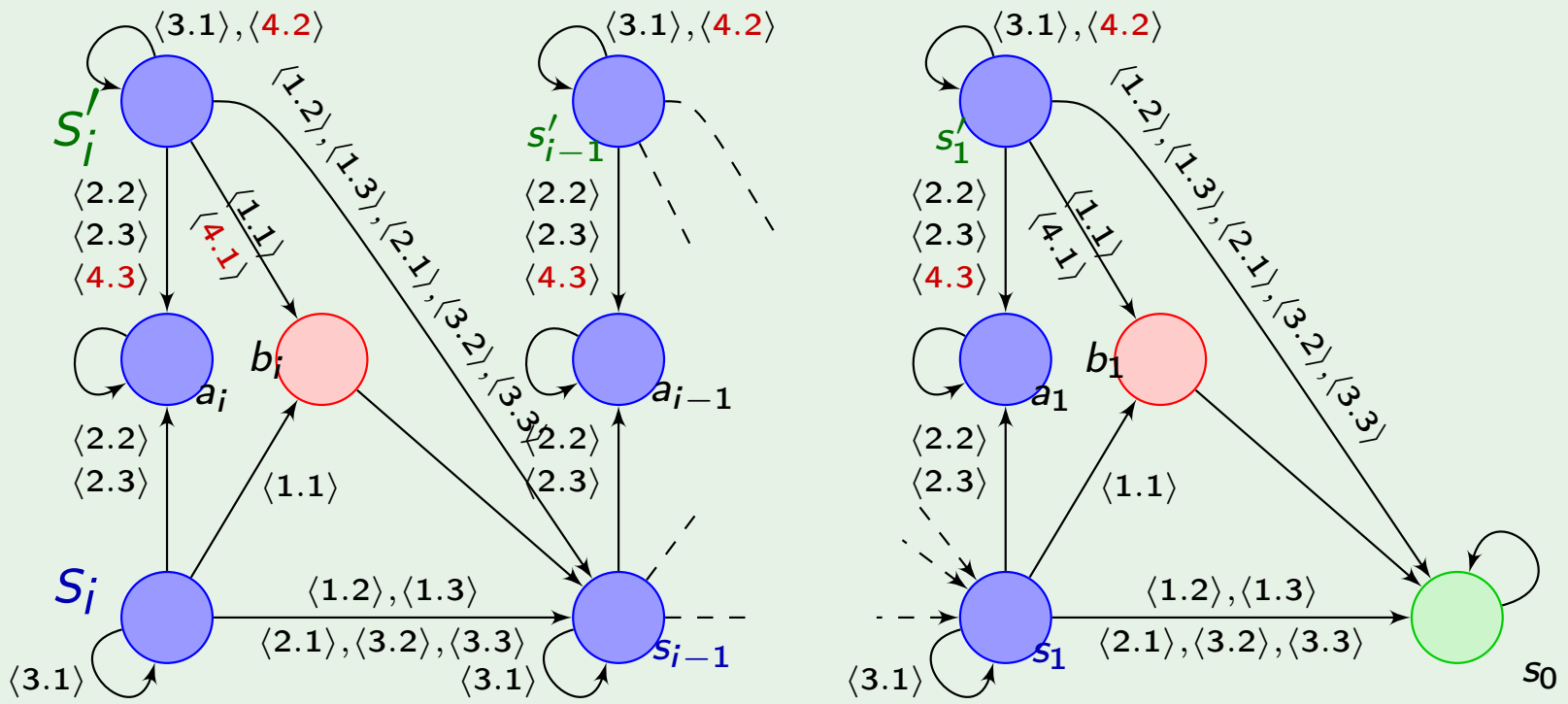


Expressiveness of ATL

Theorem

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

Proof

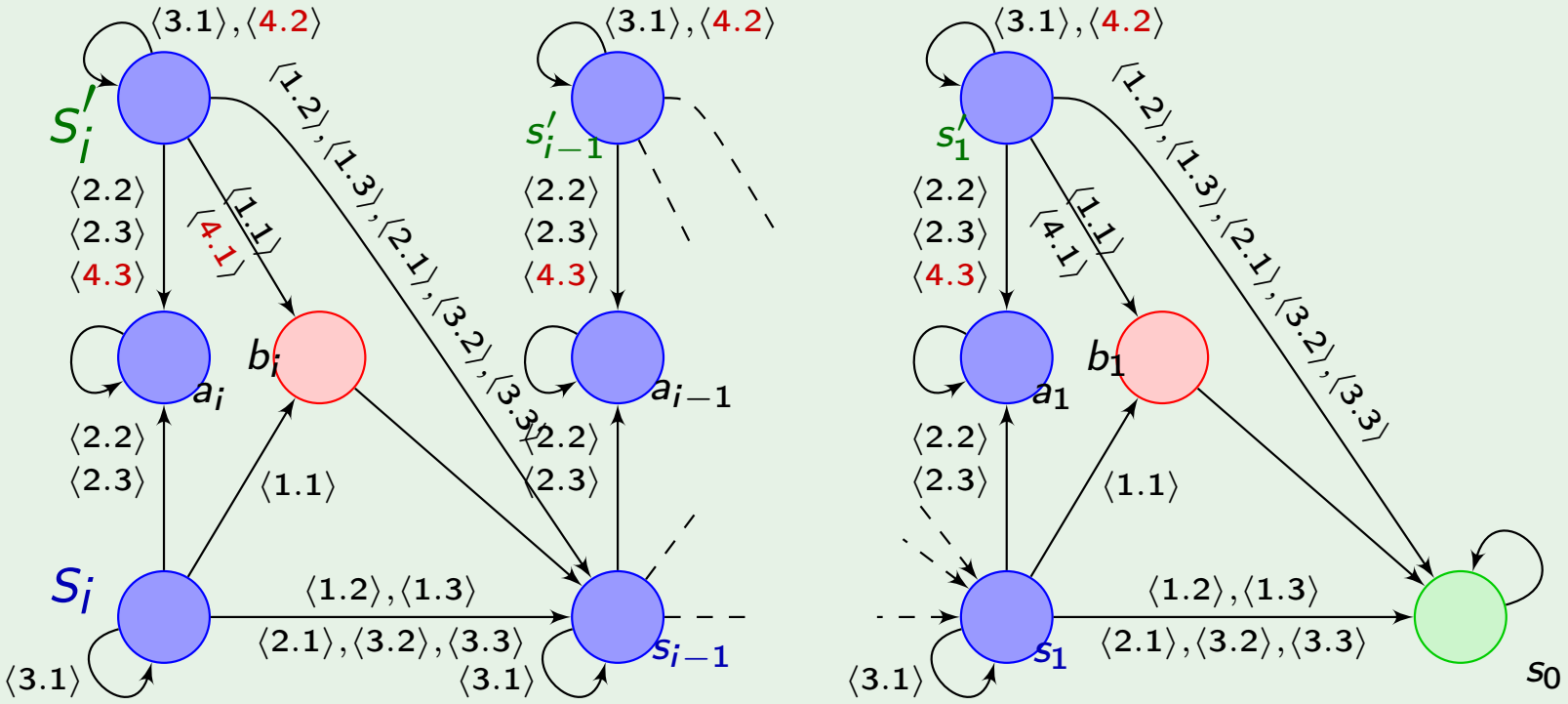


Expressiveness of ATL

Theorem

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

Proof



$s_i \not\models \langle\langle A\rangle\rangle$ ● $\mathbf{W}$ ○ but $s_i' \models \langle\langle A\rangle\rangle$ ● $\mathbf{W}$ ○

Expressiveness of ATL

Theorem

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

Remark

Turn-based games are determined:

$$\neg\,(\,\langle\langle A \rangle\rangle\, \varphi) \equiv \langle\langle \mathrm{Agt} \setminus A \rangle\rangle\, \neg\, \varphi.$$

Expressiveness of ATL

Theorem

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

Remark

Turn-based games are determined:

$$\neg(\langle\langle A \rangle\rangle\, \varphi) \equiv \langle\langle \mathrm{Agt} \setminus A \rangle\rangle\, \neg\varphi.$$

Then:

$$\langle\langle A \rangle\rangle\, \bigcirc\!\!\!\!\bullet\ \mathbf{W}\ \bigcirc \equiv \langle\langle A \rangle\rangle\, \neg((\neg \bigcirc)\ \mathbf{U}\ (\neg \bigcirc \wedge \neg \bigcirc\!\!\!\!\bullet))$$

Expressiveness of ATL

Theorem

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

Remark

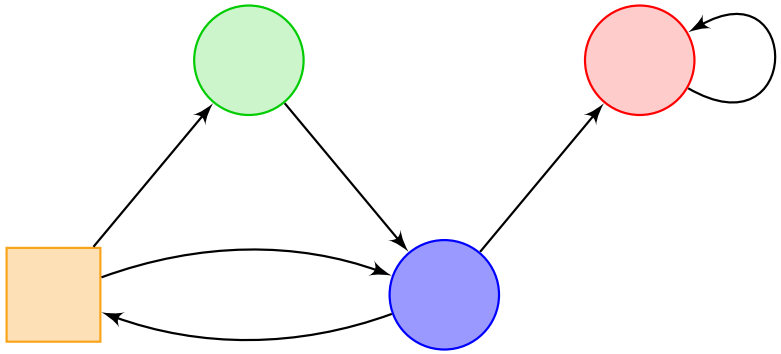
Turn-based games are determined:

$$\neg(\langle\langle A \rangle\rangle\, \varphi) \equiv \langle\langle \mathrm{Agt} \setminus A \rangle\rangle\, \neg\varphi.$$

Then:

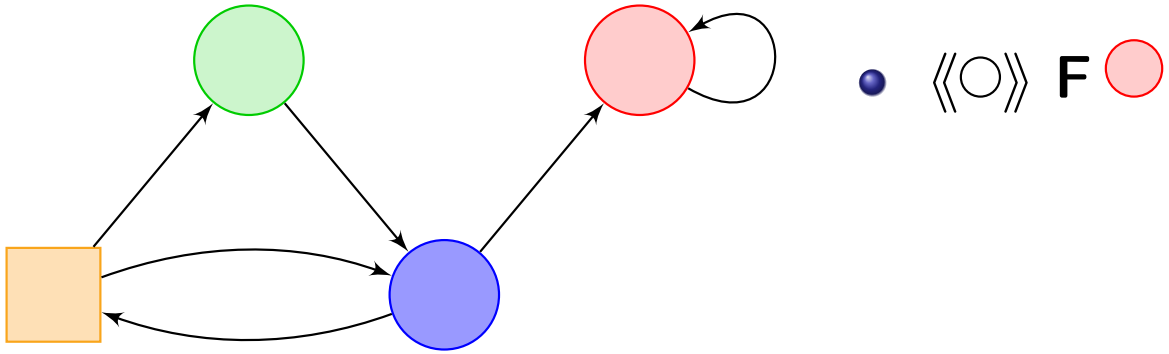
$$\begin{aligned}\langle\langle A \rangle\rangle\, \bigcirc\!\!\!\!\bullet\ \mathbf{W}\ \bigcirc &\equiv \langle\langle A \rangle\rangle\, \neg((\neg\bigcirc)\ \mathbf{U}\ (\neg\bigcirc \wedge \neg\bigcirc\!\!\!\!\bullet)) \\ &\equiv_{\text{t.b.}} \neg\, \langle\langle \mathrm{Agt} \setminus A \rangle\rangle\, (\neg\bigcirc)\ \mathbf{U}\ (\neg\bigcirc \wedge \neg\bigcirc\!\!\!\!\bullet)\end{aligned}$$

Model checking ATL [AHK02]



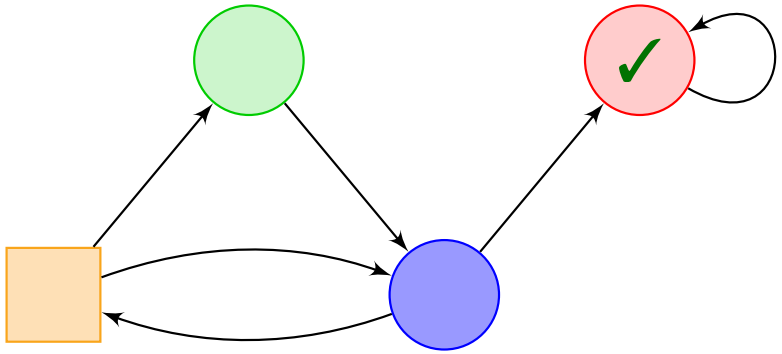
[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Model checking ATL [AHK02]



[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

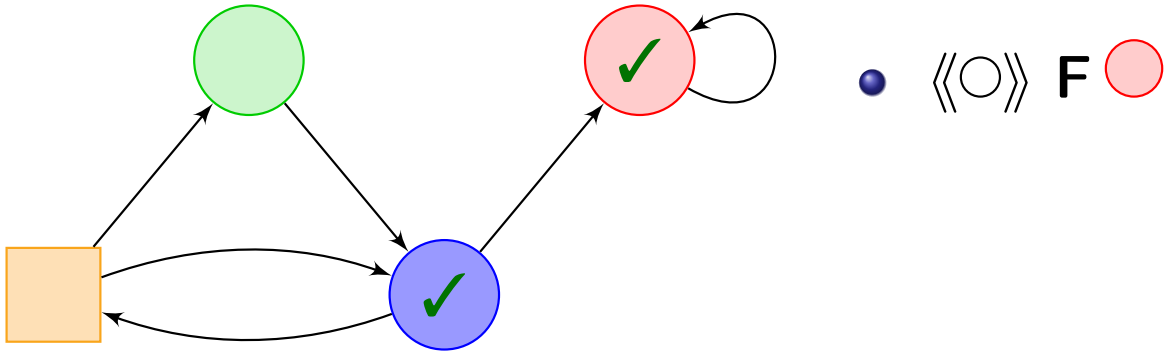
Model checking ATL [AHK02]



- $\langle\langle \bigcirc \rangle\rangle$ **F**

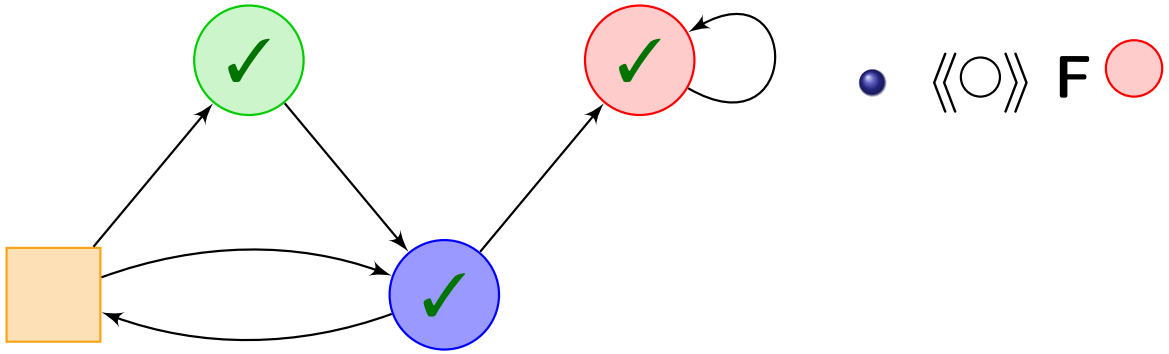
[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Model checking ATL [AHK02]



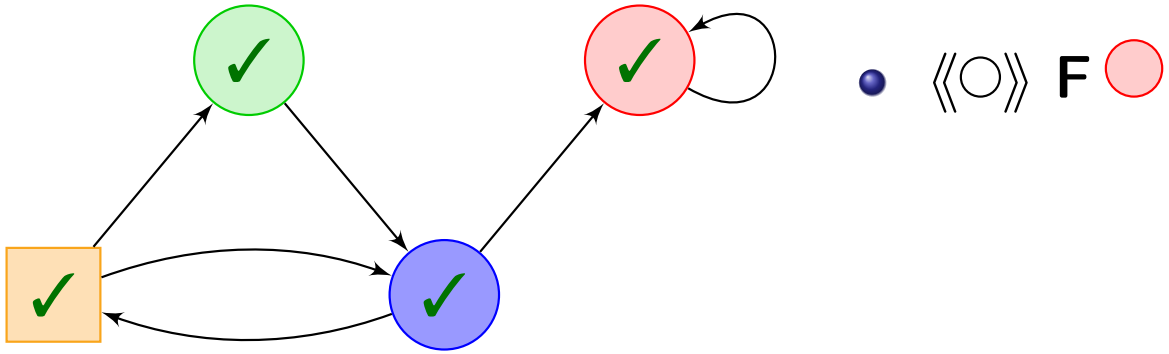
[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Model checking ATL [AHK02]



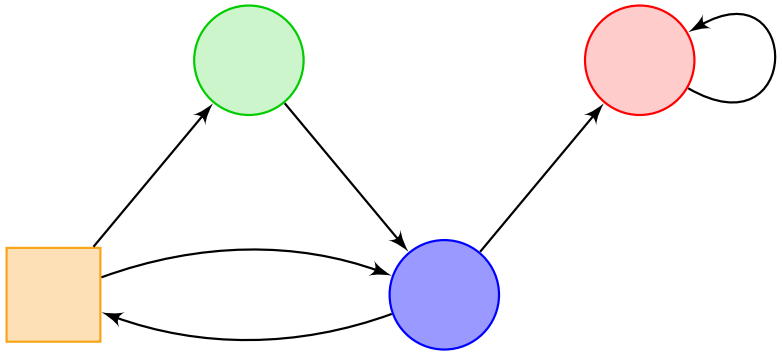
[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Model checking ATL [AHK02]



[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

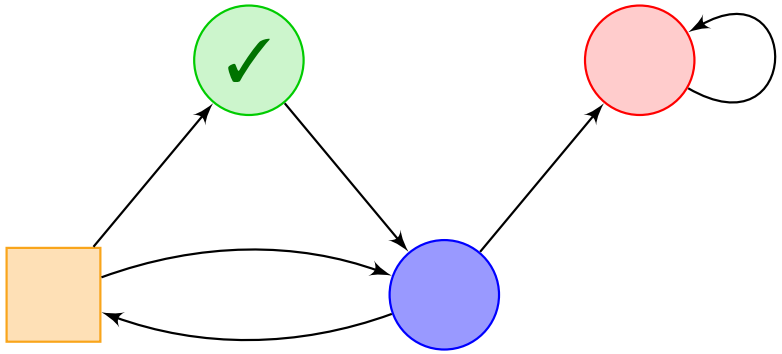
Model checking ATL [AHK02]



- $\langle\langle \bigcirc \rangle\rangle$ **F** 
- $\langle\langle \Box \rangle\rangle$ **F** 

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

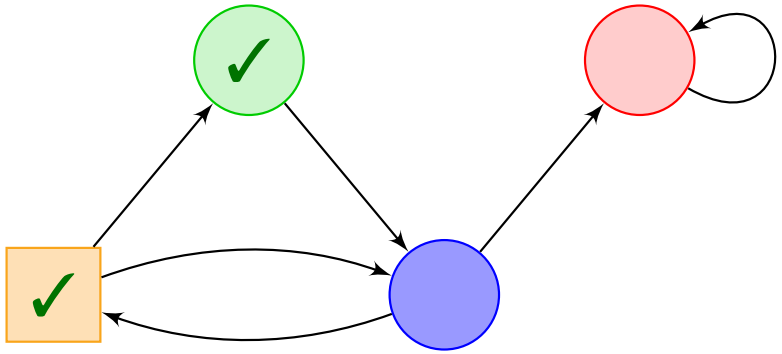
Model checking ATL [AHK02]



- $\langle\langle \bigcirc \rangle\rangle$ **F**
- $\langle\langle \Box \rangle\rangle$ **F**

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

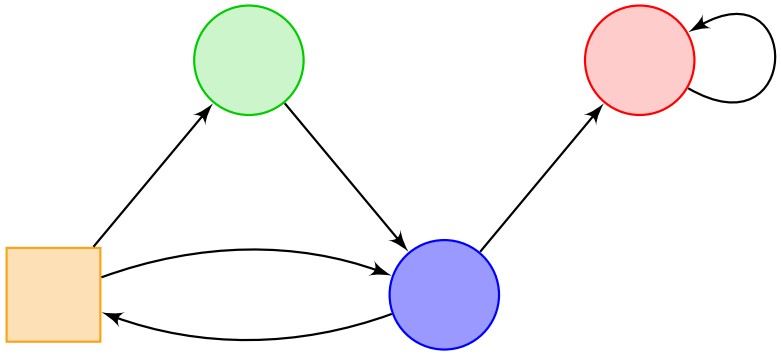
Model checking ATL [AHK02]



- $\langle\langle \bigcirc \rangle\rangle$ **F**
- $\langle\langle \Box \rangle\rangle$ **F**

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

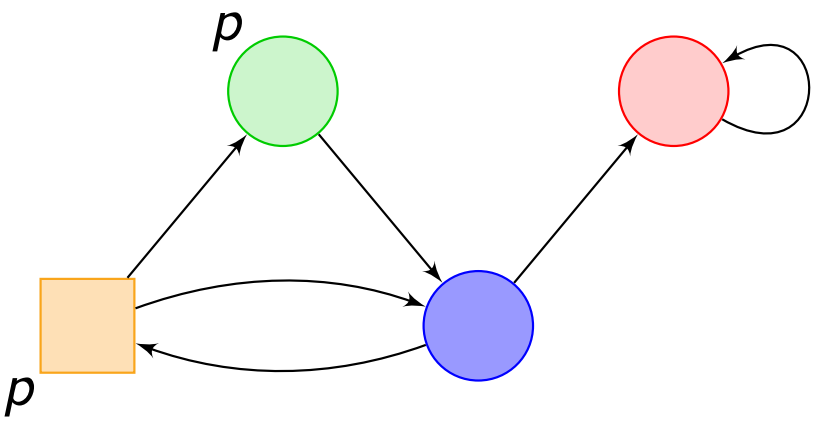
Model checking ATL [AHK02]



- $\langle\langle\bigcirc\rangle\rangle\ \mathbf{F}$ ●
- $\langle\langle\Box\rangle\rangle\ \mathbf{F}$ ●
- $\langle\langle\bigcirc\rangle\rangle\ \mathbf{G}(\ \langle\langle\Box\rangle\rangle\ \mathbf{F}$ ●$)$

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

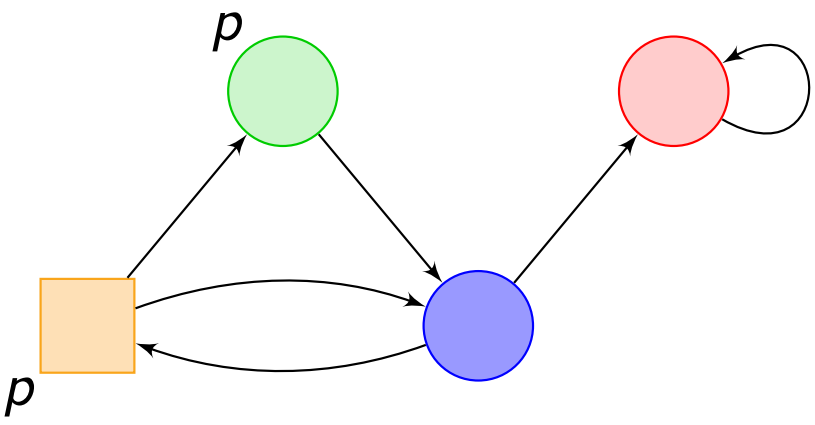
Model checking ATL [AHK02]



- $\langle\langle \bigcirc \rangle\rangle\ \mathbf{F}$
- $\langle\langle \Box \rangle\rangle\ \mathbf{F}$
- $\langle\langle \bigcirc \rangle\rangle\ \mathbf{G}(\underbrace{\langle\langle \Box \rangle\rangle\ \mathbf{F}}_{p})$

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

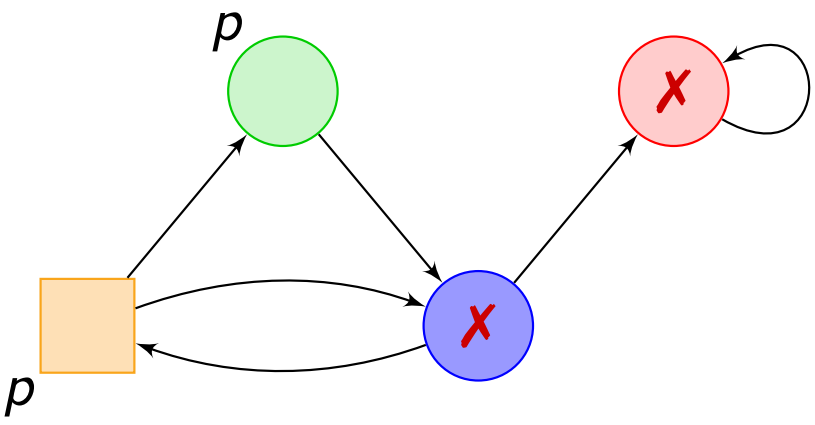
Model checking ATL [AHK02]



- $\langle\langle \bigcirc \rangle\rangle\ \mathbf{F}$
- $\langle\langle \Box \rangle\rangle\ \mathbf{F}$
- $\langle\langle \bigcirc \rangle\rangle\ \mathbf{G}(\underbrace{\langle\langle \Box \rangle\rangle\ \mathbf{F}}_{p}) \equiv \langle\langle \bigcirc \rangle\rangle\ \mathbf{G}\ p$

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

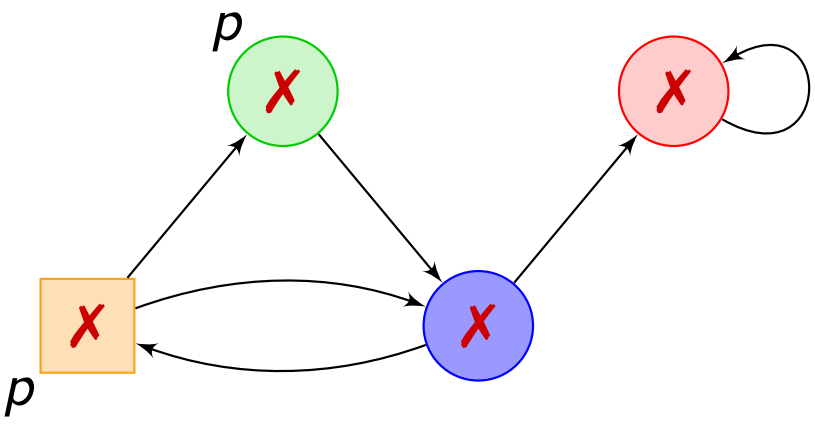
Model checking ATL [AHK02]



- $\langle\langle \bigcirc \rangle\rangle\ \mathbf{F}$
- $\langle\langle \Box \rangle\rangle\ \mathbf{F}$
- $\langle\langle \bigcirc \rangle\rangle\ \mathbf{G}(\underbrace{\langle\langle \Box \rangle\rangle\ \mathbf{F}}_{p}) \equiv \langle\langle \bigcirc \rangle\rangle\ \mathbf{G}\ p$

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

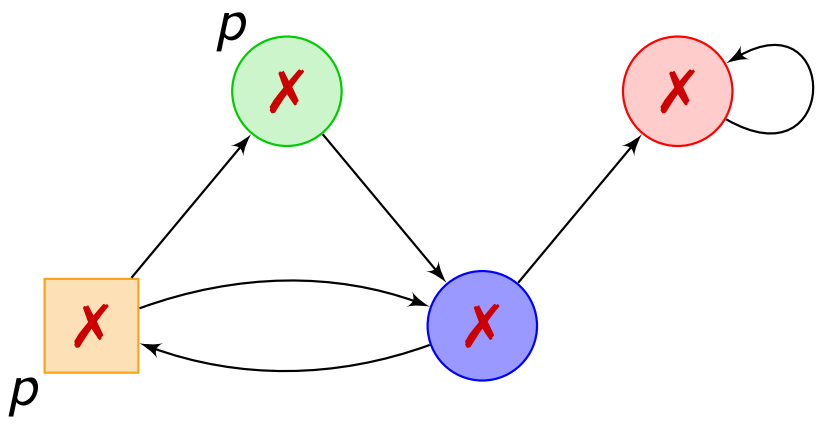
Model checking ATL [AHK02]



- $\langle\langle \bigcirc \rangle\rangle\ \mathbf{F}$ ●
- $\langle\langle \Box \rangle\rangle\ \mathbf{F}$ ●
- $\langle\langle \bigcirc \rangle\rangle\ \mathbf{G}(\underbrace{\langle\langle \Box \rangle\rangle\ \mathbf{F}\ \bigcirc}_{p}) \equiv \langle\langle \bigcirc \rangle\rangle\ \mathbf{G}\ p$

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Model checking ATL [AHK02]



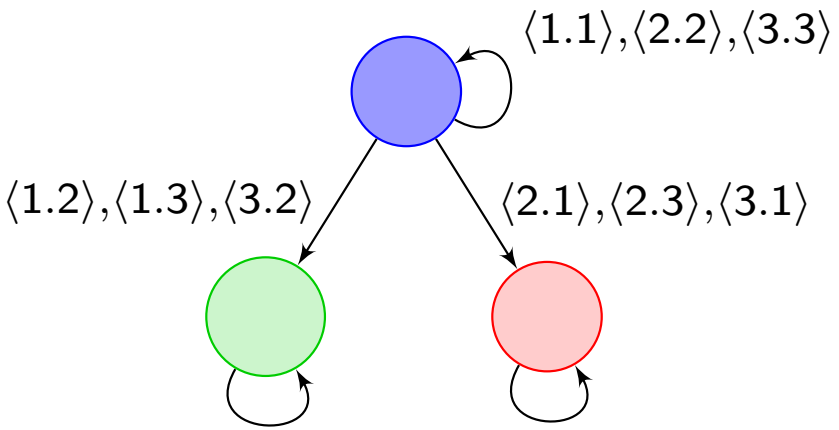
- $\langle\langle \bigcirc \rangle\rangle\ \mathbf{F}$
- $\langle\langle \Box \rangle\rangle\ \mathbf{F}$
- $\langle\langle \bigcirc \rangle\rangle\ \mathbf{G}(\underbrace{\langle\langle \Box \rangle\rangle\ \mathbf{F}}_{p}) \equiv \langle\langle \bigcirc \rangle\rangle\ \mathbf{G}\ p$

Theorem ([AHK02])

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

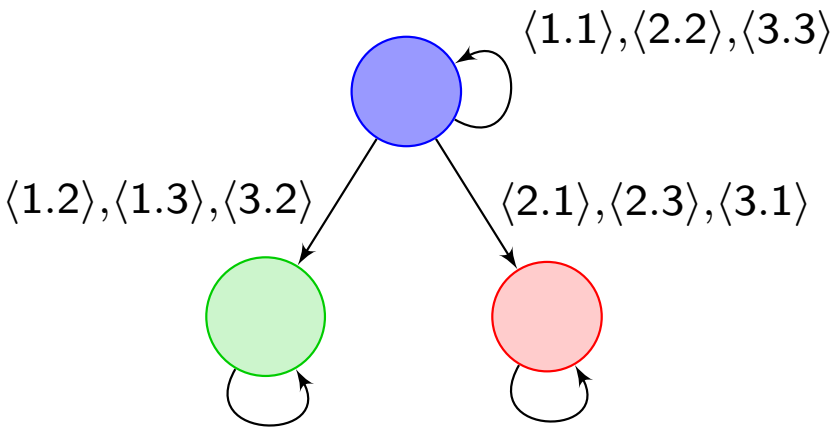
[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

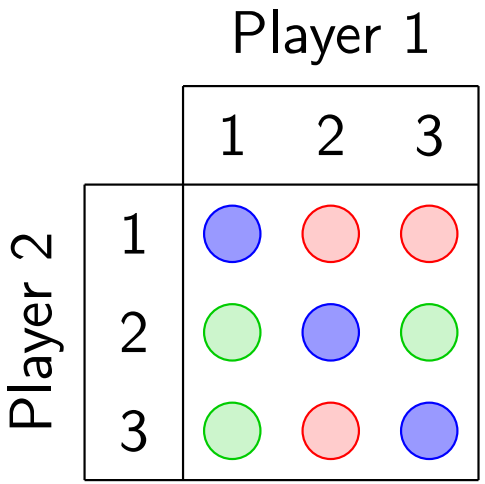
Model checking ATL [AHK02]



[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

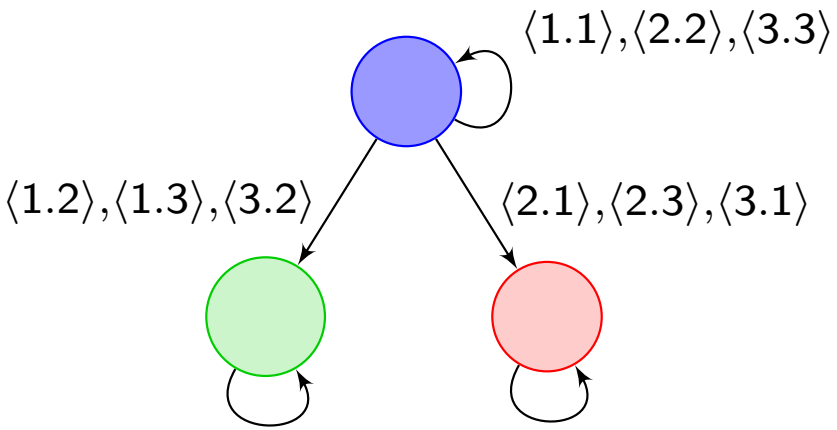
Model checking ATL [AHK02]

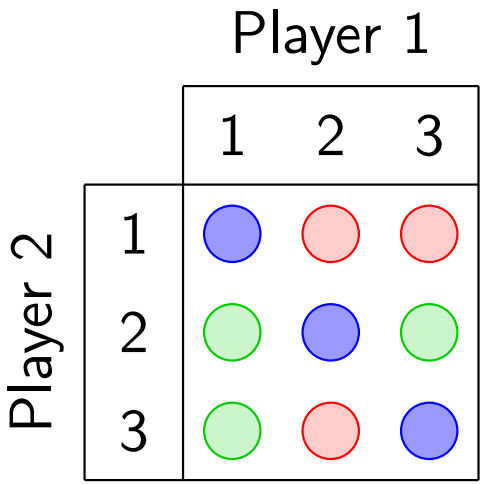




[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Model checking ATL [AHK02]

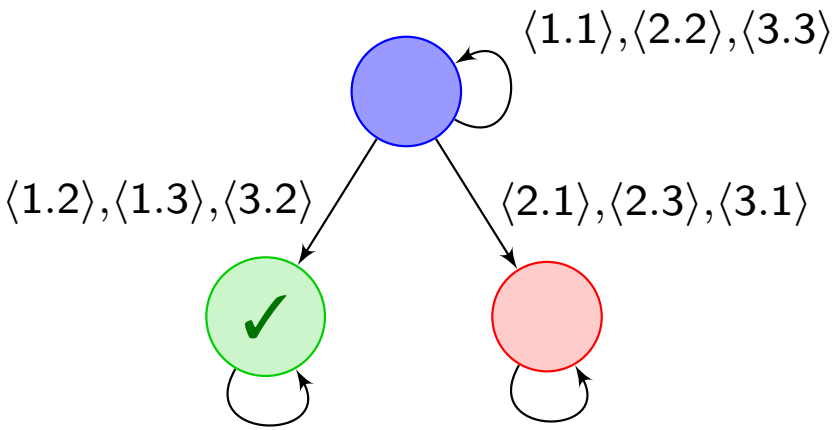


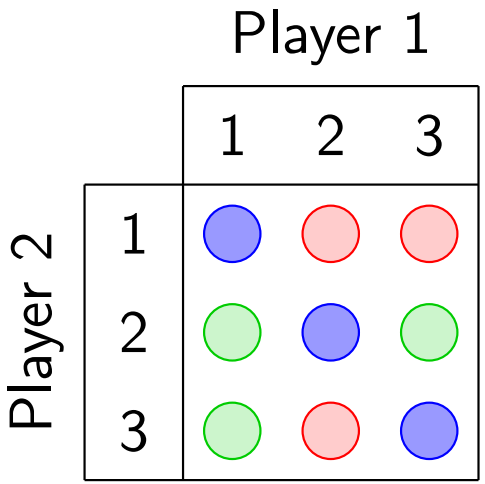


$\langle\langle 1 \rangle\rangle$ **F** 

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Model checking ATL [AHK02]

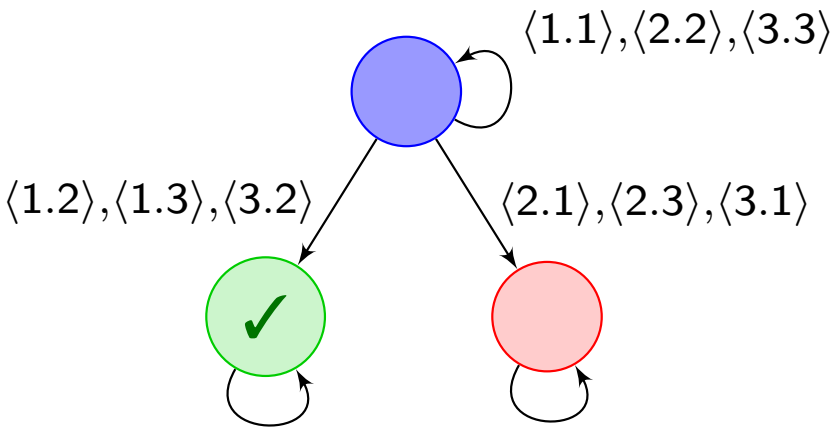


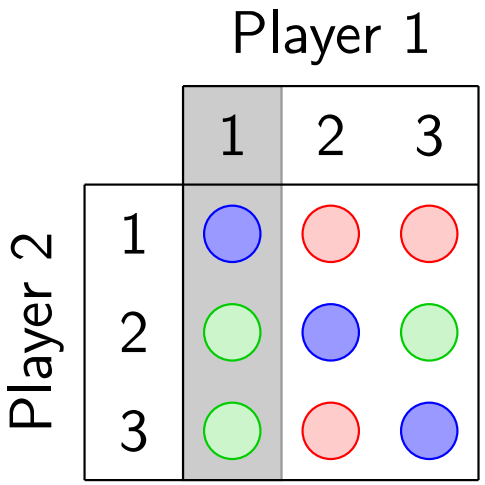


$\langle\langle 1 \rangle\rangle$ **F** ●

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Model checking ATL [AHK02]

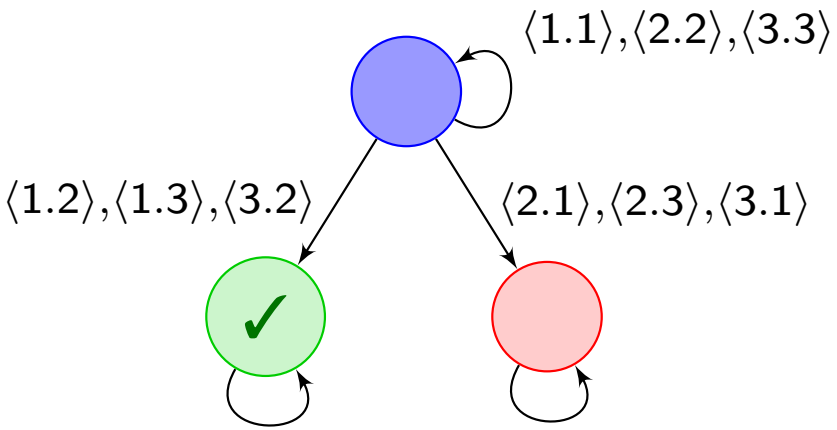


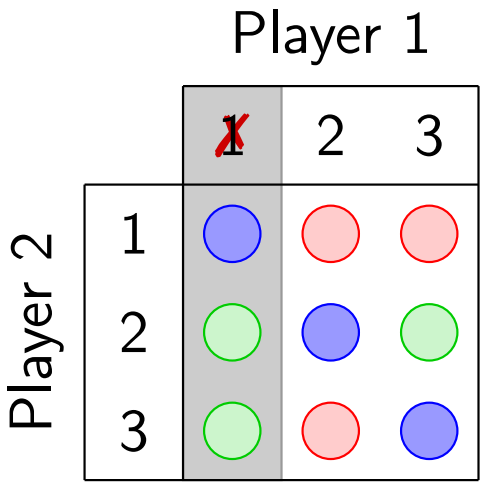


$\langle\langle 1 \rangle\rangle$ **F** ●

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Model checking ATL [AHK02]

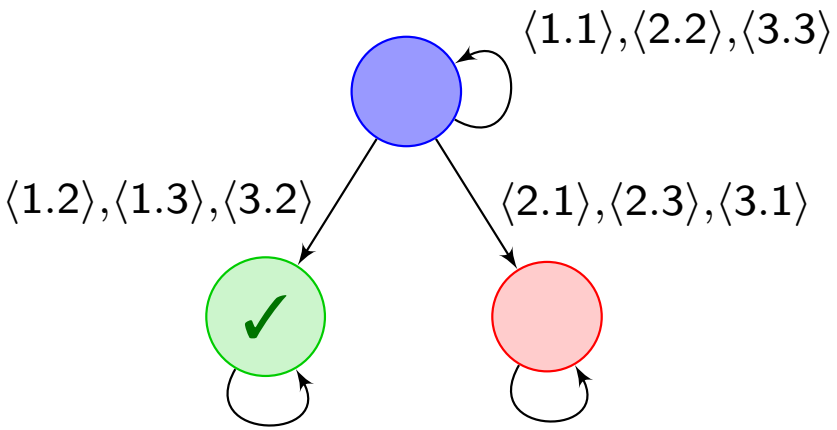


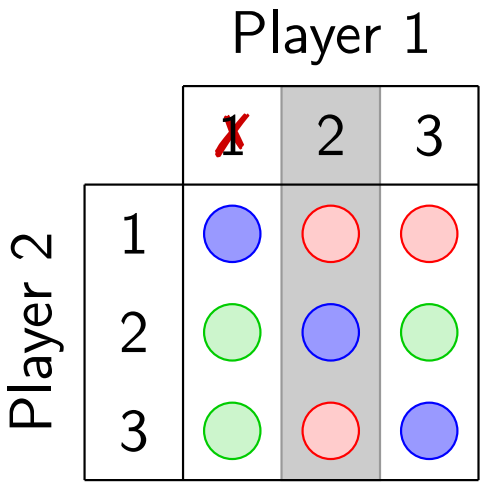


$\langle\langle 1 \rangle\rangle$ **F**

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Model checking ATL [AHK02]

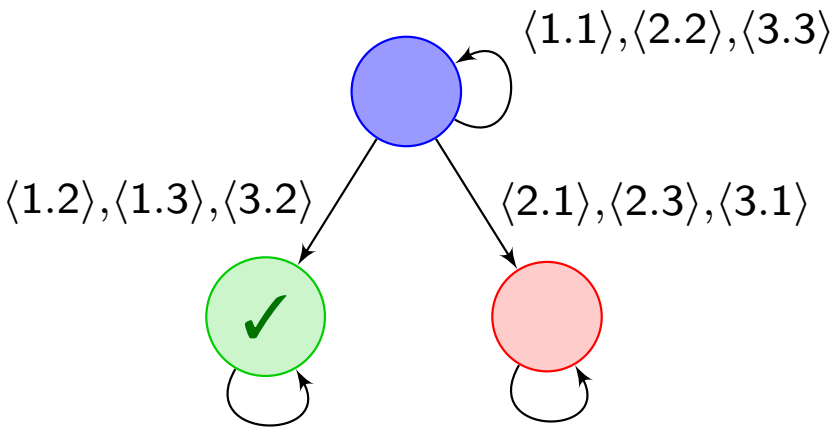


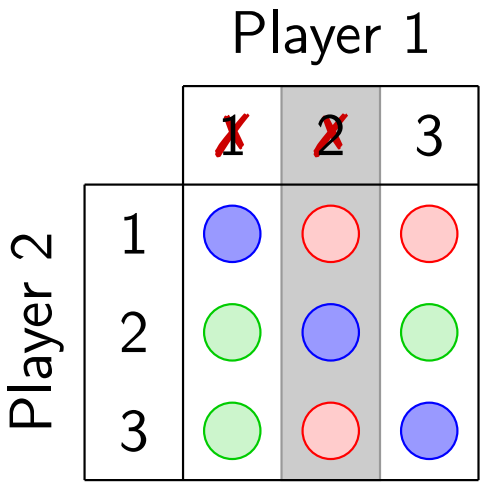


$\langle\langle 1 \rangle\rangle$ **F**

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Model checking ATL [AHK02]

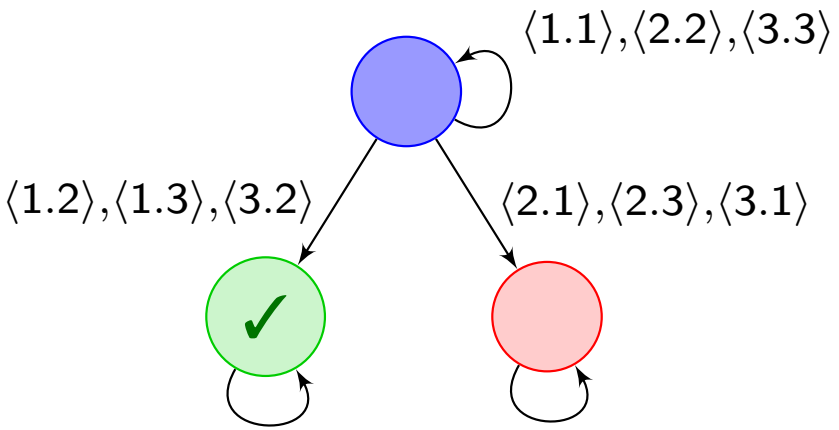


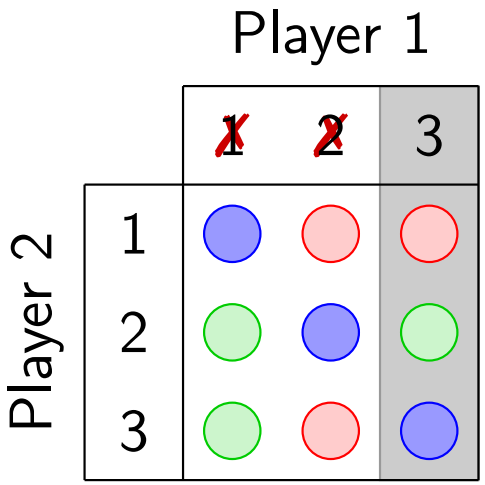


$\langle\langle 1 \rangle\rangle$ **F** ●

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Model checking ATL [AHK02]

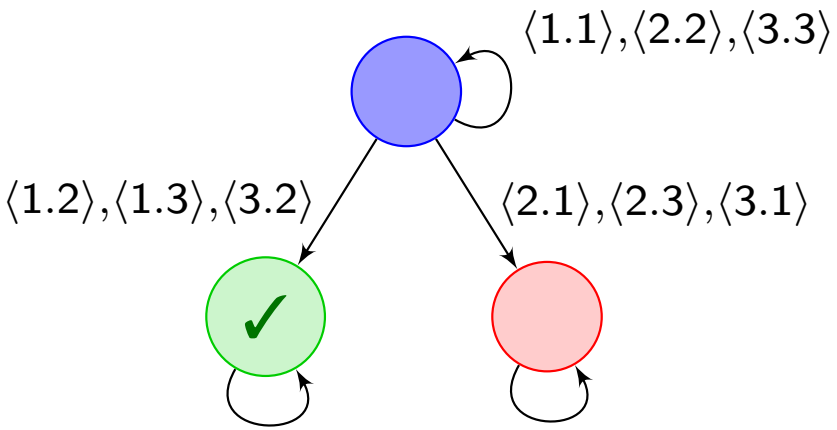


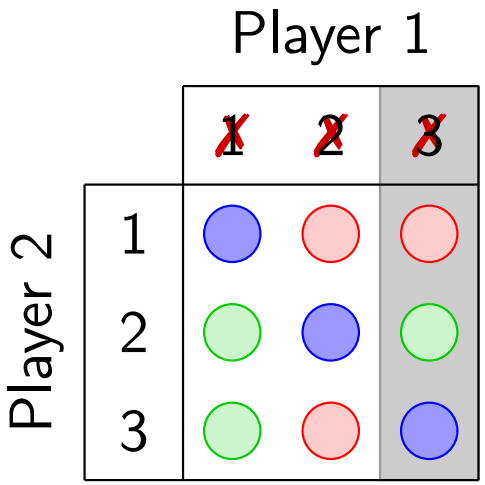


$\langle\langle 1 \rangle\rangle$ **F** ●

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Model checking ATL [AHK02]

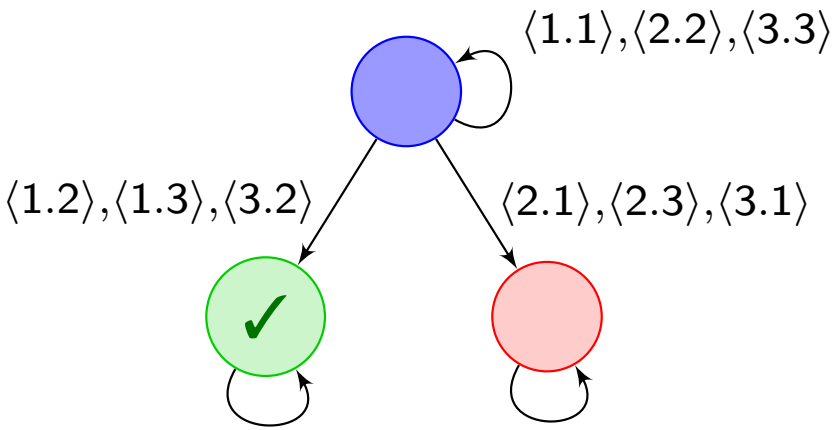


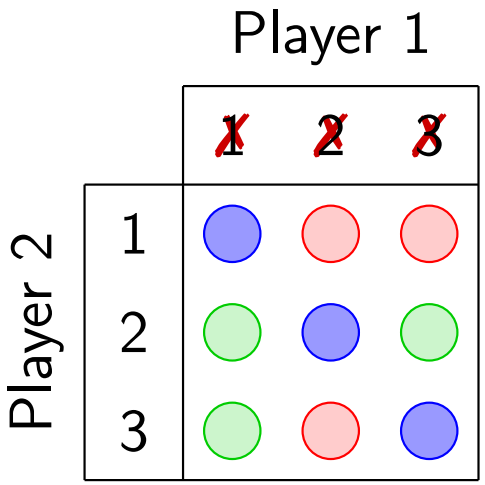


$\langle\langle 1 \rangle\rangle$ **F** ●

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Model checking ATL [AHK02]





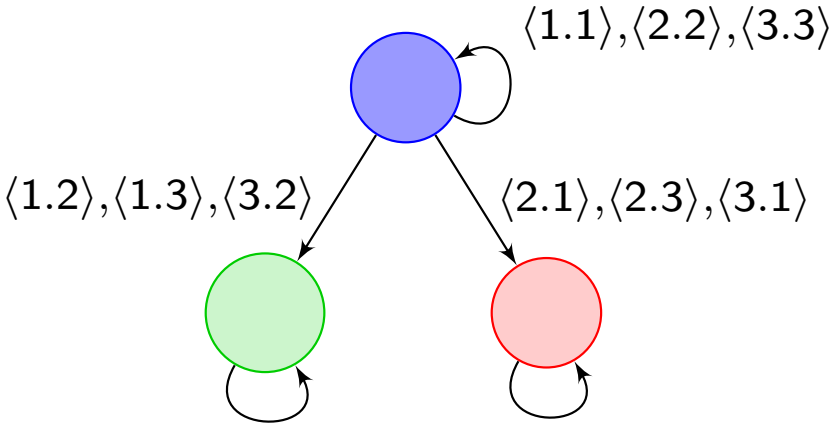
$\langle\langle 1 \rangle\rangle$ **F**

Theorem ([AHK02])

Model checking ATL over concurrent games is PTIME*-complete.*

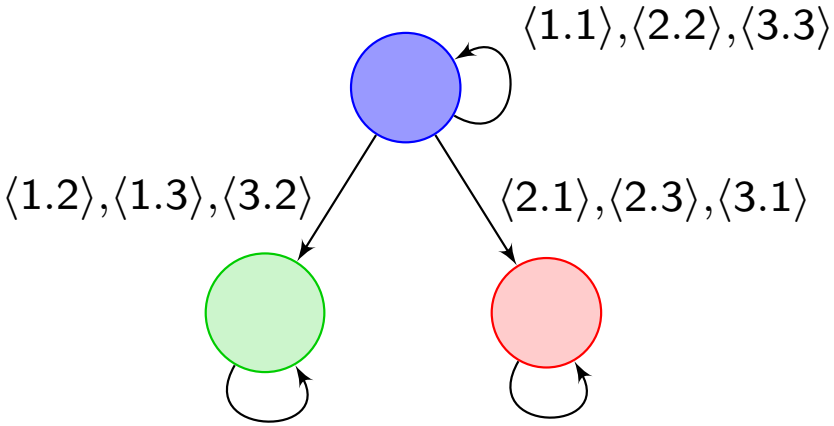
[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Model checking ATL [AHK02]



[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.
[LMO07] Laroussinie, Markey, Oreiby. On the Expressiveness and Complexity of ATL. CONCUR, 2007.

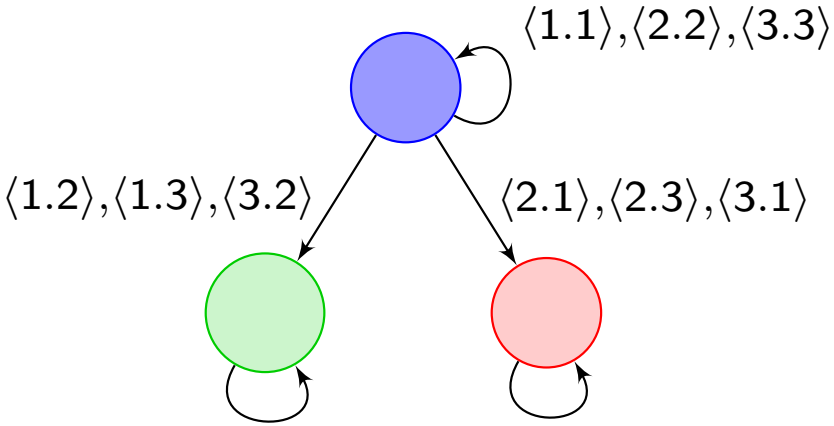
Model checking ATL [AHK02]



- if $m_1 = m_2$ goto ;
- otherwise, if $m_1 = 1$ or $m_2 = 2$ goto ;
- otherwise, goto ;

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.
[LMO07] Laroussinie, Markey, Oreiby. On the Expressiveness and Complexity of ATL. CONCUR, 2007.

Model checking ATL [AHK02]

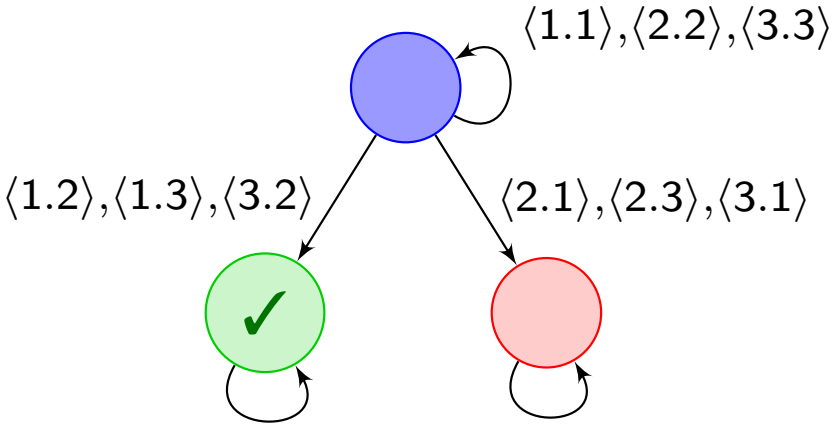


- if $m_1 = m_2$ goto ;
- otherwise, if $m_1 = 1$ or $m_2 = 2$ goto ;
- otherwise, goto ;

$\langle\langle 1 \rangle\rangle$ **F**

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.
[LMO07] Laroussinie, Markey, Oreiby. On the Expressiveness and Complexity of ATL. CONCUR, 2007.

Model checking ATL [AHK02]

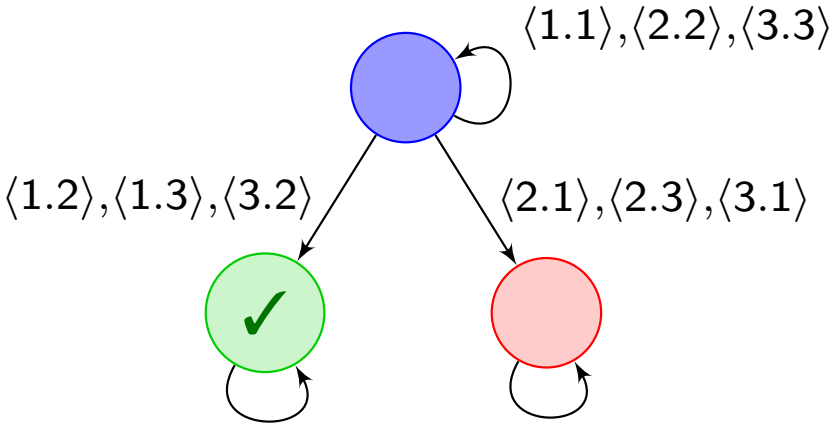


- if $m_1 = m_2$ goto ;
- otherwise, if $m_1 = 1$ or $m_2 = 2$ goto ;
- otherwise, goto ;

$\langle\langle 1 \rangle\rangle$ **F**

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.
[LMO07] Laroussinie, Markey, Oreiby. On the Expressiveness and Complexity of ATL. CONCUR, 2007.

Model checking ATL [AHK02]



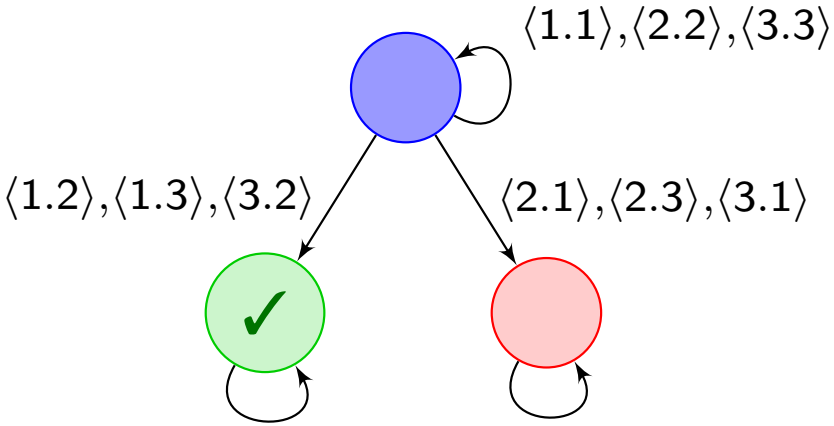
- if $m_1 = m_2$ goto ;
- otherwise, if $m_1 = 1$ or $m_2 = 2$ goto ;
- otherwise, goto ;

$\langle\langle 1 \rangle\rangle$ **F**

$\rightsquigarrow$ for each move of Player 1, need to solve several instances of SAT

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.
[LMO07] Laroussinie, Markey, Oreiby. On the Expressiveness and Complexity of ATL. CONCUR, 2007.

Model checking ATL [AHK02]



- if $m_1 = m_2$ goto ●;
- otherwise, if $m_1 = 1$ or $m_2 = 2$ goto ●;
- otherwise, goto ●;

$\langle\langle 1 \rangle\rangle$ **F** ●

$\rightsquigarrow$ for each move of Player 1, need to solve several instances of SAT

Theorem ([LMO07])

Model checking ATL over concurrent games with symbolic transition table is Δ_3^P-complete (hence in PSPACE, NP*-hard).*

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.
[LMO07] Laroussinie, Markey, Oreiby. On the Expressiveness and Complexity of ATL. CONCUR, 2007.

The full logic ATL*

The full logic ATL*

$$\langle\langle A \rangle\rangle \left[\mathbf{G}\,\mathbf{F}\, \boxed{\mathsf{serv}_i} \Rightarrow \mathbf{G}\big(\boxed{\mathsf{req}_0} \Rightarrow \mathbf{F}\, \boxed{\mathsf{serv}_0} \big) \right]$$

The full logic ATL*

$$\langle\langle A \rangle\rangle \left[\mathbf{G}\,\mathbf{F}\, \boxed{\mathsf{serv}_i} \Rightarrow \mathbf{G}(\boxed{\mathsf{req}_0} \Rightarrow \mathbf{F}\, \boxed{\mathsf{serv}_0}) \right]$$

Theorem ([AHK02])

Model checking ATL is 2EXPTIME-complete.*

[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

The full logic ATL*

$$\langle\langle A \rangle\rangle \left[\mathbf{G}\,\mathbf{F}\,\boxed{\mathsf{serv}_i} \Rightarrow \mathbf{G}(\boxed{\mathsf{req}_0} \Rightarrow \mathbf{F}\,\boxed{\mathsf{serv}_0}) \right]$$

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 $O(|Q|^{|p|})$.

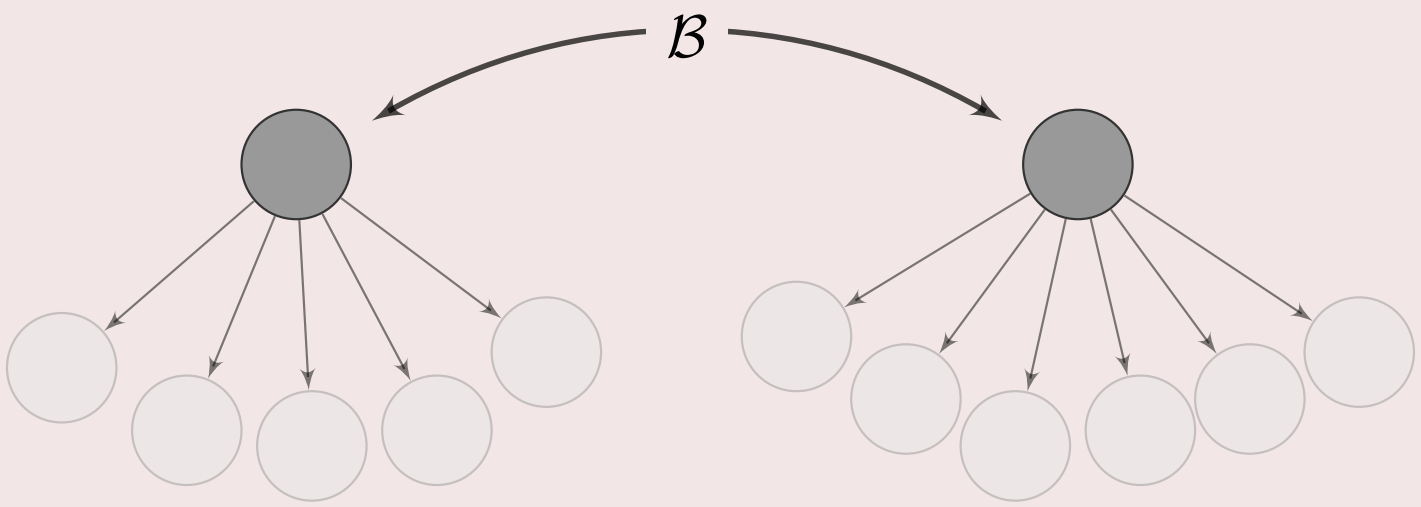
[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Behavioural equivalence for ATL

Behavioural equivalence for ATL

Definition

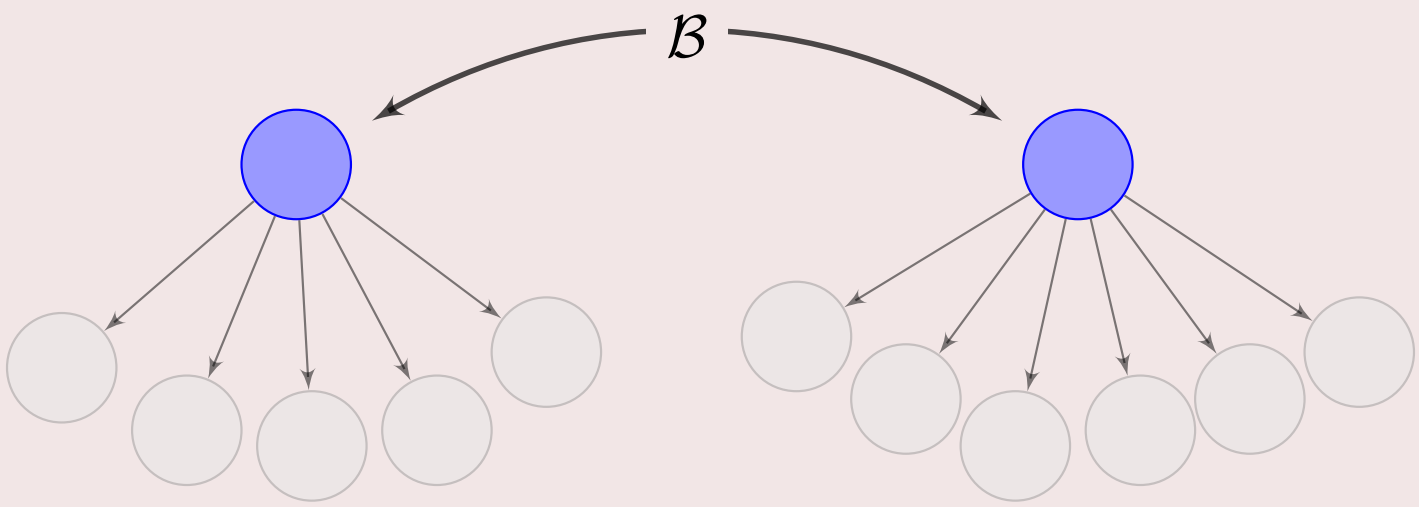
A relation $\mathcal{B} \subseteq Q \times Q$ is an alternating bisimultion if



Behavioural equivalence for ATL

Definition

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

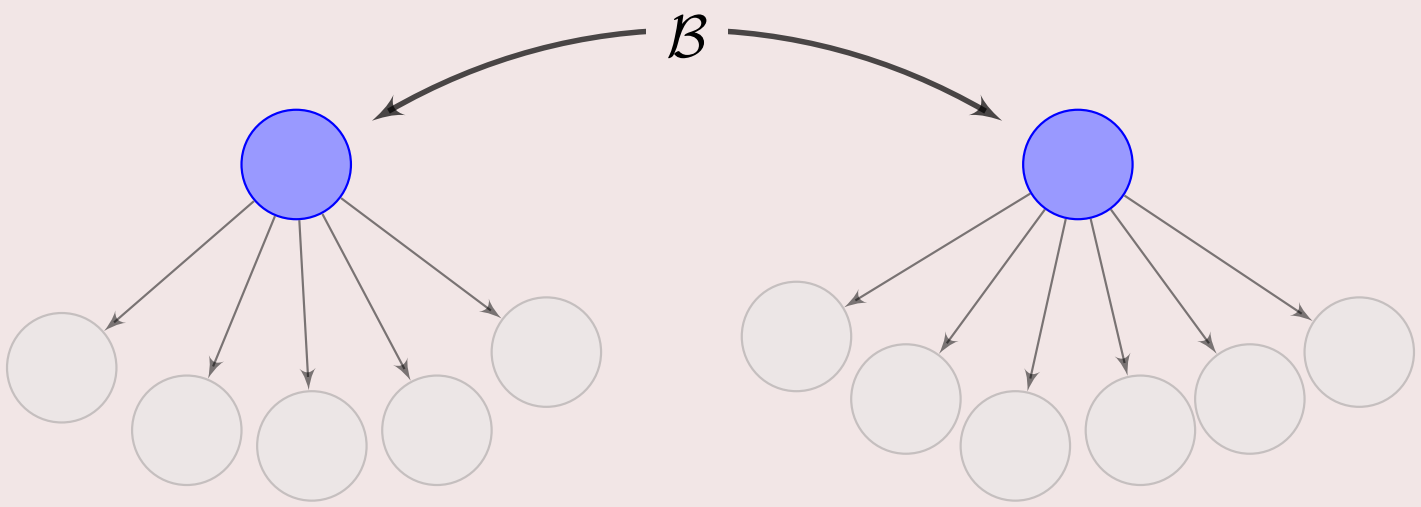


- same label;

Behavioural equivalence for ATL

Definition

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

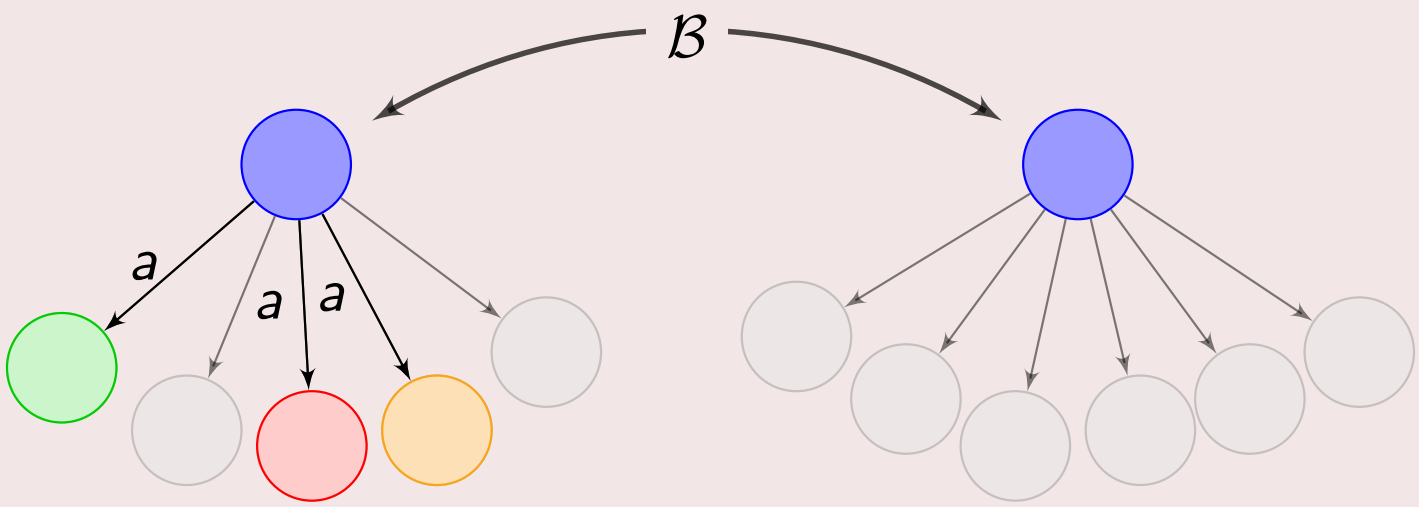


- same label;
- any action on one side has a corresponding action on other side.

Behavioural equivalence for ATL

Definition

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

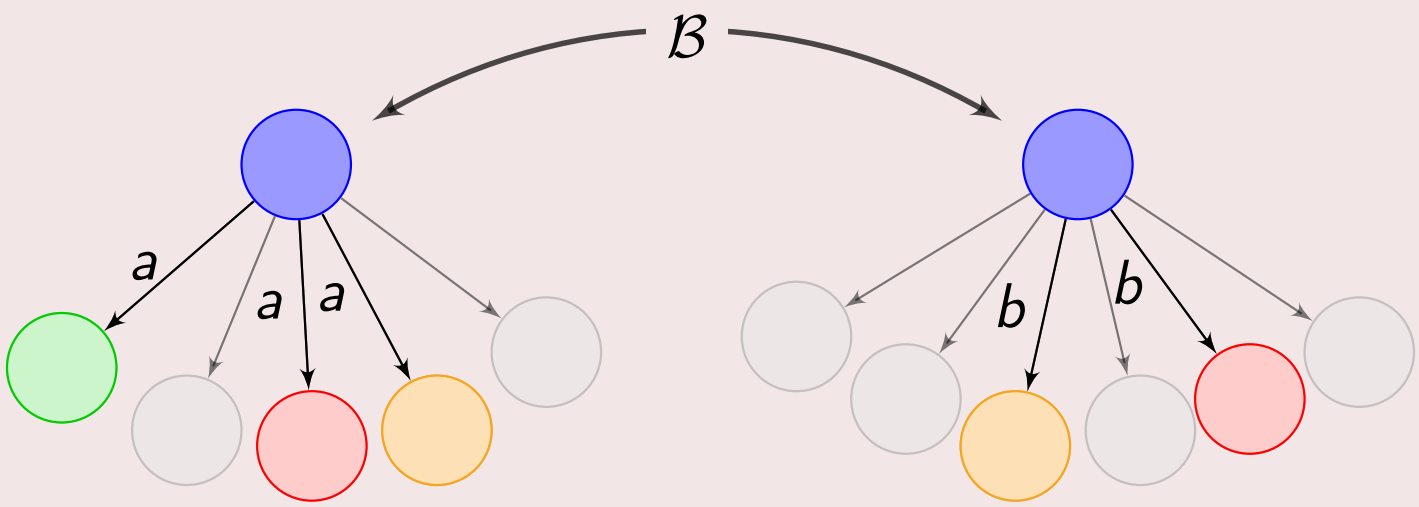


- same label;
- any action on one side has a corresponding action on other side.

Behavioural equivalence for ATL

Definition

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

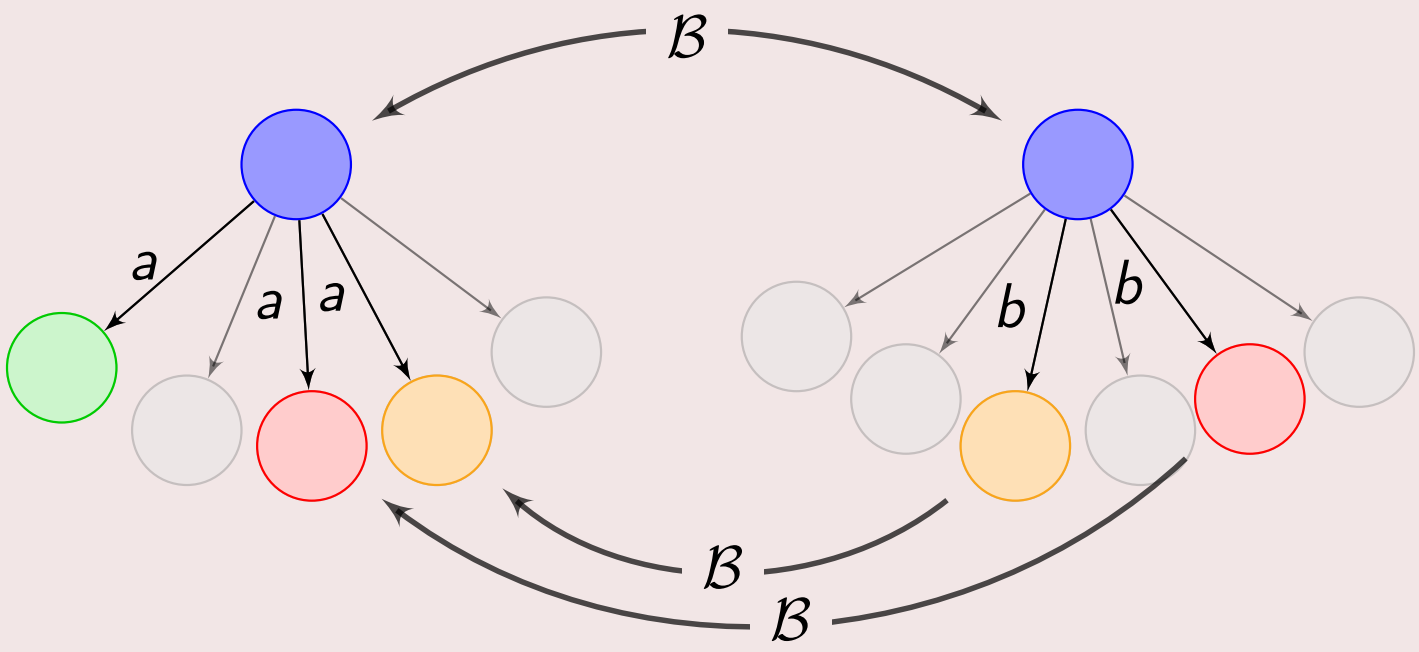


- same label;
- any action on one side has a corresponding action on other side.

Behavioural equivalence for ATL

Definition

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

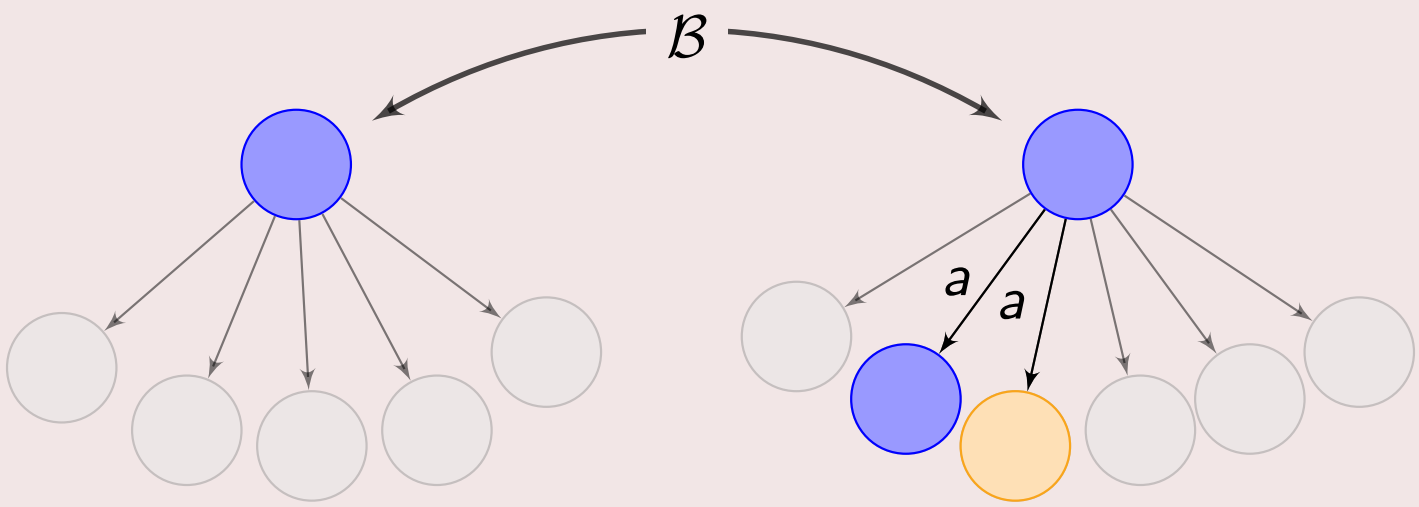


- same label;
- any action on one side has a corresponding action on other side.

Behavioural equivalence for ATL

Definition

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

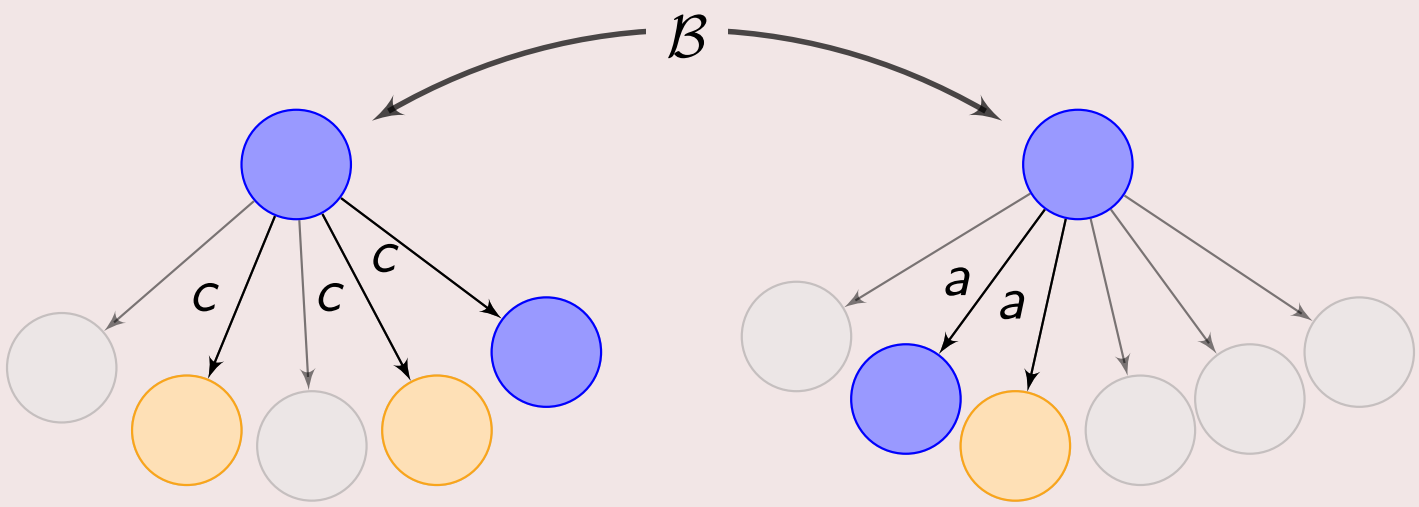


- same label;
- any action on one side has a corresponding action on other side.

Behavioural equivalence for ATL

Definition

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

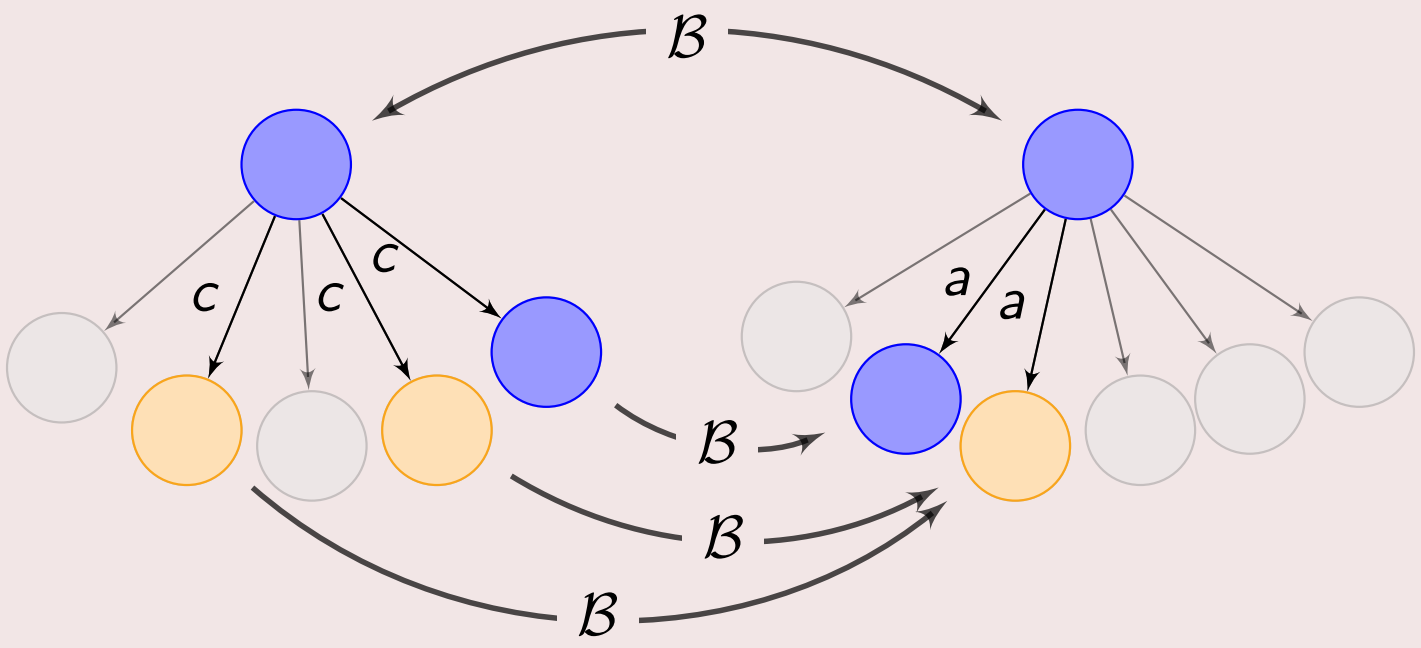


- same label;
- any action on one side has a corresponding action on other side.

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.

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.

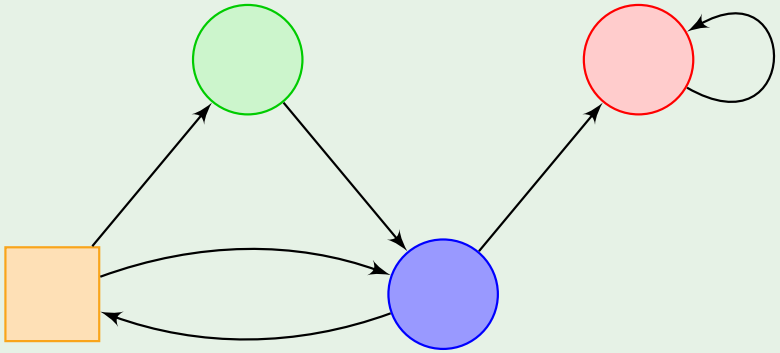
Theorem ([AHKV98])

(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.

[AHKV98] Alur, Henzinger, Kupferman, Vardi. Alternating refinement relations. CONCUR, 1998.

ATL with strategy contexts [BDLM09,DLM10]

Example



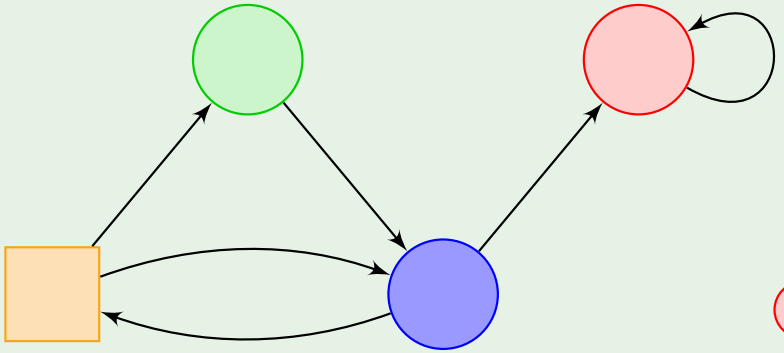
$\langle\langle \bigcirc \rangle\rangle\ \mathbf{G}(\ \langle\langle \Box \rangle\rangle\ \mathbf{F}\ \bigcirc\ )$

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

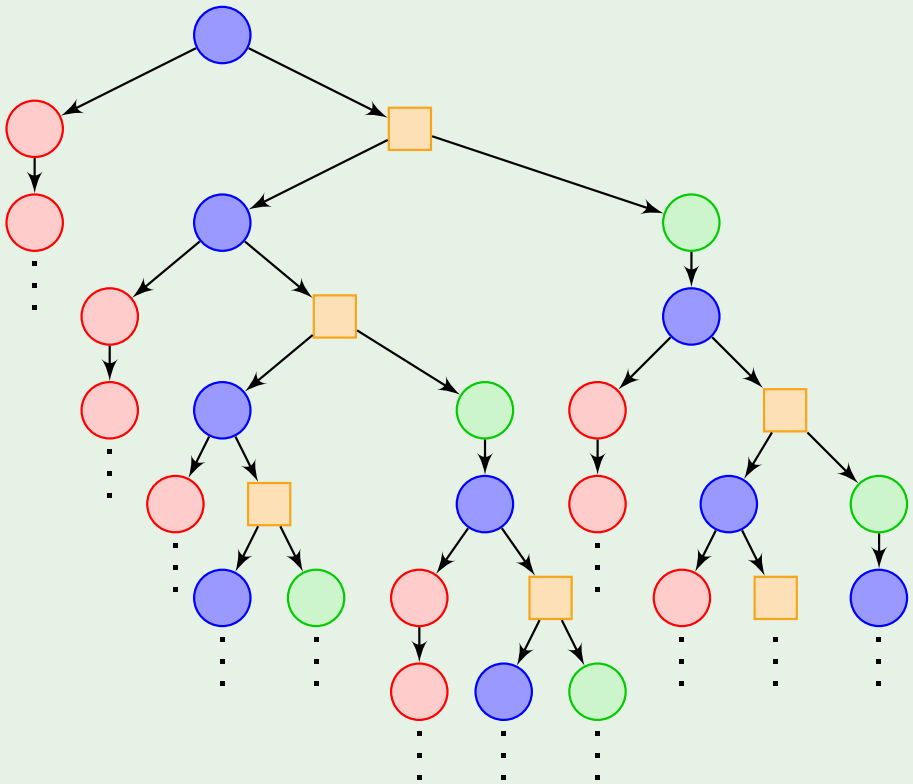
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.

ATL with strategy contexts [BDLM09,DLM10]

Example



$\langle\langle \bigcirc \rangle\rangle\ \mathbf{G}(\ \langle\langle \Box \rangle\rangle\ \mathbf{F}\ \bigcirc\ )$

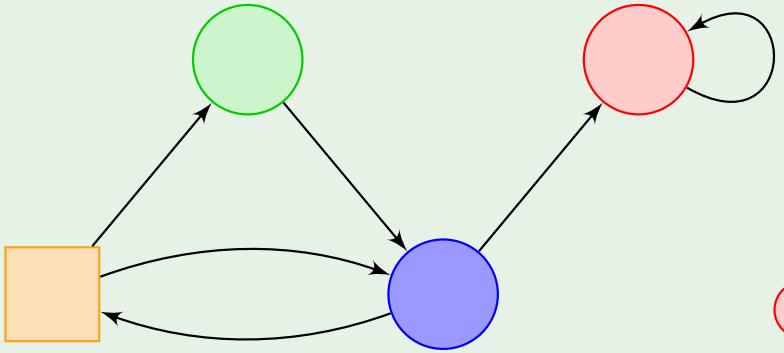


Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.

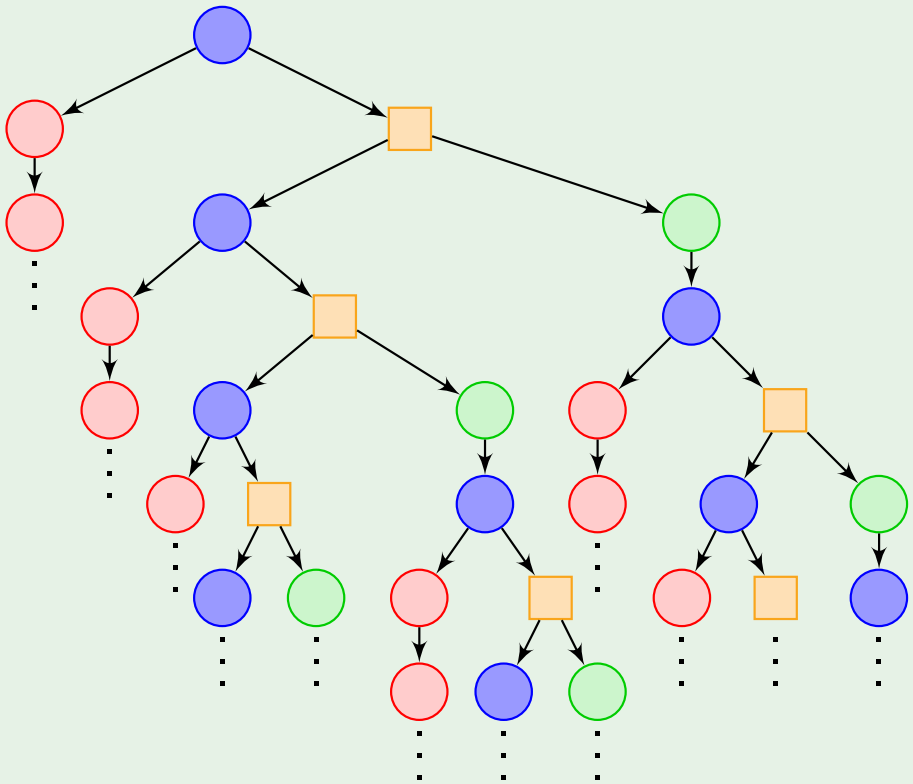
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.

ATL with strategy contexts [BDLM09,DLM10]

Example



$\langle\langle \bigcirc \rangle\rangle\ \mathbf{G}(\ \langle\langle \square \rangle\rangle\ \mathbf{F}\ \bigcirc\ )$

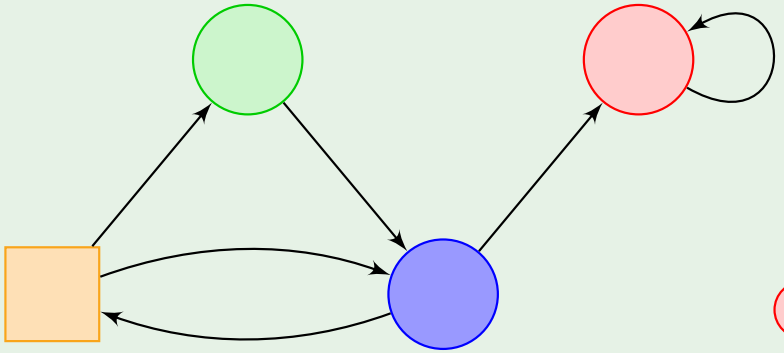


- Player ○ in ● always plays to □.

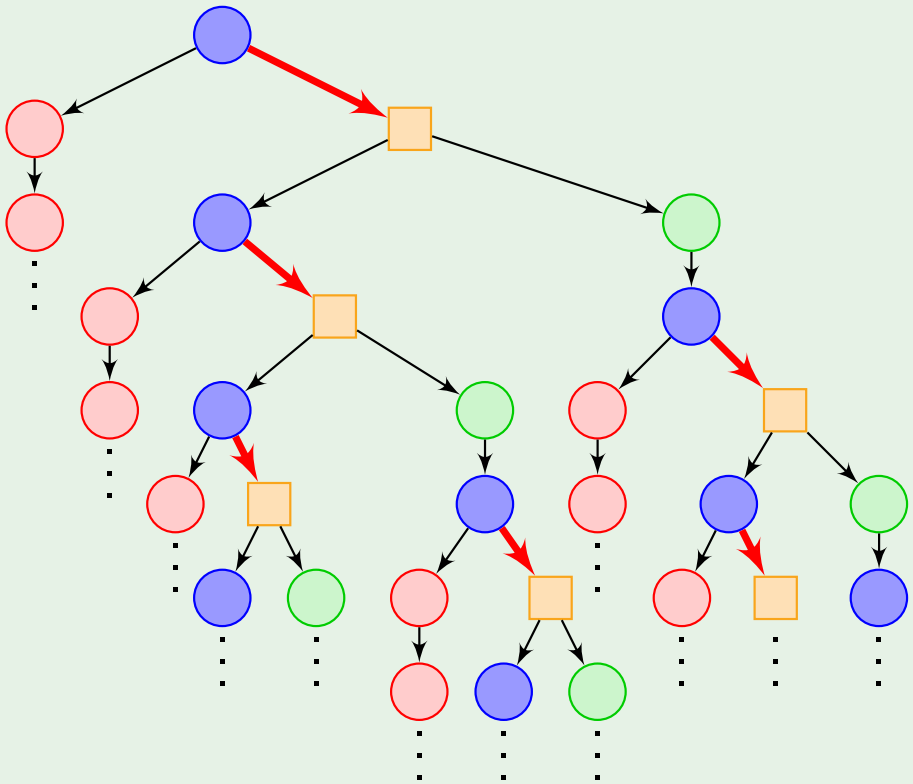
Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.

ATL with strategy contexts [BDLM09,DLM10]

Example



$\langle\langle \bigcirc \rangle\rangle\, \mathbf{G}(\, \langle\langle \square \rangle\rangle\, \mathbf{F}\, \bigcirc\,)$

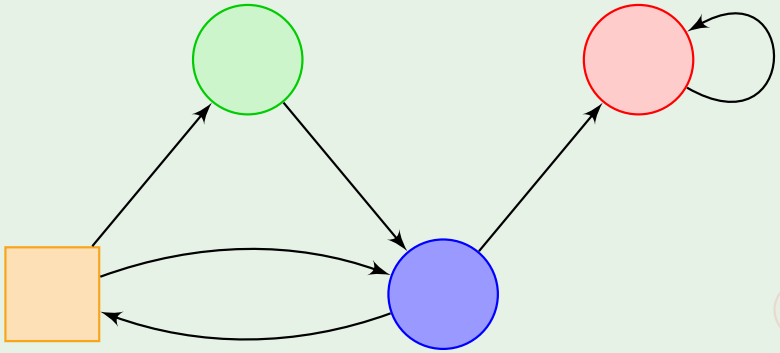


- Player ○ in ● always plays to □.

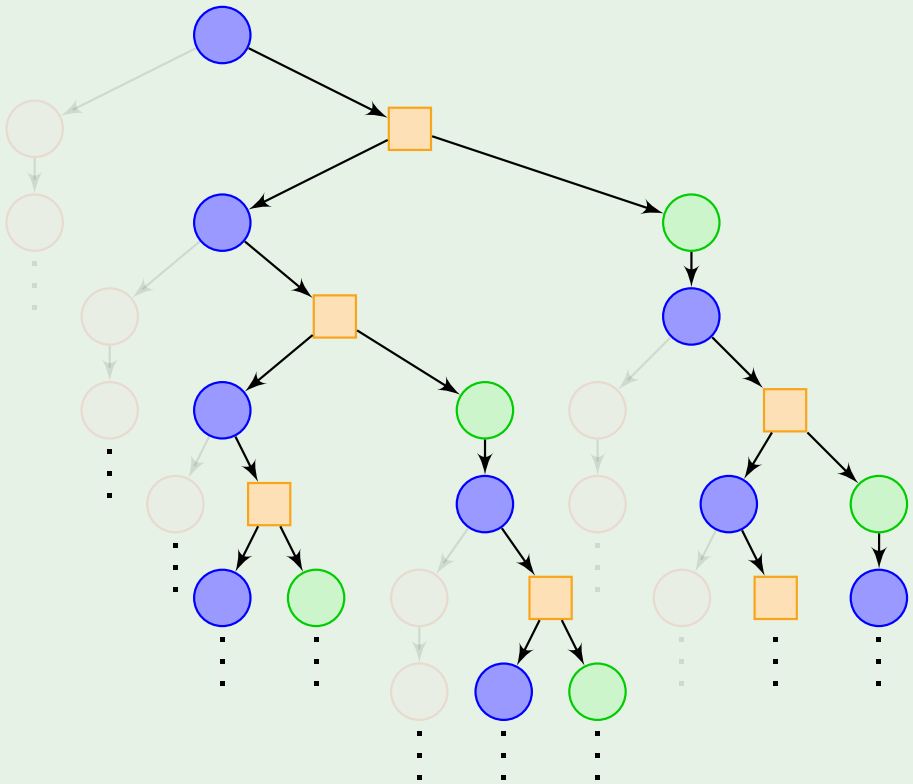
Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.

ATL with strategy contexts [BDLM09,DLM10]

Example



$\langle\langle \bigcirc \rangle\rangle\ \mathbf{G}(\ \langle\langle \square \rangle\rangle\ \mathbf{F}\ \bigcirc\ )$

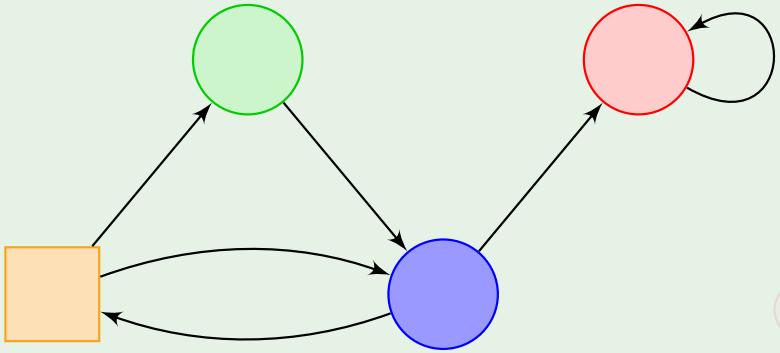


- Player ○ in ● always plays to □.

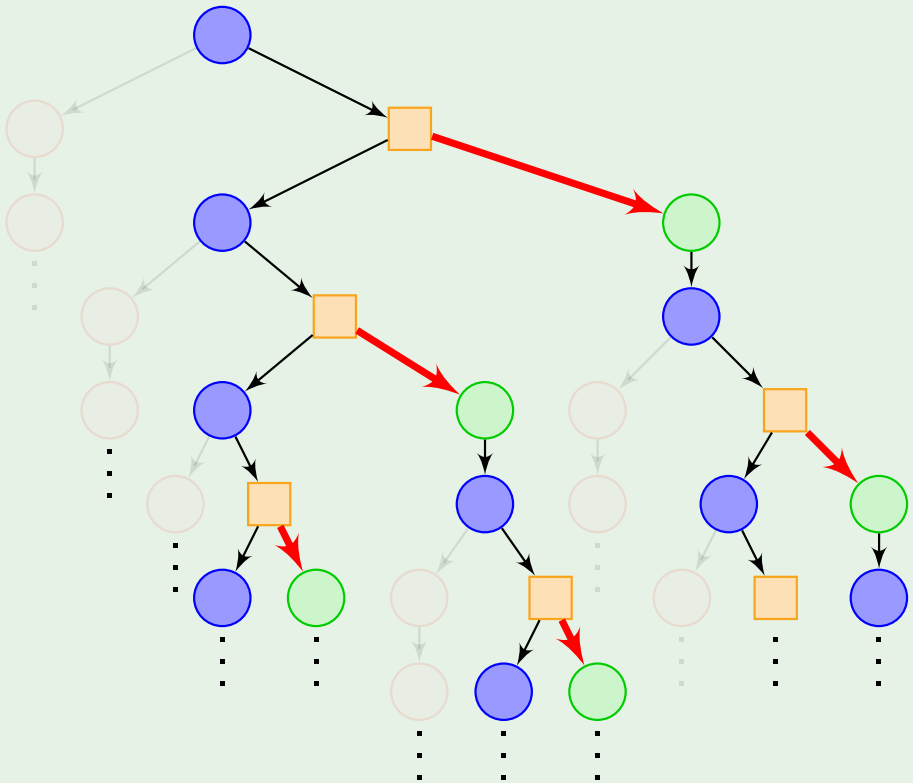
Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.

ATL with strategy contexts [BDLM09,DLM10]

Example



$\langle\langle \bigcirc \rangle\rangle\ \mathbf{G}(\ \langle\langle \square \rangle\rangle\ \mathbf{F}\ \bigcirc\ )$



- Player ○ in ● always plays to □;
- Player □ in □ then plays to ○.

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

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

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 φ;

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

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 φ;
- $\langle\cdot\overline{A}\cdot\rangle \equiv \langle\cdot \mathrm{Agt} \setminus A\cdot\rangle$, 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;

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 φ;
- $\langle\cdot\overline{A}\cdot\rangle \equiv \langle\cdot \mathrm{Agt} \setminus A\cdot\rangle$, 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;
- $(\!|A|\!)$ drops the assigned strategies for A;

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 φ;
- $\langle\cdot\overline{A}\cdot\rangle \equiv \langle\cdot \mathrm{Agt} \setminus A\cdot\rangle$, 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;
- $(\!|A|\!)$ drops the assigned strategies for A;
- $[\cdot A\cdot]$ is dual to $\langle\cdot A\cdot\rangle$:

$$[\cdot A\cdot]\,\varphi \equiv \neg\,\langle\cdot A\cdot\rangle\,\neg\,\varphi$$

Semantics of $\langle\cdot A\cdot\rangle\,\varphi$

Definition

Semantics of ATL strategy quantifier:

$$\mathcal{G}, \bigcirc \models \langle\langle A \rangle\rangle\, \varphi \quad \Leftrightarrow \quad \exists \sigma_A.\ \forall \pi \in \mathsf{Out}(\bigcirc, \sigma_A).\ \pi \models \varphi$$

Semantics of $\langle\cdot A\cdot\rangle\,\varphi$

Definition

Semantics of ATL strategy quantifier:

$$\mathcal{G}, \bigcirc \models \langle\langle A\rangle\rangle\,\varphi \quad \Leftrightarrow \quad \exists\sigma_A.\ \forall\pi \in \mathsf{Out}(\bigcirc, \sigma_A).\ \pi \models \varphi$$

Semantics of ATL_{sc} strategy quantifier:

$$\mathcal{G}, \bigcirc \models_{\sigma_B} \langle\cdot A\cdot\rangle\,\varphi \quad \Leftrightarrow \quad \exists\sigma_A.\ \forall\pi \in \mathsf{Out}(\bigcirc, \sigma_A \circ \sigma_B).\ \pi \models_{\sigma_A\circ\sigma_B} \varphi$$

Semantics of $\langle\cdot A\cdot\rangle\,\varphi$

Definition

Semantics of ATL strategy quantifier:

$$\mathcal{G}, \bigcirc \models \langle\langle A\rangle\rangle\,\varphi \quad \Leftrightarrow \quad \exists\sigma_A.\ \forall\pi \in \mathrm{Out}(\bigcirc, \sigma_A).\ \pi \models \varphi$$

Semantics of ATL_{sc} strategy quantifier:

$$\mathcal{G}, \bigcirc \models_{\sigma_B} \langle\cdot A\cdot\rangle\,\varphi \quad \Leftrightarrow \quad \exists\sigma_A.\ \forall\pi \in \mathrm{Out}(\bigcirc, \sigma_A \circ \sigma_B).\ \pi \models_{\sigma_A\circ\sigma_B} \varphi$$

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

$$\sigma_A \circ \sigma_B : \quad \begin{array}{ll} a \mapsto \sigma_A(a) & \text{if } a \in A \setminus B \\ b \mapsto \sigma_B(b) & \text{if } b \in B \setminus A \\ c \mapsto \sigma_A(c) & \text{if } c \in B \cap A \end{array}$$

What ATL_{sc} can express

What ATL_{sc} can express

- A has a strategy to eventually reach :

What ATL_{sc} can express

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

What ATL_{sc} can express

- A has a strategy to eventually reach ◯:
  - $\langle\cdot A\cdot\rangle\ \mathbf{F}$ ◯
  - $(\!|\overline{\varnothing}|\!)\ \langle\cdot A\cdot\rangle\ \mathbf{F}$ ◯

What ATL_{sc} can express

- A has a strategy to eventually reach ◯:
  - $\langle\cdot A\cdot\rangle\ \mathbf{F}$ ◯
  - $(\!|\overline{\varnothing}|\!)\ \langle\cdot A\cdot\rangle\ \mathbf{F}$ ◯ $\equiv\ \langle\!\langle A\rangle\!\rangle\ \mathbf{F}$ ◯

What ATL_{sc} can express

- A has a strategy to eventually reach ◯:
  - $\langle\cdot A\cdot\rangle\ \mathbf{F}$ ◯
  - $(\!|\overline{\varnothing}|\!)\ \langle\cdot A\cdot\rangle\ \mathbf{F}$ ◯ $\equiv\ \langle\!\langle A\rangle\!\rangle\ \mathbf{F}$ ◯
  - $\langle\cdot A\cdot\rangle\ [\cdot\overline{A}\cdot]\ \mathbf{F}$ ◯

What ATL_{sc} can express

- A has a strategy to eventually reach ◯:
  - $\langle\cdot A\cdot\rangle\ \mathbf{F}$ ◯
  - $(\!|\overline{\varnothing}|\!)\ \langle\cdot A\cdot\rangle\ \mathbf{F}$ ◯ $\equiv\ \langle\!\langle A\rangle\!\rangle\ \mathbf{F}$ ◯
  - $\langle\cdot A\cdot\rangle\ [\cdot\overline{A}\cdot]\ \mathbf{F}$ ◯

- CTL^* and ATL^* properties:

What ATL_{sc} can express

- A has a strategy to eventually reach ●:
  - $\langle\cdot A\cdot\rangle\ \mathbf{F}$ ●
  - $(\!|\overline{\varnothing}|\!)\ \langle\cdot A\cdot\rangle\ \mathbf{F}$ ● $\equiv\ \langle\!\langle A\rangle\!\rangle\ \mathbf{F}$ ●
  - $\langle\cdot A\cdot\rangle\ [\cdot\overline{A}\cdot]\ \mathbf{F}$ ●

- CTL* and ATL* properties:
  - $\langle\cdot\overline{\varnothing}\cdot\rangle\ \mathbf{G}\ \langle\cdot\varnothing\cdot\rangle\ \mathbf{F}$ ●

What ATL_{sc} can express

- A has a strategy to eventually reach ●:
  - $\langle\cdot A\cdot\rangle\ \mathbf{F}$ ●
  - $(\!|\overline{\varnothing}|\!)\ \langle\cdot A\cdot\rangle\ \mathbf{F}$ ● $\equiv\ \langle\!\langle A\rangle\!\rangle\ \mathbf{F}$ ●
  - $\langle\cdot A\cdot\rangle\ [\cdot\overline{A}\cdot]\ \mathbf{F}$ ●

- CTL* and ATL* properties:
  - $\langle\cdot\overline{\varnothing}\cdot\rangle\ \mathbf{G}\ \langle\cdot\varnothing\cdot\rangle\ \mathbf{F}$ ● $\equiv\ \mathbf{E}\,\mathbf{G}\,\mathbf{F}$ ●

What ATL_{sc} can express

- A has a strategy to eventually reach ●:
  - $\langle\cdot A\cdot\rangle\ \mathbf{F}$ ●
  - $(\!|\overline{\varnothing}|\!)\ \langle\cdot A\cdot\rangle\ \mathbf{F}$ ● $\equiv\ \langle\!\langle A\rangle\!\rangle\ \mathbf{F}$ ●
  - $\langle\cdot A\cdot\rangle\ [\cdot\overline{A}\cdot]\ \mathbf{F}$ ●

- CTL* and ATL* properties:
  - $\langle\cdot\overline{\varnothing}\cdot\rangle\ \mathbf{G}\ \langle\cdot\varnothing\cdot\rangle\ \mathbf{F}$ ● $\equiv\ \mathbf{E}\,\mathbf{G}\,\mathbf{F}$ ●
  - $\langle\cdot\overline{\varnothing}\cdot\rangle\,(\mathbf{G}\ \langle\cdot\varnothing\cdot\rangle\ \mathbf{F}$ ● $\wedge\ \mathbf{G}\ \langle\cdot\varnothing\cdot\rangle\ \mathbf{F}$ ●$)$

What ATL_{sc} can express

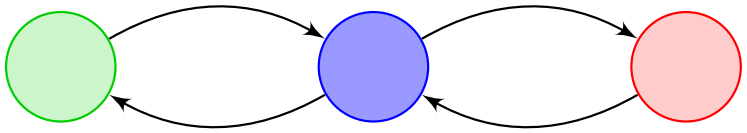
- A has a strategy to eventually reach ●:
  - $\langle\cdot A\cdot\rangle\ \mathbf{F}$ ●
  - $(\!|\overline{\varnothing}|\!)\ \langle\cdot A\cdot\rangle\ \mathbf{F}$ ● $\equiv\ \langle\!\langle A\rangle\!\rangle\ \mathbf{F}$ ●
  - $\langle\cdot A\cdot\rangle\ [\cdot\overline{A}\cdot]\ \mathbf{F}$ ●

- CTL* and ATL* properties:
  - $\langle\cdot\overline{\varnothing}\cdot\rangle\ \mathbf{G}\ \langle\cdot\varnothing\cdot\rangle\ \mathbf{F}$ ● $\equiv \mathbf{E}\,\mathbf{G}\,\mathbf{F}$ ●
  - $\langle\cdot\overline{\varnothing}\cdot\rangle\,(\mathbf{G}\ \langle\cdot\varnothing\cdot\rangle\ \mathbf{F}$ ● $\wedge\ \mathbf{G}\ \langle\cdot\varnothing\cdot\rangle\ \mathbf{F}$ ●$) \equiv \mathbf{E}(\mathbf{G}\,\mathbf{F}$ ● $\wedge\ \mathbf{G}\,\mathbf{F}$ ●$)$

What ATL_{sc} can express

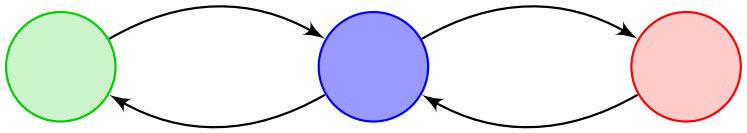
- A has a strategy to eventually reach ●:
  - $\langle\cdot A\cdot\rangle\ \mathbf{F}$ ●
  - $(\!|\overline{\varnothing}|\!)\ \langle\cdot A\cdot\rangle\ \mathbf{F}$ ● $\equiv\ \langle\!\langle A\rangle\!\rangle\ \mathbf{F}$ ●
  - $\langle\cdot A\cdot\rangle\ [\cdot\overline{A}\cdot]\ \mathbf{F}$ ●

- CTL* and ATL* properties:
  - $\langle\cdot\overline{\varnothing}\cdot\rangle\ \mathbf{G}\ \langle\cdot\varnothing\cdot\rangle\ \mathbf{F}$ ● $\equiv \mathbf{E}\,\mathbf{G}\,\mathbf{F}$ ●
  - $\langle\cdot\overline{\varnothing}\cdot\rangle\,(\mathbf{G}\ \langle\cdot\varnothing\cdot\rangle\ \mathbf{F}$ ● $\wedge\ \mathbf{G}\ \langle\cdot\varnothing\cdot\rangle\ \mathbf{F}$ ●$) \equiv \mathbf{E}(\mathbf{G}\,\mathbf{F}$ ● $\wedge\ \mathbf{G}\,\mathbf{F}$ ●$)$



What ATL_{sc} can express

- A has a strategy to eventually reach ●:
  - $\langle\cdot A\cdot\rangle\ \mathbf{F}$ ●
  - $(\!|\overline{\varnothing}|\!)\ \langle\cdot A\cdot\rangle\ \mathbf{F}$ ● $\equiv\ \langle\!\langle A\rangle\!\rangle\ \mathbf{F}$ ●
  - $\langle\cdot A\cdot\rangle\ [\cdot\overline{A}\cdot]\ \mathbf{F}$ ●
- CTL* and ATL* properties:
  - $\langle\cdot\overline{\varnothing}\cdot\rangle\ \mathbf{G}\ \langle\cdot\varnothing\cdot\rangle\ \mathbf{F}$ ● $\equiv\ \mathbf{E}\,\mathbf{G}\,\mathbf{F}$ ●
  - $\langle\cdot\overline{\varnothing}\cdot\rangle\,(\mathbf{G}\ \langle\cdot\varnothing\cdot\rangle\ \mathbf{F}$ ● $\wedge\ \mathbf{G}\ \langle\cdot\varnothing\cdot\rangle\ \mathbf{F}$ ●$)\ \equiv\ \mathbf{E}(\mathbf{G}\,\mathbf{F}$ ● $\wedge\ \mathbf{G}\,\mathbf{F}$ ●$)$



$\rightsquigarrow$ memory needed!

What ATL_{sc} can express

What ATL_{sc} can express

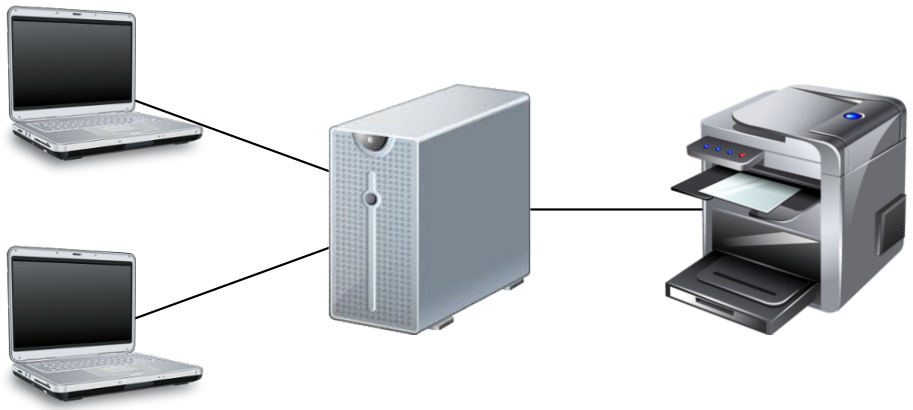
- existence of a correct controller for the elevator:

$$\langle\cdot\mathrm{ctrl}\cdot\rangle\ \mathbf{G}\left[\begin{array}{l} \bigwedge_{i,j} \langle\cdot\mathrm{user}\cdot\rangle_0\ \mathbf{F}\left(\mathrm{op}_i \wedge \mathbf{F}\ \mathrm{op}_j\right) \\ \wedge \\ \bigwedge_{i\in\{0,1,2\}} \mathrm{op}_i \Rightarrow \mathrm{serv}_i) \end{array}\right]$$

What ATL_{sc} can express

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

$$\langle\cdot \mathrm{Server} \cdot\rangle\ \mathbf{G} \left[\begin{array}{l} \bigwedge_{c \in \mathrm{Clients}} \langle\cdot c \cdot\rangle\ \mathbf{F}\ \mathrm{access}_c \\ \wedge \\ \neg \bigwedge_{c \neq c'} \mathrm{access}_c \wedge \mathrm{access}_{c'} \end{array} \right]$$



What ATL_{sc} can express

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

$$\langle\cdot \mathrm{Server} \cdot\rangle\ \mathbf{G} \left[\begin{array}{l} \bigwedge_{c \in \mathrm{Clients}} \langle\cdot c \cdot\rangle\ \mathbf{F}\ \mathrm{access}_c \\ \wedge \\ \neg \bigwedge_{c \neq c'} \mathrm{access}_c \wedge \mathrm{access}_{c'} \end{array} \right]$$

- Existence of Nash equilibria:

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

What ATL_{sc} can express

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

$$\langle\cdot \mathrm{Server} \cdot\rangle\ \mathbf{G} \left[\begin{array}{l} \bigwedge_{c \in \mathrm{Clients}} \langle\cdot c \cdot\rangle\ \mathbf{F}\ \mathrm{access}_c \\ \wedge \\ \neg \bigwedge_{c \neq c'} \mathrm{access}_c \wedge \mathrm{access}_{c'} \end{array} \right]$$

- 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:

$$\langle\cdot A \cdot\rangle\ [\cdot B \cdot]\, (\neg \varphi \Rightarrow [\cdot A \cdot]\, \neg \varphi)$$

More expressiveness results

Theorem

ATL_{sc} is strictly more expressive than ATL.

More expressiveness results

Theorem

ATL_{sc} is strictly more expressive than ATL.

Proof

- $\langle\langle A \rangle\rangle\, \varphi \equiv (\!|\mathrm{Agt}|\!)\ \langle\cdot A \cdot\rangle\, \hat{\varphi}$

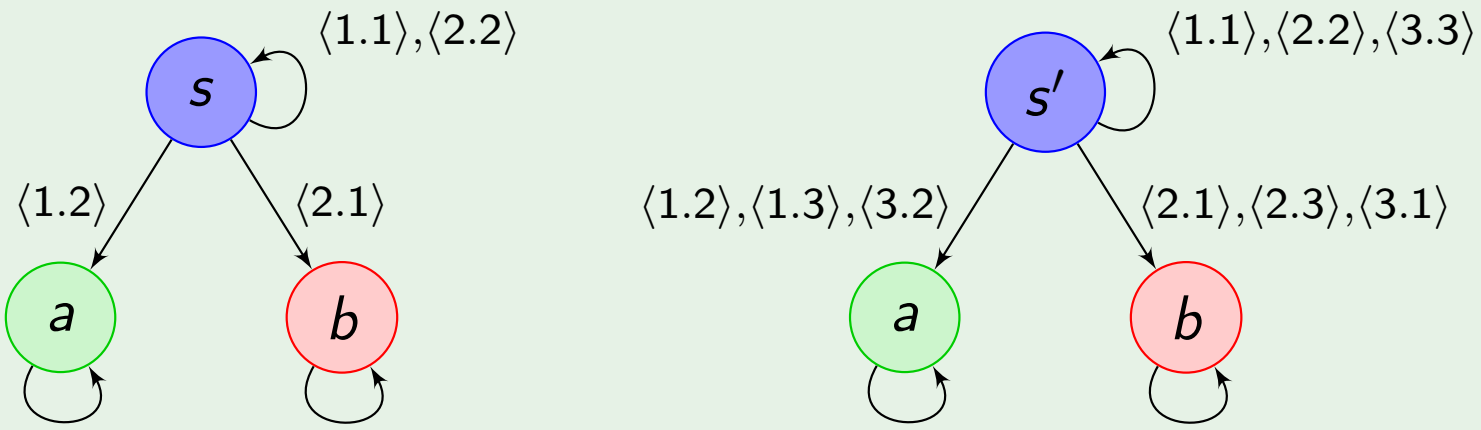
More expressiveness results

Theorem

ATL_{sc} is strictly more expressive than ATL.

Proof

- $\langle\langle A \rangle\rangle\, \varphi \equiv (\!|\mathrm{Agt}|\!)\ \langle\cdot A \cdot\rangle\, \hat{\varphi}$
- $\langle\cdot 1 \cdot\rangle\, (\, \langle\cdot 2 \cdot\rangle\ \mathbf{X}\, a \wedge\ \langle\cdot 2 \cdot\rangle\ \mathbf{X}\, b)$ is true in s' and not in s.



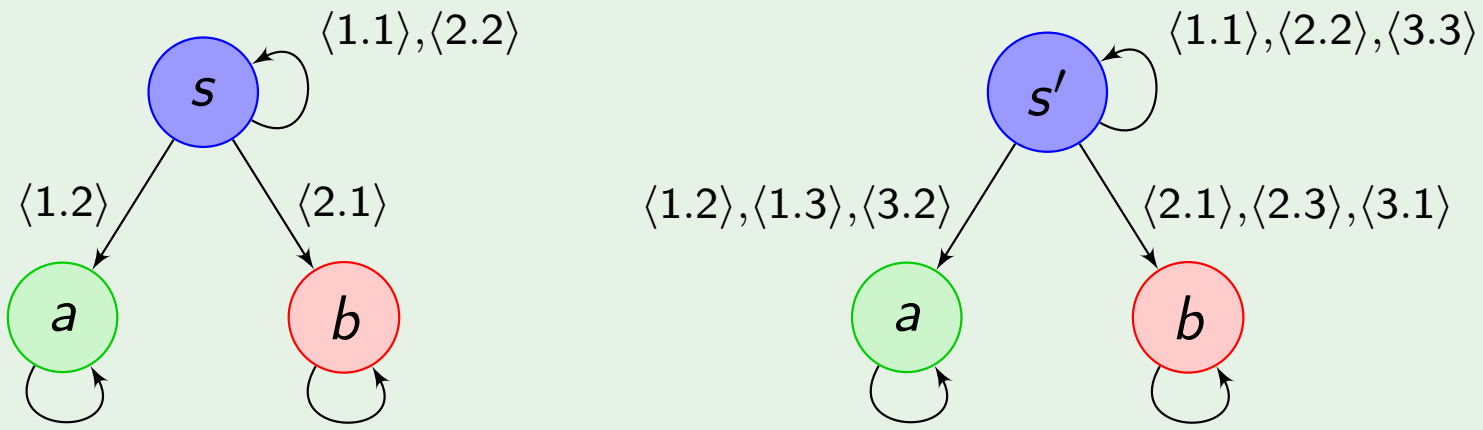
More expressiveness results

Theorem

ATL_{sc} is strictly more expressive than ATL.

Proof

- $\langle\langle A \rangle\rangle\, \varphi \equiv (\!|\mathrm{Agt}|\!)\ \langle\cdot A \cdot\rangle\, \hat{\varphi}$
- $\langle\cdot 1 \cdot\rangle\, (\, \langle\cdot 2 \cdot\rangle\ \mathbf{X}\, a \wedge \ \langle\cdot 2 \cdot\rangle\ \mathbf{X}\, b)$ is true in s' and not in s.
  But s and s' are alternating bisimilar.



More expressiveness results

Theorem

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

More expressiveness results

Theorem

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

Proof

- Make implicit quantification explicit:

$$\langle\cdot A\cdot\rangle\,\varphi \equiv \langle\cdot A\cdot\rangle\,[\cdot \mathrm{Agt} \setminus A\cdot]\,\widehat{\varphi}$$

More expressiveness results

Theorem

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

Proof

- Make implicit quantification explicit:

$$\langle\cdot A\cdot\rangle\,\varphi \equiv \langle\cdot A\cdot\rangle\,[\cdot \mathrm{Agt} \setminus A\cdot]\,\widehat{\varphi}$$

  - 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};*
- *$(\!|A|\!)$ does not add expressive power.*

Proof

- Make implicit quantification explicit:

$$\langle\cdot A\cdot\rangle\,\varphi \equiv \langle\cdot A\cdot\rangle\,[\cdot \mathrm{Agt} \setminus A\cdot]\,\widehat{\varphi}$$

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

- $(\!|A|\!)\,\varphi$ is then equivalent to $[\cdot A\cdot]\,\widetilde{\varphi}$;

More expressiveness results

Theorem

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

Proof

- Make implicit quantification explicit:

$$\langle\cdot A\cdot\rangle\,\varphi \equiv \langle\cdot A\cdot\rangle\,[\cdot \mathrm{Agt} \setminus A\cdot]\,\widehat{\varphi}$$

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

- $(\!|A|\!)\,\varphi$ is then equivalent to $[\cdot A\cdot]\,\widetilde{\varphi}$;
- for full context, insert $\langle\cdot\varnothing\cdot\rangle$ between temporal modalities:

$$\mathbf{G}\,\mathbf{F}\,\varphi \equiv \langle\cdot\varnothing\cdot\rangle\,\mathbf{G}\,\langle\cdot\varnothing\cdot\rangle\,\mathbf{F}\,\varphi$$

Related works – Strategy logic [CHP07,MMV10]

[CHP07] Chatterjee, Henzinger, Piterman. Strategy Logic. CONCUR, 2007.
[MMV10] Mogavero, Murano, Vardi. Reasoning about strategies. FSTTCS, 2010.

Related works – Strategy logic [CHP07,MMV10]

Explicit manipulation of strategies

- first-order quantification over strategies;

$$\exists\sigma.\ \varphi(\sigma) \qquad \textit{there is a strategy } \sigma \textit{ such that } \varphi \textit{ holds}$$

- assignment of strategies to agents.

$$\mathsf{assign}(A \mapsto \sigma)\varphi \qquad \textit{if } A \textit{ plays } \sigma \textit{ then } \varphi \textit{ holds}$$

[CHP07] Chatterjee, Henzinger, Piterman. Strategy Logic. CONCUR, 2007.
[MMV10] Mogavero, Murano, Vardi. Reasoning about strategies. FSTTCS, 2010.

Related works – Strategy logic [CHP07,MMV10]

Explicit manipulation of strategies

- first-order quantification over strategies;

$$\exists\sigma.\ \varphi(\sigma) \qquad \textit{there is a strategy } \sigma \textit{ such that } \varphi \textit{ holds}$$

- assignment of strategies to agents.

$$\mathsf{assign}(A \mapsto \sigma)\varphi \qquad \textit{if } A \textit{ plays } \sigma \textit{ then } \varphi \textit{ holds}$$

Example

$$\underbrace{\exists\sigma_1.\ \mathsf{assign}(A \mapsto \sigma_1)}_{\langle\cdot A\cdot\rangle}.\ \underbrace{\forall\sigma_2.\ \mathsf{assign}(B \mapsto \sigma_2)}_{[\cdot B\cdot]}.\ \mathbf{F}$$

[CHP07] Chatterjee, Henzinger, Piterman. Strategy Logic. CONCUR, 2007.

[MMV10] Mogavero, Murano, Vardi. Reasoning about strategies. FSTTCS, 2010.

Related works – Strategy logic [CHP07,MMV10]

Explicit manipulation of strategies

- first-order quantification over strategies;

$$\exists \sigma.\ \varphi(\sigma) \qquad \textit{there is a strategy } \sigma \textit{ such that } \varphi \textit{ holds}$$

- assignment of strategies to agents.

$$\mathsf{assign}(A \mapsto \sigma)\varphi \qquad \textit{if } A \textit{ plays } \sigma \textit{ then } \varphi \textit{ holds}$$

Example

$$\underbrace{\exists \sigma_1.\ \mathsf{assign}(A \mapsto \sigma_1)}_{\langle \cdot A \cdot \rangle}.\ \underbrace{\forall \sigma_2.\ \mathsf{assign}(B \mapsto \sigma_2)}_{[\cdot B \cdot]}.\ \mathbf{F}\,\bigcirc \equiv \langle \cdot A \cdot \rangle\ \mathbf{F}\,\bigcirc$$

[CHP07] Chatterjee, Henzinger, Piterman. Strategy Logic. CONCUR, 2007.

[MMV10] Mogavero, Murano, Vardi. Reasoning about strategies. FSTTCS, 2010.

Related works – Strategy logic [CHP07,MMV10]

Explicit manipulation of strategies

- first-order quantification over strategies;

$$\exists\sigma.\ \varphi(\sigma) \qquad \textit{there is a strategy } \sigma \textit{ such that } \varphi \textit{ holds}$$

- assignment of strategies to agents.

$$\mathsf{assign}(A \mapsto \sigma)\varphi \qquad \textit{if } A \textit{ plays } \sigma \textit{ then } \varphi \textit{ holds}$$

Example

$$\underbrace{\exists\sigma_1.\ \mathsf{assign}(A \mapsto \sigma_1)}_{\langle\cdot A\cdot\rangle}.\ \underbrace{\forall\sigma_2.\ \mathsf{assign}(B \mapsto \sigma_2)}_{[\cdot B\cdot]}.\ \mathbf{F}\,\bigcirc \equiv \langle\cdot A\cdot\rangle\ \mathbf{F}\,\bigcirc$$

$$\exists\sigma_1.\ \mathbf{A}\,\mathbf{G}(\mathsf{assign}(A \mapsto \sigma_1).\ \forall\sigma_2.\ \mathsf{assign}(B \mapsto \sigma_2)\,\mathbf{F}\,\bigcirc)$$

[CHP07] Chatterjee, Henzinger, Piterman. Strategy Logic. CONCUR, 2007.
[MMV10] Mogavero, Murano, Vardi. Reasoning about strategies. FSTTCS, 2010.

Related works – Strategy logic [CHP07,MMV10]

Explicit manipulation of strategies

- first-order quantification over strategies;

$$\exists\sigma.\ \varphi(\sigma) \qquad \textit{there is a strategy } \sigma \textit{ such that } \varphi \textit{ holds}$$

- assignment of strategies to agents.

$$\mathsf{assign}(A \mapsto \sigma)\varphi \qquad \textit{if } A \textit{ plays } \sigma \textit{ then } \varphi \textit{ holds}$$

Proposition

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

[CHP07] Chatterjee, Henzinger, Piterman. Strategy Logic. CONCUR, 2007.
[MMV10] Mogavero, Murano, Vardi. Reasoning about strategies. FSTTCS, 2010.

Related works – Strategy logic [CHP07,MMV10]

Explicit manipulation of strategies

- first-order quantification over strategies;

$$\exists\sigma.\ \varphi(\sigma) \qquad \textit{there is a strategy } \sigma \textit{ such that } \varphi \textit{ holds}$$

- assignment of strategies to agents.

$$\mathsf{assign}(A \mapsto \sigma)\varphi \qquad \textit{if } A \textit{ plays } \sigma \textit{ then } \varphi \textit{ holds}$$

Proposition

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

Theorem

SL *model checking is decidable;* SL *satisfiability is undecidable.*

[CHP07] Chatterjee, Henzinger, Piterman. Strategy Logic. CONCUR, 2007.
[MMV10] Mogavero, Murano, Vardi. Reasoning about strategies. FSTTCS, 2010.

Some other related approaches

Some other related approaches

(Basic) strategy interaction logic [WSH15]

- ATL augmented with strategy interaction quantifiers:

$$\langle\langle A \rangle\rangle \, (\, \langle\langle +B \rangle\rangle \, \varphi_B \, \wedge \, \langle\langle +C \rangle\rangle \, \varphi_c)$$

[WSH15] Wang, Schewe, Huang. An Extension of ATL With Strategy Interaction. ACM ToPLAS, 2015.

Some other related approaches

(Basic) strategy interaction logic [WSH15]

- ATL augmented with strategy interaction quantifiers:

$$\langle\langle A \rangle\rangle \, (\, \langle\langle +B \rangle\rangle \, \varphi_B \, \wedge \, \langle\langle +C \rangle\rangle \, \varphi_c)$$

- context is reset after temporal modalities

[WSH15] Wang, Schewe, Huang. An Extension of ATL With Strategy Interaction. ACM ToPLAS, 2015.

Some other related approaches

(Basic) strategy interaction logic [WSH15]

- ATL augmented with strategy interaction quantifiers:

$$\langle\langle A \rangle\rangle \, (\, \langle\langle +B \rangle\rangle \, \varphi_B \, \wedge \, \langle\langle +C \rangle\rangle \, \varphi_c)$$

- context is reset after temporal modalities
- only existential SIQ

[WSH15] Wang, Schewe, Huang. An Extension of ATL With Strategy Interaction. ACM ToPLAS, 2015.

Some other related approaches

(Basic) strategy interaction logic [WSH15]

- ATL augmented with strategy interaction quantifiers:

$$\langle\langle A \rangle\rangle \, (\, \langle\langle +B \rangle\rangle \, \varphi_B \, \wedge \, \langle\langle +C \rangle\rangle \, \varphi_c)$$

- context is reset after temporal modalities
- only existential SIQ

Proposition ([WSH15])

Model checking BSIL is PSPACE*-complete.*

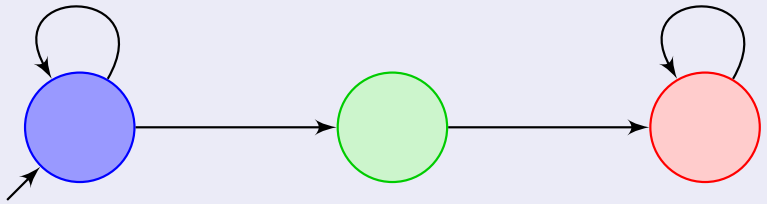
[WSH15] Wang, Schewe, Huang. An Extension of ATL With Strategy Interaction. ACM ToPLAS, 2015.

Some other related approaches

(Basic) strategy interaction logic [WSH15]

ATL with irrevocable strategies [ÅGJ07]

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



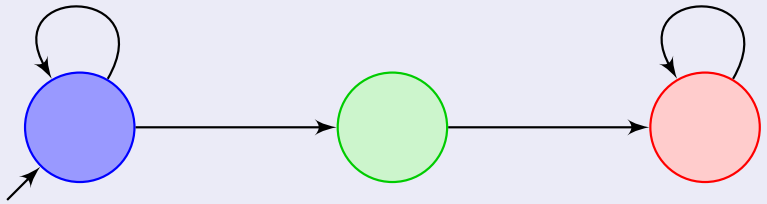
[ÅGJ07] Ågostnes, Goranko, Jamroga. ATL with Irrevocable Strategies. TARK, 2007.

Some other related approaches

(Basic) strategy interaction logic [WSH15]

ATL with irrevocable strategies [ÅGJ07]

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



- better use one single strategy per agent;

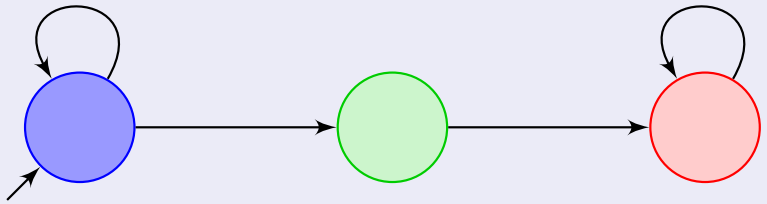
[ÅGJ07] Ågostnes, Goranko, Jamroga. ATL with Irrevocable Strategies. TARK, 2007.

Some other related approaches

(Basic) strategy interaction logic [WSH15]

ATL with irrevocable strategies [ÅGJ07]

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



- better use one single strategy per agent;
- strategies are persistent.

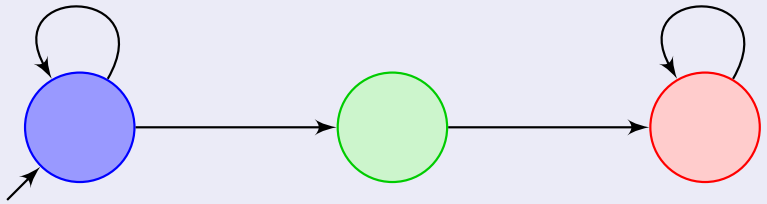
[ÅGJ07] Ågostnes, Goranko, Jamroga. ATL with Irrevocable Strategies. TARK, 2007.

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.*

[ÅGJ07] Ågostnes, Goranko, Jamroga. ATL with Irrevocable Strategies. TARK, 2007.

Some other related approaches

(Basic) strategy interaction logic [WSH15]

ATL with irrevocable strategies [ÅGJ07]

ATL with explicit strategies [WHW07]

- $\langle\langle A \rangle\rangle_\rho \varphi$: ρ explicitly imposes a strategy for some players;

[WHW07] Walther, van der Hoek, Wooldridge. ATL with Explicit Strategies. TARK, 2007.

Some other related approaches

(Basic) strategy interaction logic [WSH15]

ATL with irrevocable strategies [ÅGJ07]

ATL with explicit strategies [WHW07]

- $\langle\langle A \rangle\rangle_\rho \varphi$: ρ explicitly imposes a strategy for some players;

Proposition

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

[WHW07] Walther, van der Hoek, Wooldridge. ATL with Explicit Strategies. TARK, 2007.

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 ATL_{sc} to stochastic setting;

[BBGK07] Baier, Brázdil, Größer, Kučera. Stochastic Game Logic. QEST, 2007.

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 ATL_{sc} to stochastic setting;
- quantitative objectives make model checking undecidable;

[BBGK07] Baier, Brázdil, Größer, Kučera. Stochastic Game Logic. QEST, 2007.

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 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.

[BBGK07] Baier, Brázdil, Größer, Kučera. Stochastic Game Logic. QEST, 2007.