ATL with strategy contexts

Nicolas Markey
LSV – ENS Cachan

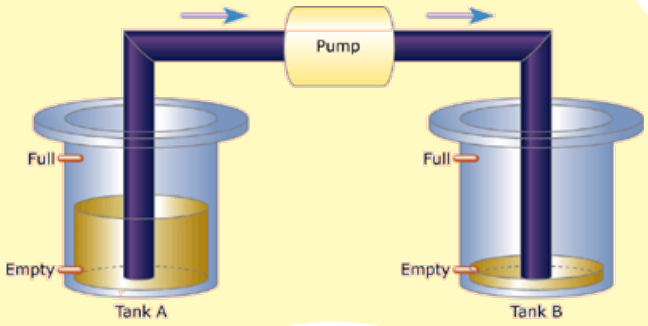
(based on joint works with Thomas Brihaye,
Arnaud Da Costa-Lopes, François Laroussinie)

Laboratoire d'Algorithmique, Complexité et Logique

Créteil, November 9, 2015

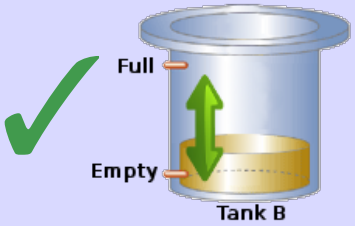
Model checking and synthesis

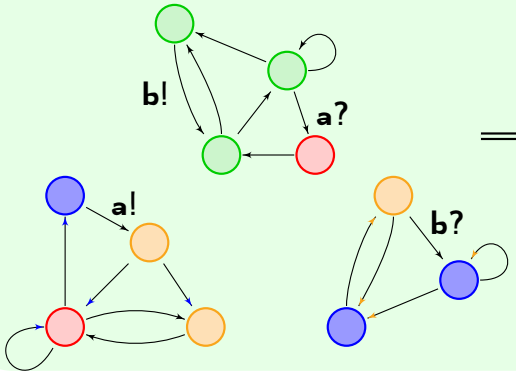
system:



[http://www.embedded.com]

property

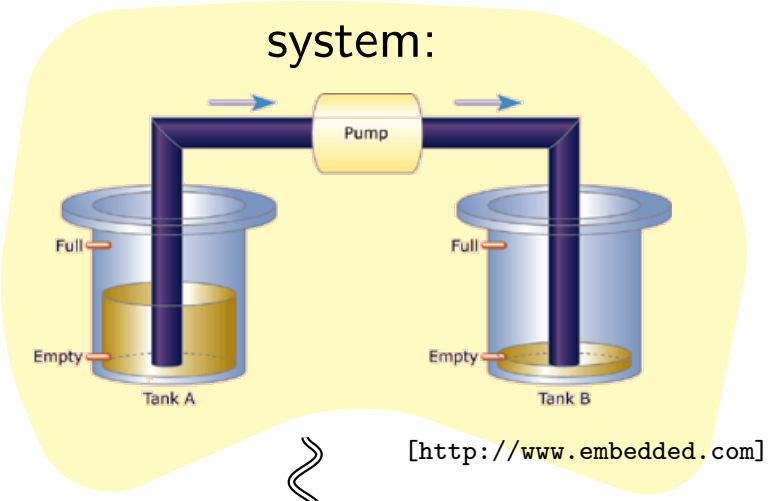


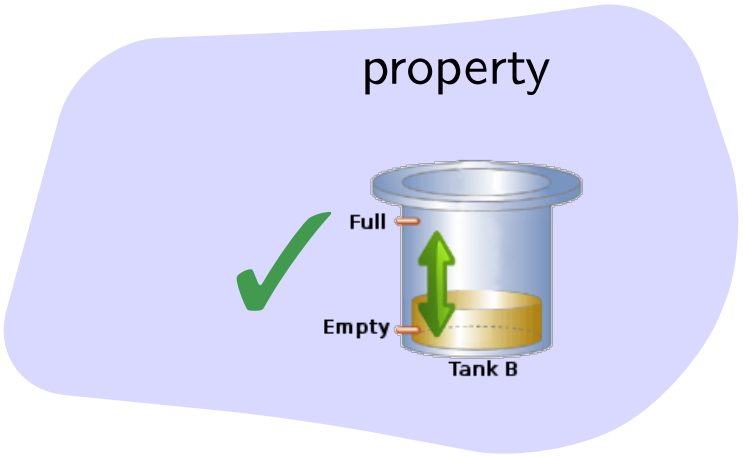


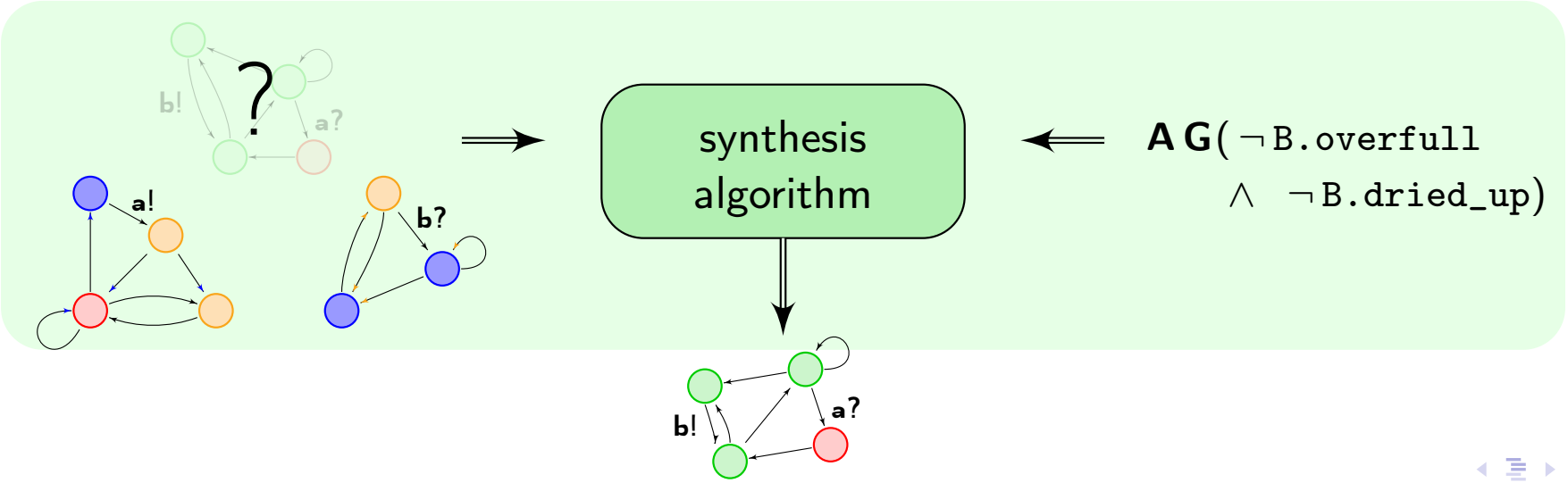
$\Longrightarrow$ model-checking algorithm $\Longleftarrow$ $\mathbf{A}\,\mathbf{G}(\neg\,\texttt{B.overfull} \wedge \neg\,\texttt{B.dried\_up})$

yes/no

Model checking and synthesis







Outline of the presentation

1. Introduction
2. Basics of CTL and ATL
   - expressing properties of reactive systems
   - efficient verification algorithms
3. Temporal logics for multi-agent systems
   - specifying properties of complex interacting systems
   - expressive power of ATL_{sc}
   - translation into Quantified CTL (QCTL)
   - algorithms for ATL_{sc}
4. Conclusions and future works

Outline of the presentation

1. Introduction
2. Basics of CTL and ATL
   - expressing properties of reactive systems
   - efficient verification algorithms
3. Temporal logics for multi-agent systems
   - specifying properties of complex interacting systems
   - expressive power of ATL_{sc}
   - translation into Quantified CTL (QCTL)
   - algorithms for ATL_{sc}
4. Conclusions and future works

Computation-Tree Logic (CTL)

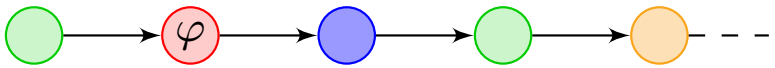
- **atomic propositions:**  ,  , ...

Computation-Tree Logic (CTL)

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

Computation-Tree Logic (CTL)

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

$\mathbf{X}\,\varphi$ 

"next φ"

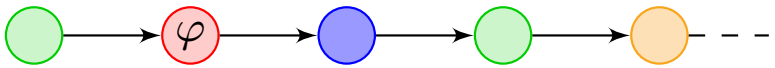
$\varphi\ \mathbf{U}\ \psi$ 

"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"

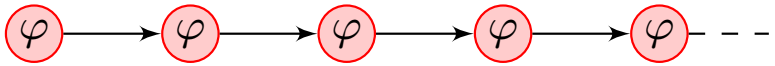
$\varphi\ \mathbf{U}\ \psi$

"φ until ψ"

$\texttt{true}\ \mathbf{U}\ \varphi \equiv \mathbf{F}\,\varphi$

"eventually φ"

$\neg\ \mathbf{F}\ \neg\varphi \equiv \mathbf{G}\,\varphi$

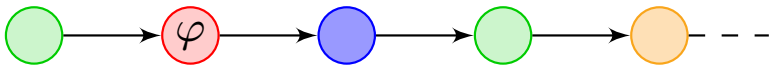


"always φ"

Computation-Tree Logic (CTL)

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

$\mathbf{X}\,\varphi$



"next φ"

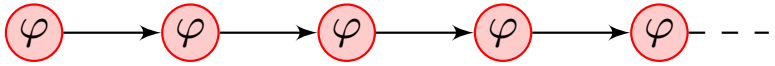
$\varphi\ \mathbf{U}\ \psi$

"φ until ψ"

$\texttt{true}\ \mathbf{U}\ \varphi \equiv \mathbf{F}\,\varphi$

"eventually φ"

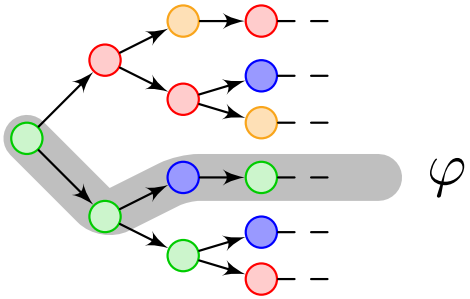
$\neg\,\mathbf{F}\,\neg\varphi \equiv \mathbf{G}\,\varphi$



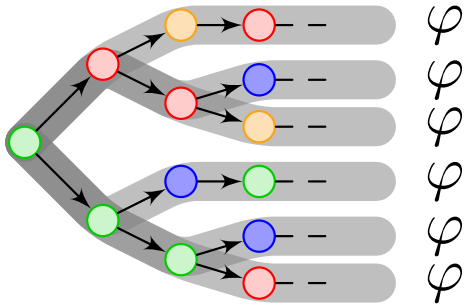
"always φ"

- **path quantifiers:**

$\mathbf{E}\varphi$



$\mathbf{A}\varphi$



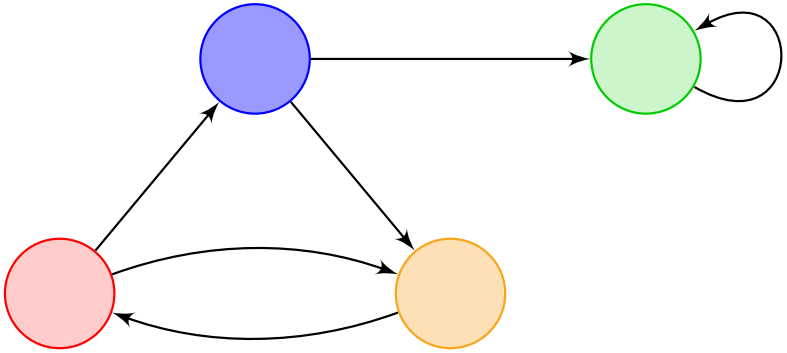
Examples of CTL formulas

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

Examples of CTL formulas

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

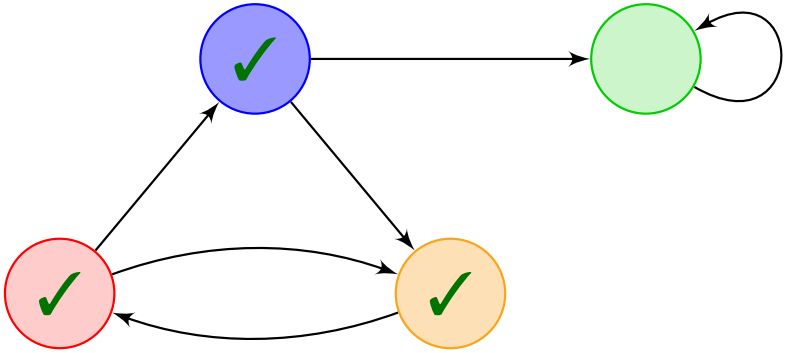
E F is reachable



Examples of CTL formulas

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

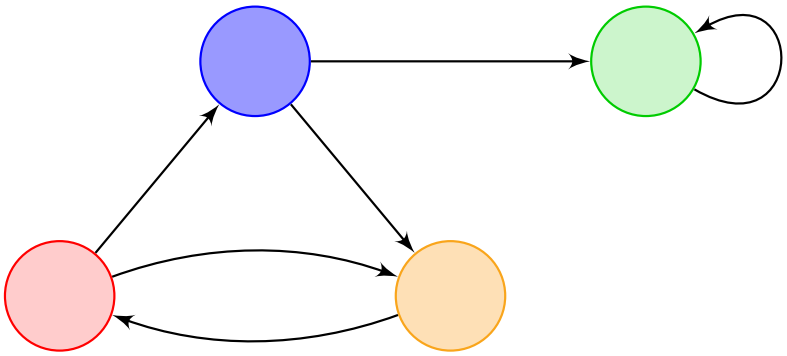
E F ● ● is reachable



Examples of CTL formulas

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

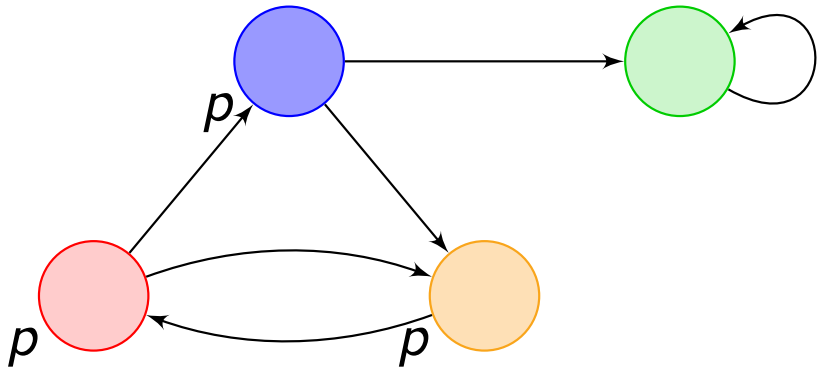
$\mathbf{E}\,\mathbf{G}(\neg$ ● $\wedge\ \mathbf{E}\,\mathbf{F}$ ●$)$ there is a path along which ● is always reachable, but never reached



Examples of CTL formulas

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

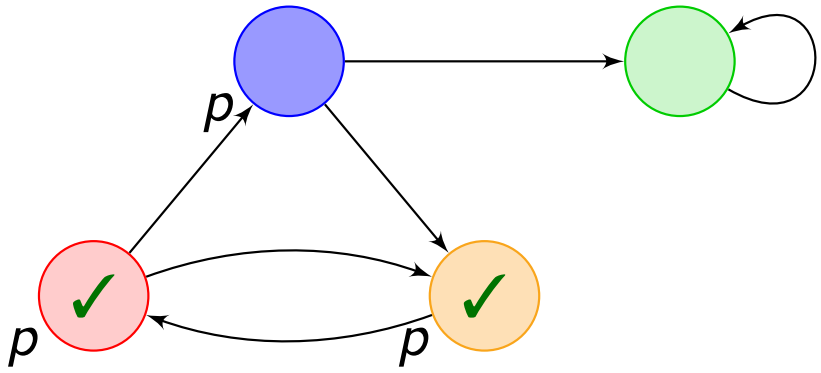
$\mathbf{E}\,\mathbf{G}(\neg \bigcirc \wedge \underbrace{\mathbf{E}\,\mathbf{F} \bigcirc}_{p})$ there is a path along which $\bigcirc$ is always reachable, but never reached



Examples of CTL formulas

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

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



Examples of CTL formulas

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

Theorem ([CE81,QS82])

CTL model checking is PTIME*-complete.*

[CE81] Clarke, Emerson. Design and Synthesis of Synchronization Skeletons... LOP, 1981.
[QS82] Queille, Sifakis. Specification and verification of concurrent systems in CESAR. SOP, 1982.

Examples of CTL formulas

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

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 of Synchronization Skeletons... LOP, 1981.
[QS82] Queille, Sifakis. Specification and verification of concurrent systems in CESAR. SOP, 1982.
[KVW94] Kupferman, Vardi, Wolper. An automata-theoretic approach to branching-time... CAV, 1994.

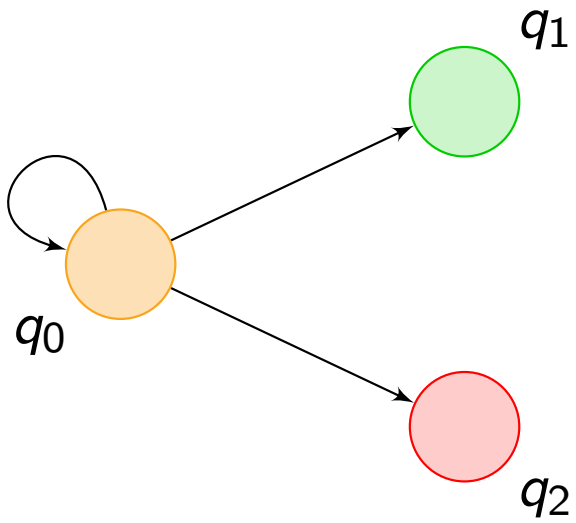
Reasoning about open systems

Reasoning about open systems

Concurrent games

A concurrent game is made of

- a transition system;

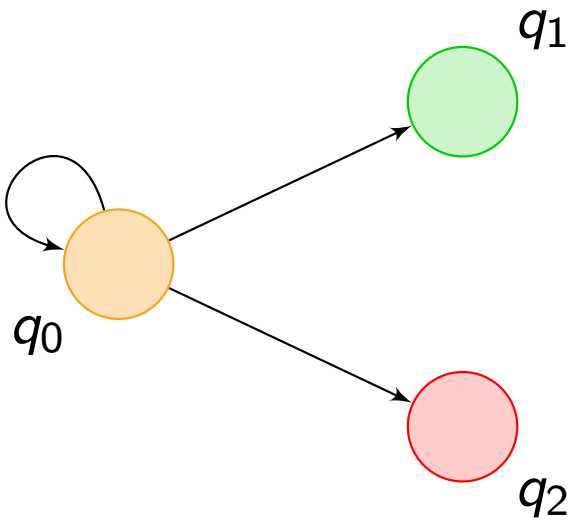


Reasoning about open systems

Concurrent games

A concurrent game is made of

- a transition system;
- a set of agents (or players);

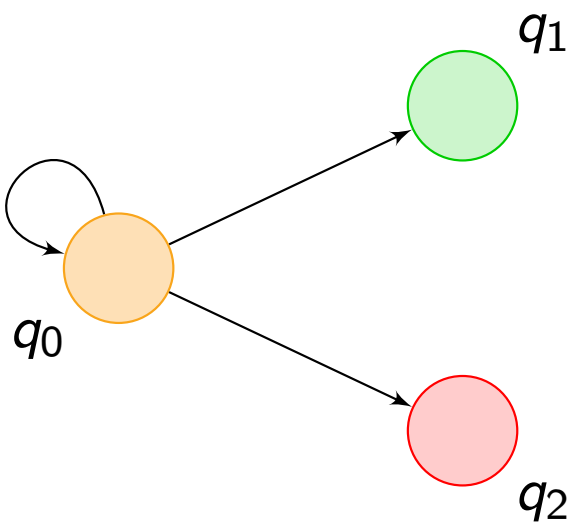


Reasoning about open systems

Concurrent games

A concurrent game is made of

- a transition system;
- 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

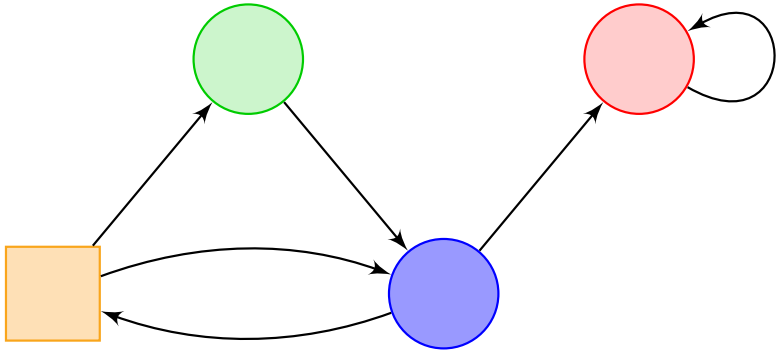
Concurrent games

A concurrent game is made of

- a transition system;
- 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 is a game where only one agent plays at a time.



Reasoning about open systems

Strategies

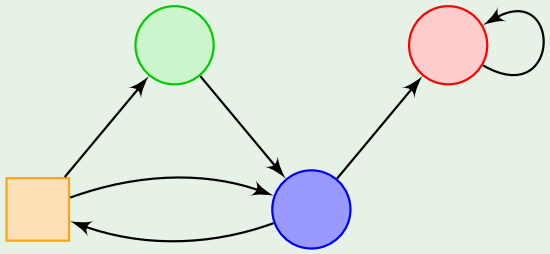
A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Reasoning about open systems

Strategies

A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Example



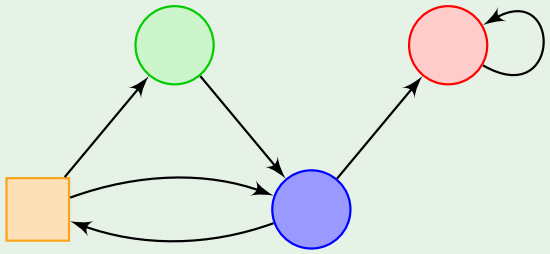
Reasoning about open systems

Strategies

A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Example

Strategy for player □



Reasoning about open systems

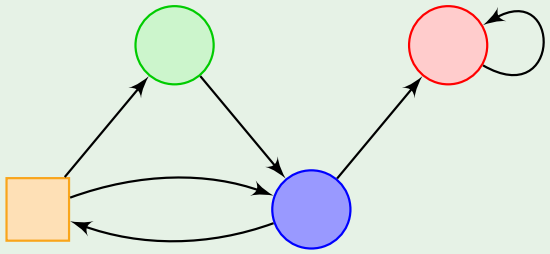
Strategies

A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Example

Strategy for player □

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



Reasoning about open systems

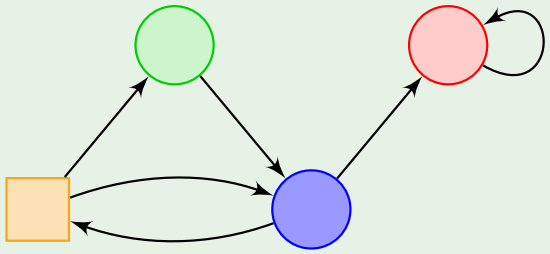
Strategies

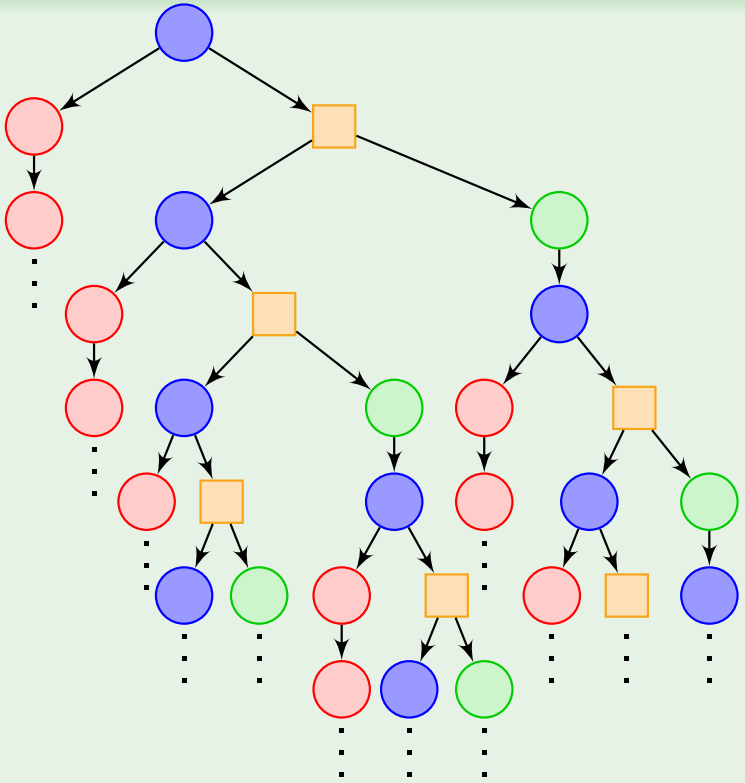
A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Example

Strategy for player □

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





Reasoning about open systems

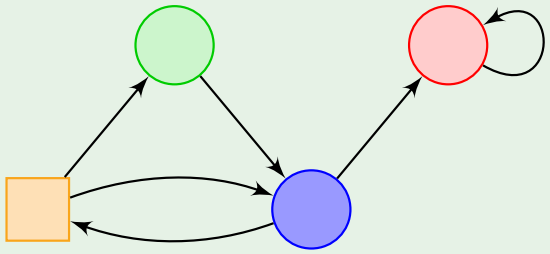
Strategies

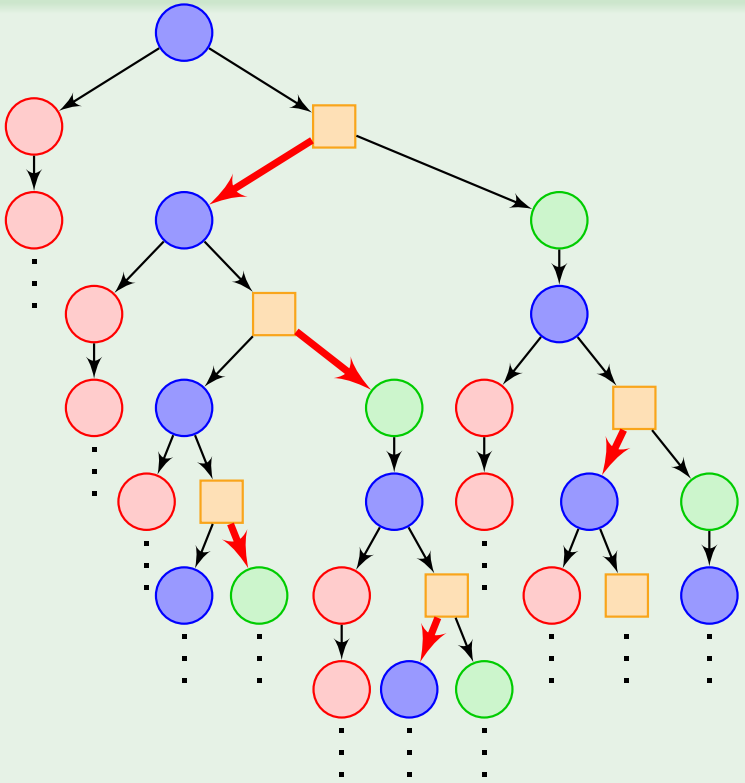
A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Example

Strategy for player □

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





Reasoning about open systems

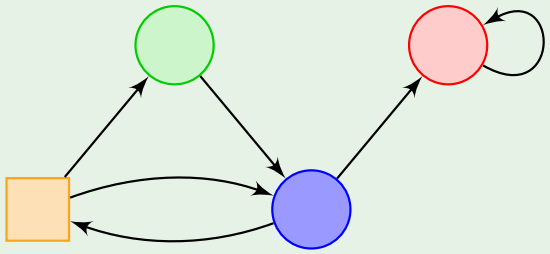
Strategies

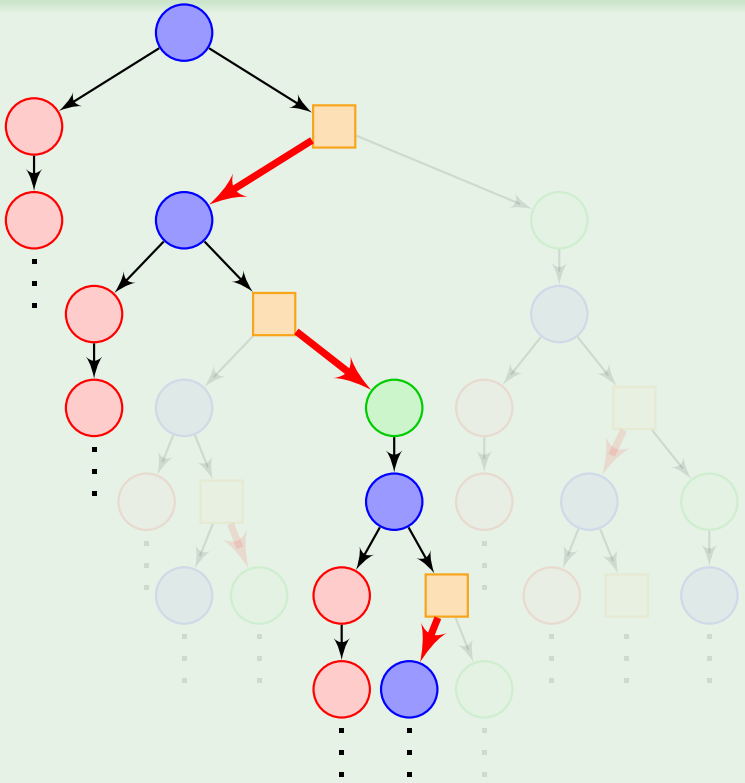
A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Example

Strategy for player □

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





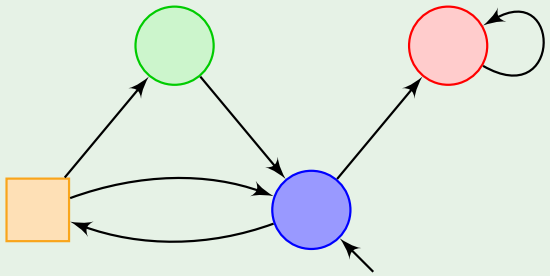
Reasoning about open systems

Strategies

A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Example

Memoryless strategy for player □



Reasoning about open systems

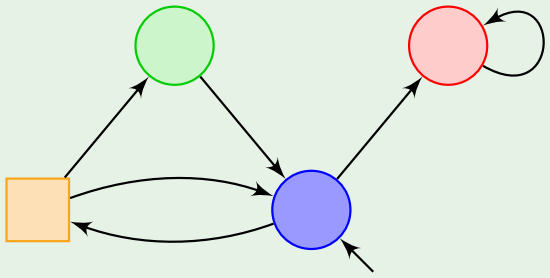
Strategies

A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Example

Memoryless strategy for player □

always go to ●.



Reasoning about open systems

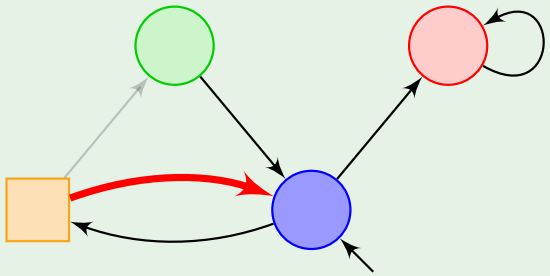
Strategies

A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Example

Memoryless strategy for player □

always go to ●.



Reasoning about open systems

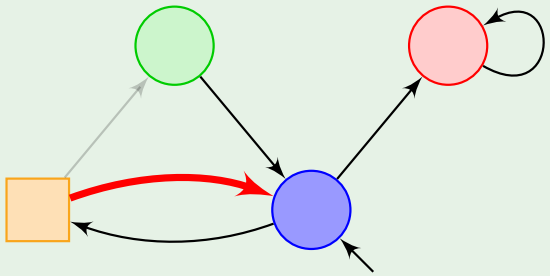
Strategies

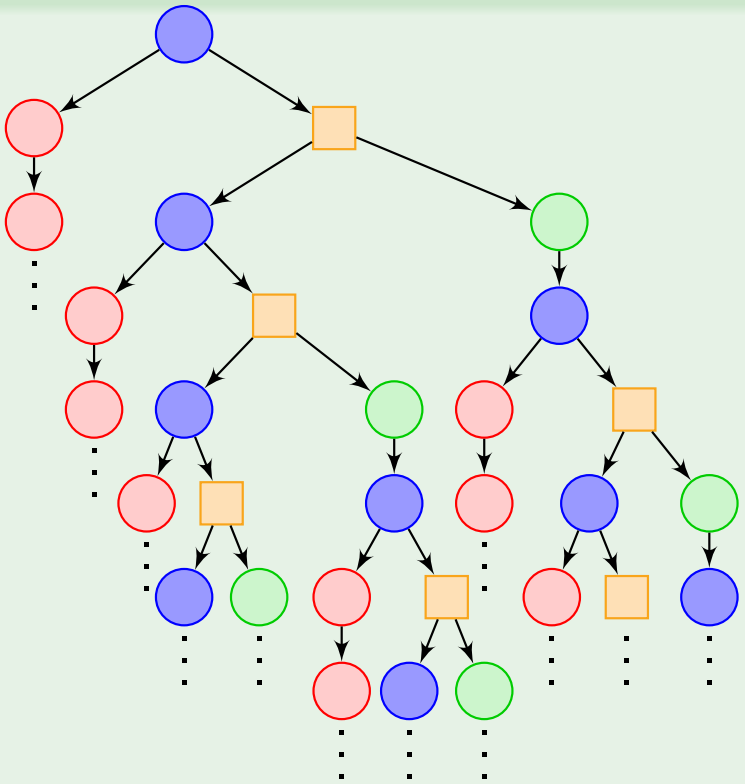
A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Example

Memoryless strategy for player □

always go to ●.





Reasoning about open systems

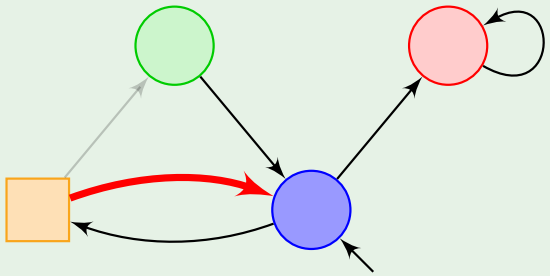
Strategies

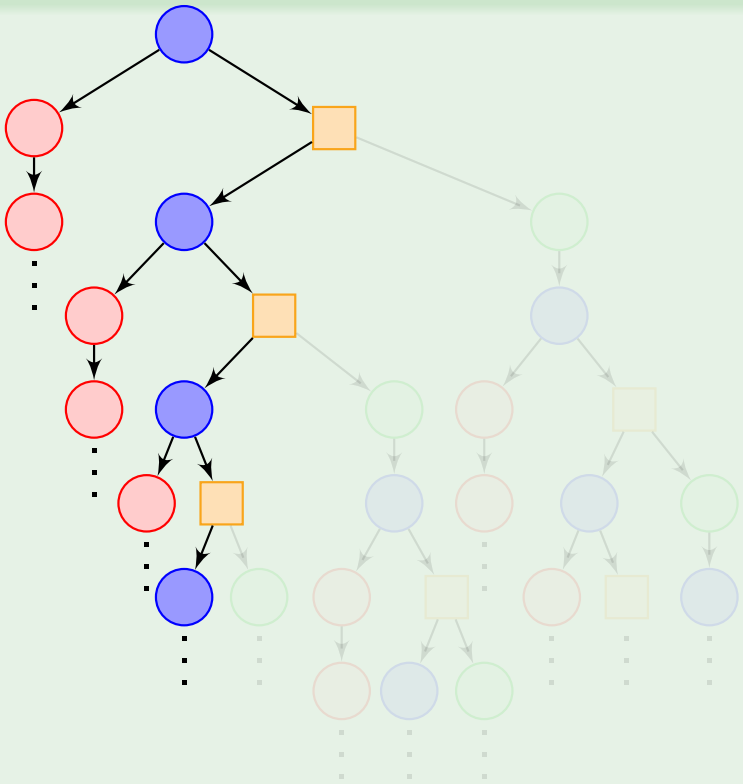
A (pure) strategy for a given player is a function telling which action to play depending on what has happened previously.

Example

Memoryless strategy for player □

always go to ●.





Temporal logics for games: ATL

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

ATL extends CTL with strategy quantifiers

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

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

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

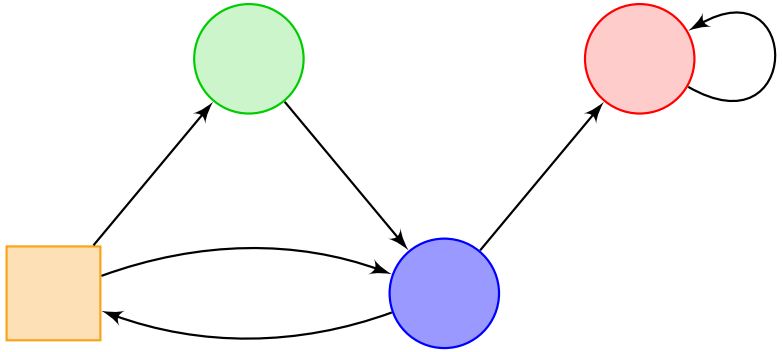
$$\mathcal{G}, \bigcirc \models \langle\langle A \rangle\rangle\, \varphi \quad \Longleftrightarrow \quad \exists \sigma_A.\ \forall \pi \in \mathrm{Out}(\bigcirc, \sigma_A).\ \pi \models \varphi.$$

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

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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



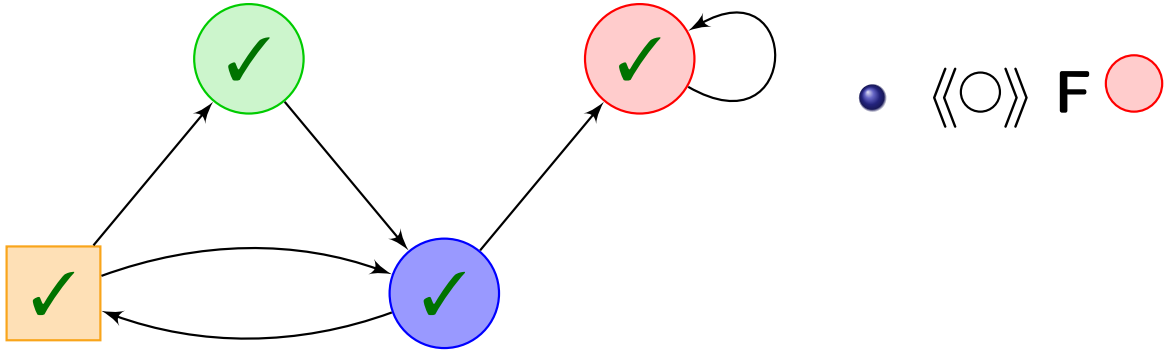
- $\langle\langle \bigcirc \rangle\rangle\, \mathbf{F}$

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

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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

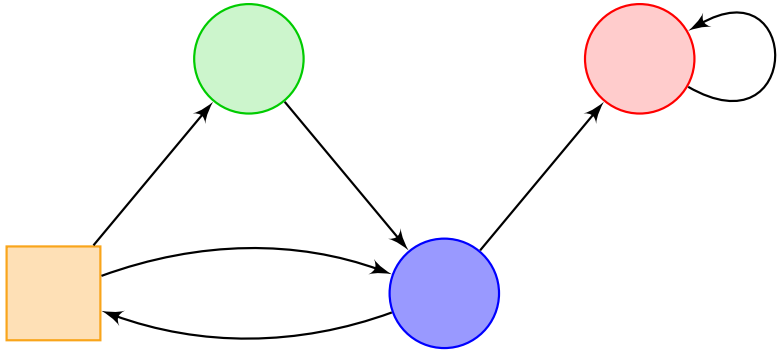


[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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



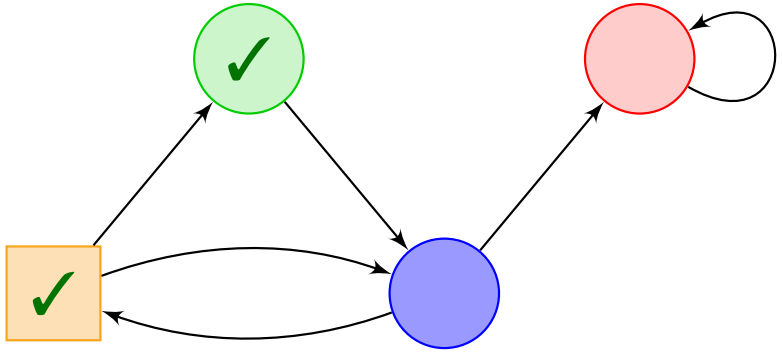
- $\langle\langle \bigcirc \rangle\rangle\, \mathbf{F}$
- $\langle\langle \Box \rangle\rangle\, \mathbf{F}$

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

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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



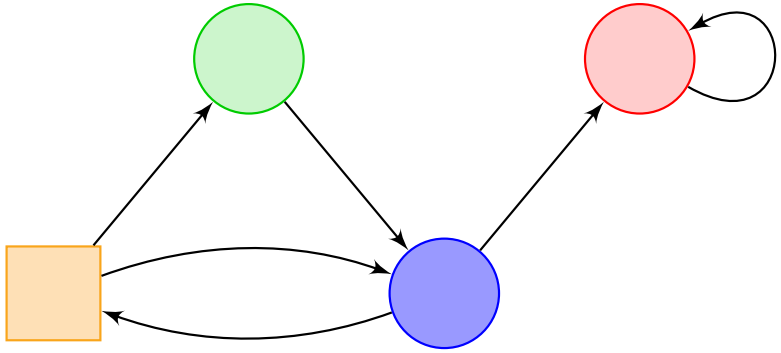
- $\langle\langle \bigcirc \rangle\rangle\, \mathbf{F}$
- $\langle\langle \Box \rangle\rangle\, \mathbf{F}$

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

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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



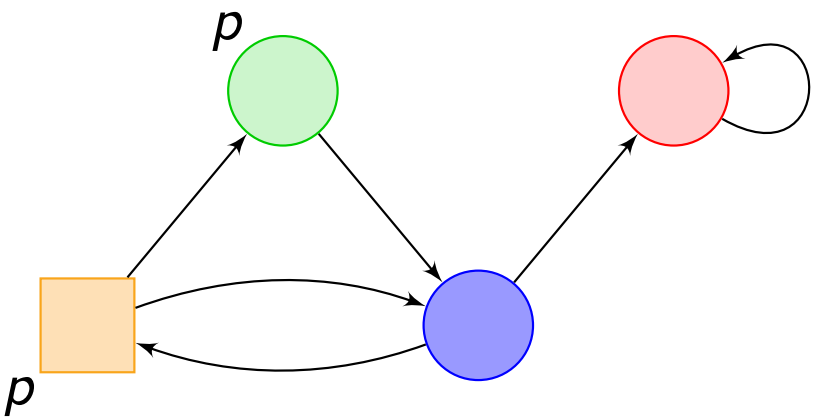
- $\langle\langle \bigcirc \rangle\rangle\ \mathbf{F}$
- $\langle\langle \Box \rangle\rangle\ \mathbf{F}$
- $\langle\langle \bigcirc \rangle\rangle\ \mathbf{G}(\ \langle\langle \Box \rangle\rangle\ \mathbf{F}\ ) \equiv \langle\langle \bigcirc \rangle\rangle\ \mathbf{G}\ p$

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

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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



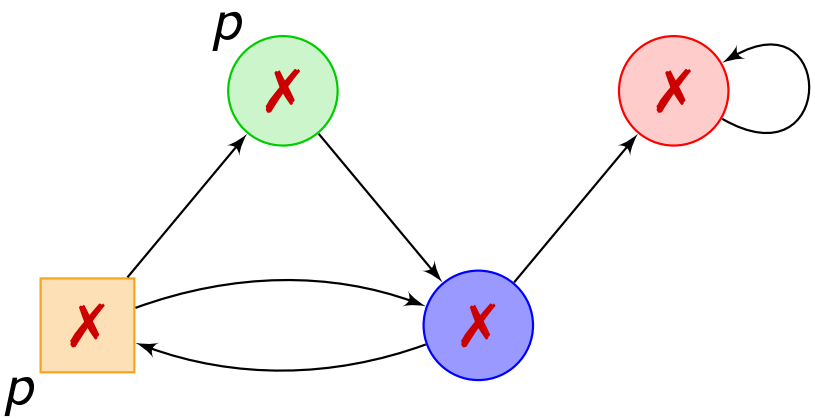
- $\langle\langle \bigcirc \rangle\rangle\, \mathbf{F}$
- $\langle\langle \Box \rangle\rangle\, \mathbf{F}$
- $\langle\langle \bigcirc \rangle\rangle\, \mathbf{G}(\underbrace{\langle\langle \Box \rangle\rangle\, \mathbf{F}}_{p}) \equiv \langle\langle \bigcirc \rangle\rangle\, \mathbf{G}\, p$

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

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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



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

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

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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

Theorem ([AHK02])

Model checking ATL is PTIME*-complete.*

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

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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

Theorem ([AHK02])

Model checking ATL is PTIME*-complete.*

For concurrent games, this assumes that the transition table is given explicitly (size $|\mathsf{Moves}|^{|\mathsf{Agt}|}$).

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

Temporal logics for games: ATL

ATL extends CTL with strategy quantifiers

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

Theorem ([AHK02])

Model checking ATL is PTIME*-complete.*

For concurrent games, this assumes that the transition table is given explicitly (size $|\mathsf{Moves}|^{|\mathsf{Agt}|}$).

Memoryless strategies are sufficient for ATL.

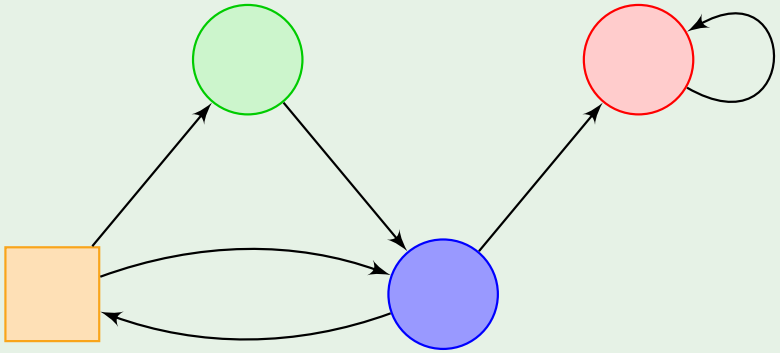
[AHK02] Alur, Henzinger, Kupferman. Alternating-time Temporal Logic. J. ACM, 2002.

Outline of the presentation

1. Introduction
2. Basics of CTL and ATL
   - expressing properties of reactive systems
   - efficient verification algorithms
3. Temporal logics for multi-agent systems
   - specifying properties of complex interacting systems
   - expressive power of ATL_{sc}
   - translation into Quantified CTL (QCTL)
   - algorithms for ATL_{sc}
4. Conclusions and future works

ATL with strategy contexts [BDLM09,DLM10]

Example



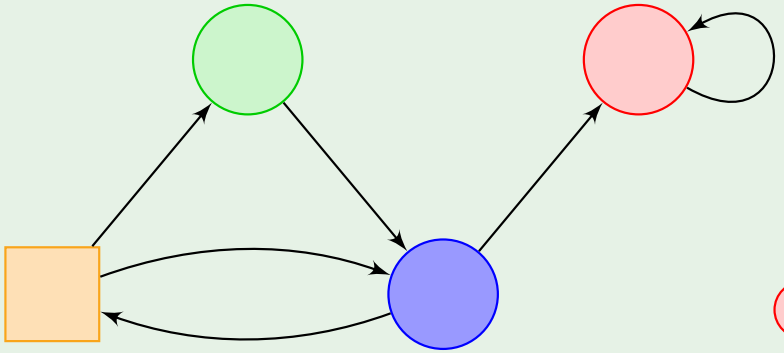
$\langle\langle \bigcirc \rangle\rangle\ \mathbf{G}(\ \langle\langle \Box \rangle\rangle\ \mathbf{F}\ \bigcirc\ )$

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

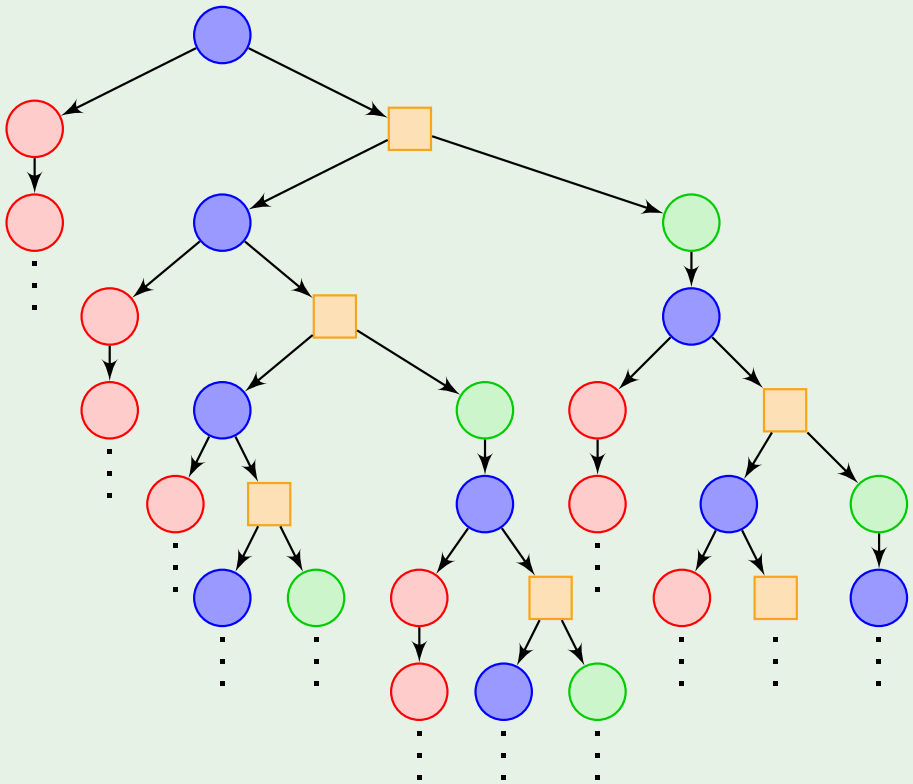
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.

ATL with strategy contexts [BDLM09,DLM10]

Example



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

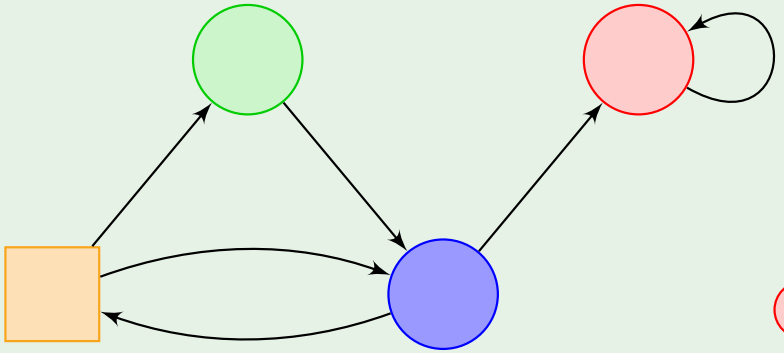


Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.

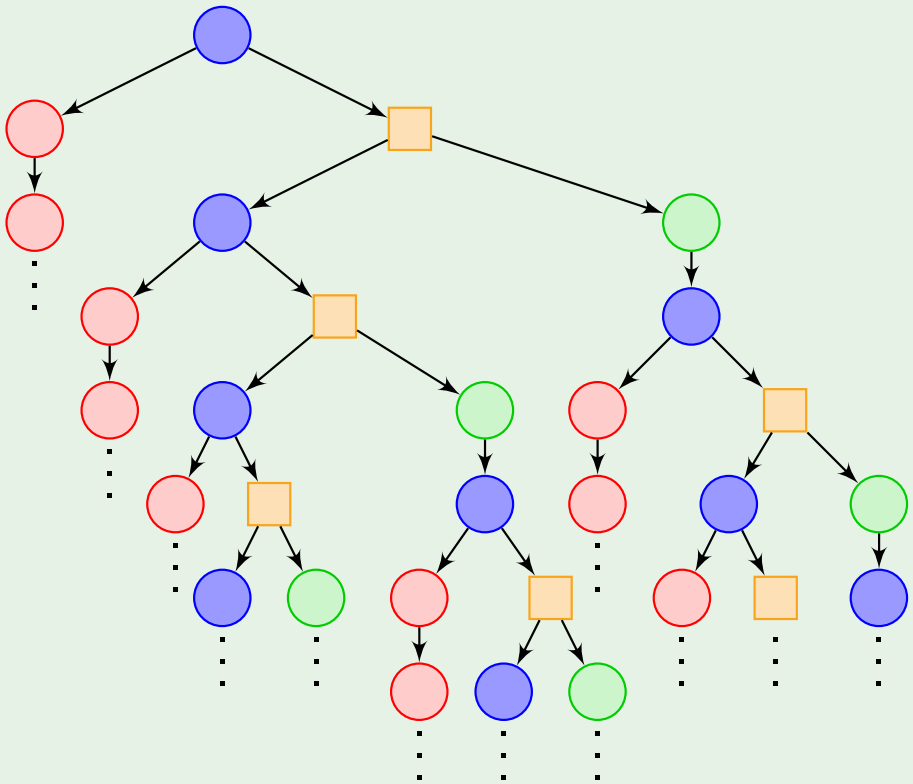
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.

ATL with strategy contexts [BDLM09,DLM10]

Example



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



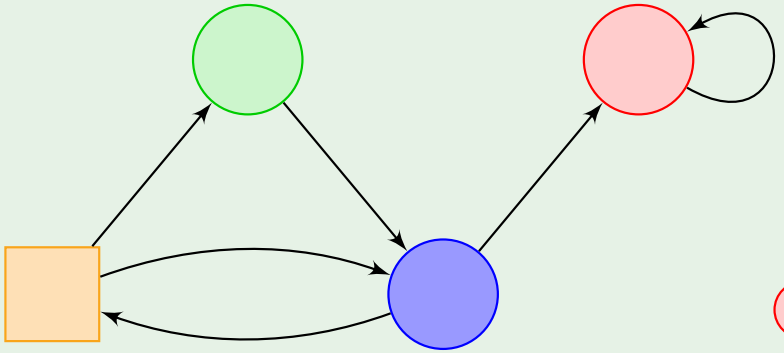
- Player ○ in ● always plays to □.

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

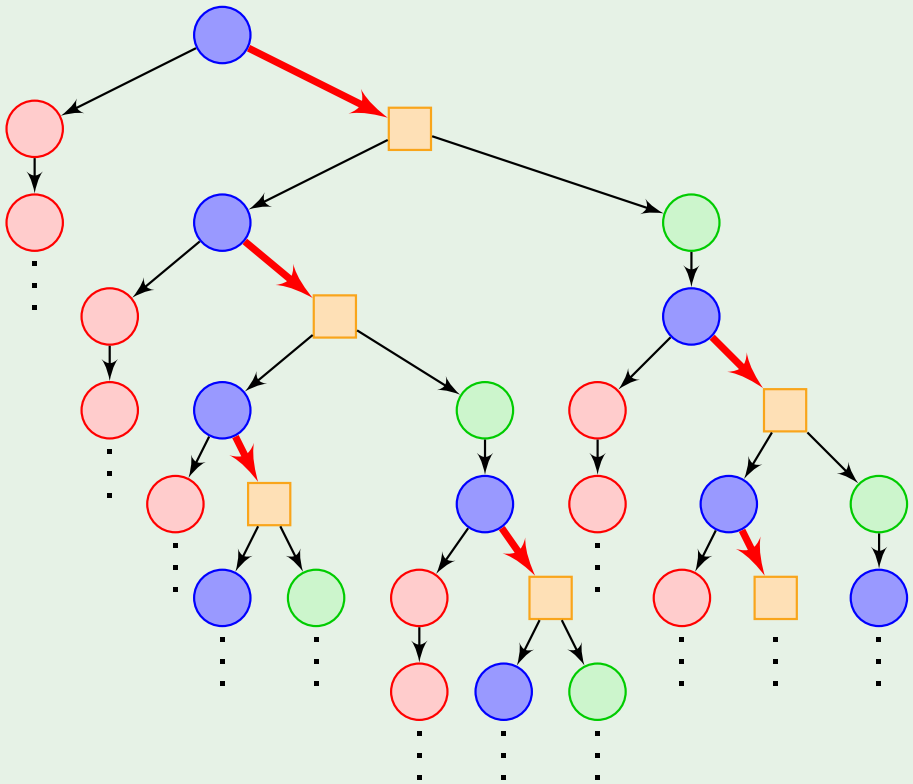
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.

ATL with strategy contexts [BDLM09,DLM10]

Example



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

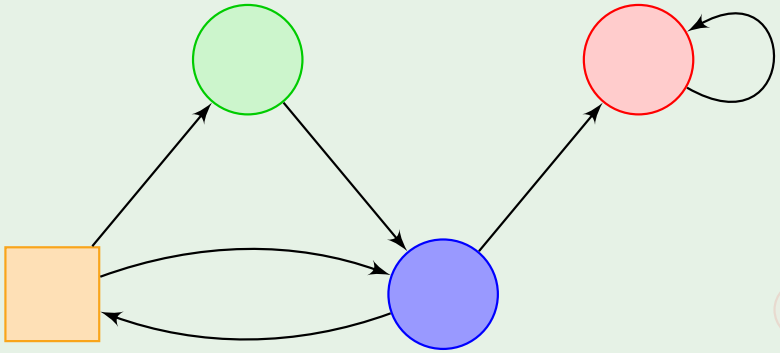


- Player ○ in ● always plays to □.

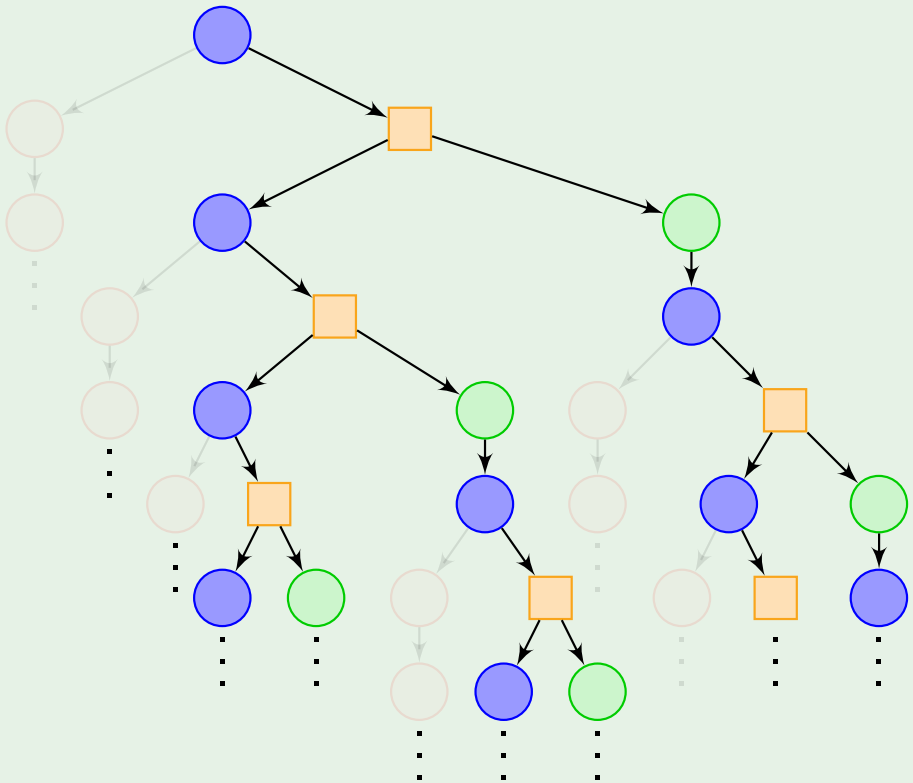
Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.

ATL with strategy contexts [BDLM09,DLM10]

Example



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

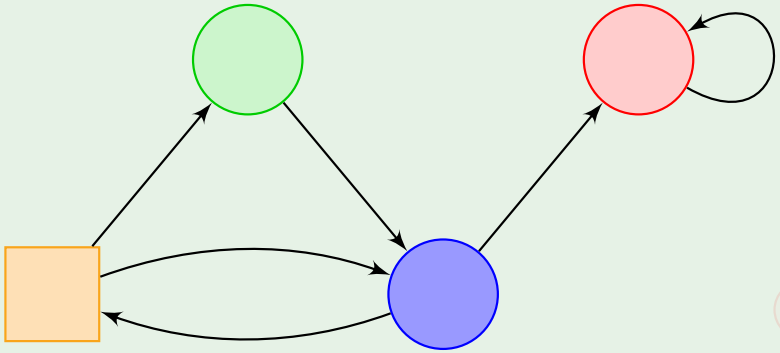


- Player ○ in ● always plays to □.

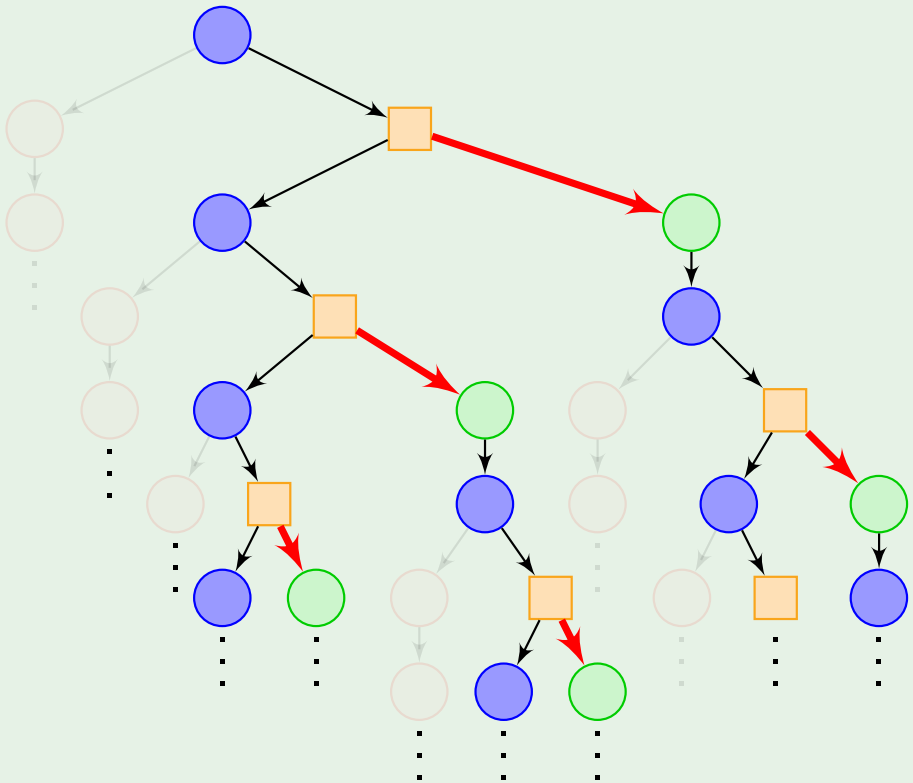
Brihaye, Da Costa, Laroussinie, Markey. ATL with strategy contexts and bounded memory. LFCS, 2009.
Da Costa, Laroussinie, Markey. ATL with strategy contexts: expressiveness and ... FSTTCS, 2010.

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 new strategy quantifiers:

- $\langle\cdot A\cdot\rangle\,\varphi$ is similar to $\langle\langle A\rangle\rangle\,\varphi$ but assigns the corresponding strategy to A for evaluating φ;

ATL with strategy contexts

Definition

ATL_{sc} has new strategy quantifiers:

- $\langle\cdot A\cdot\rangle\,\varphi$ is similar to $\langle\langle A\rangle\rangle\,\varphi$ but assigns the corresponding strategy to A for evaluating φ;
- $\langle\cdot\overline{A}\cdot\rangle\,\varphi \equiv \langle\cdot \mathrm{Agt} \setminus A\cdot\rangle\,\varphi$
  (useful for getting formulas that do not depend on Agt);

ATL with strategy contexts

Definition

ATL_{sc} has new strategy quantifiers:

- $\langle\cdot A\cdot\rangle\,\varphi$ is similar to $\langle\langle A\rangle\rangle\,\varphi$ but assigns the corresponding strategy to A for evaluating φ;
- $\langle\cdot\overline{A}\cdot\rangle\,\varphi \equiv \langle\cdot \mathrm{Agt} \setminus A\cdot\rangle\,\varphi$
  (useful for getting formulas that do not depend on Agt);
- $\langle\cdot A\cdot\rangle_0\,\varphi$ is similar to $\langle\cdot A\cdot\rangle\,\varphi$ but quantifies over memoryless strategies;

ATL with strategy contexts

Definition

ATL_{sc} has new strategy quantifiers:

- $\langle\cdot A\cdot\rangle\,\varphi$ is similar to $\langle\langle A\rangle\rangle\,\varphi$ but assigns the corresponding strategy to A for evaluating φ;
- $\langle\cdot\overline{A}\cdot\rangle\,\varphi \equiv \langle\cdot \mathrm{Agt} \setminus A\cdot\rangle\,\varphi$
  (useful for getting formulas that do not depend on Agt);
- $\langle\cdot A\cdot\rangle_0\,\varphi$ is similar to $\langle\cdot A\cdot\rangle\,\varphi$ but quantifies over memoryless strategies;
- $(\!|A|\!)\,\varphi$ drops the assigned strategies for A.

ATL with strategy contexts

Definition

ATL_{sc} has new strategy quantifiers:

- $\langle\cdot A\cdot\rangle\,\varphi$ is similar to $\langle\langle A\rangle\rangle\,\varphi$ but assigns the corresponding strategy to A for evaluating φ;
- $\langle\cdot\overline{A}\cdot\rangle\,\varphi \equiv \langle\cdot \mathrm{Agt} \setminus A\cdot\rangle\,\varphi$
  (useful for getting formulas that do not depend on Agt);
- $\langle\cdot A\cdot\rangle_0\,\varphi$ is similar to $\langle\cdot A\cdot\rangle\,\varphi$ but quantifies over memoryless strategies;
- $(\!|A|\!)\,\varphi$ drops the assigned strategies for A.
- $[\cdot A\cdot]\,\varphi$ is dual to $\langle\cdot A\cdot\rangle\,\varphi$:

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

ATL with strategy contexts

Definition

ATL_{sc} has new strategy quantifiers:

- $\langle\cdot A\cdot\rangle\,\varphi$ is similar to $\langle\langle A\rangle\rangle\,\varphi$ but assigns the corresponding strategy to A for evaluating φ;

Definition

Semantics of ATL strategy quantifier:

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

ATL with strategy contexts

Definition

ATL_{sc} has new strategy quantifiers:

- $\langle\cdot A\cdot\rangle\,\varphi$ is similar to $\langle\langle A\rangle\rangle\,\varphi$ but assigns the corresponding strategy to A for evaluating φ;

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:

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

ATL with strategy contexts

Definition

ATL_{sc} has new strategy quantifiers:

- $\langle\cdot A\cdot\rangle\,\varphi$ is similar to $\langle\langle A\rangle\rangle\,\varphi$ but assigns the corresponding strategy to A for evaluating φ;

Definition

Semantics of ATL_{sc} strategy quantifier:

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

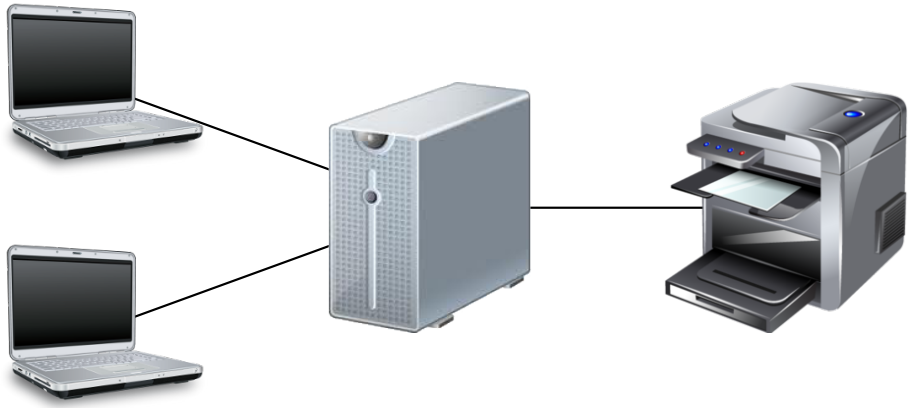
$\rightsquigarrow$ newly selected strategies added to the context:

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

What ATL_{sc} can express

- 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

- 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

- 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:

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

Expressiveness of ATL_{sc}

Theorem

ATL_{sc} is strictly more expressive than ATL

Expressiveness of ATL_{sc}

Theorem

ATL_{sc} is strictly more expressive than ATL

Proof

$$\langle\langle A \rangle\rangle\, \varphi \equiv (\!|\overline{\emptyset}|\!)\ \langle\cdot A \cdot\rangle\, \hat{\varphi}$$

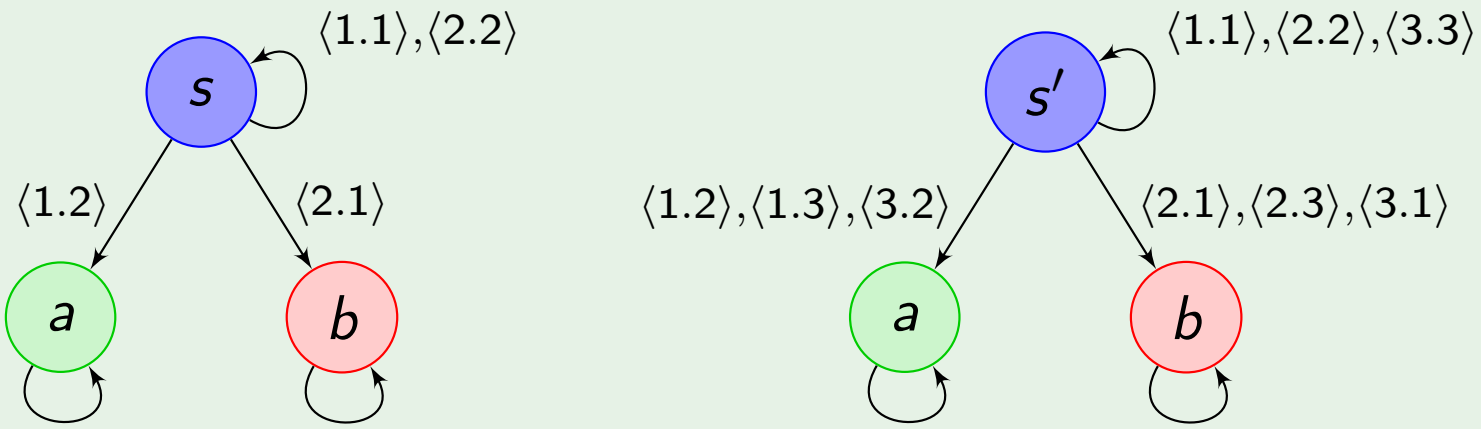
Expressiveness of ATL_{sc}

Theorem

ATL_{sc} is strictly more expressive than ATL

Proof

$\langle\cdot 1\cdot\rangle\,(\,\langle\cdot 2\cdot\rangle\ \mathbf{X}\ a \wedge\ \langle\cdot 2\cdot\rangle\ \mathbf{X}\ b)$ is only true in the second game.
But ATL cannot distinguish between these two games.



Outline of the presentation

1. Introduction
2. Basics of CTL and ATL
   - expressing properties of reactive systems
   - efficient verification algorithms
3. Temporal logics for multi-agent systems
   - specifying properties of complex interacting systems
   - expressive power of ATL_{sc}
   - translation into Quantified CTL (QCTL)
   - algorithms for ATL_{sc}
4. Conclusions and future works

Quantified CTL [ES84,Kup95,Fre01]

QCTL extends CTL with propositional quantifiers

$\exists p.\ \varphi$ means that there exists a labelling of the model with p under which φ holds.

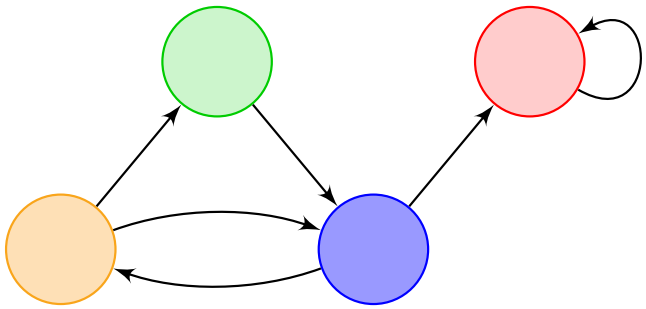
[ES84] Emerson and Sistla. Deciding Full Branching Time Logic. Information & Control, 1984.
[Kup95] Kupferman. Augmenting Branching Temporal Logics with Existential Quantification... CAV, 1995.
[Fre01] French. Decidability of Quantifed Propositional Branching Time Logics. AJCAI, 2001.

Quantified CTL [ES84,Kup95,Fre01]

QCTL extends CTL with propositional quantifiers

$\exists p.\ \varphi$ means that there exists a labelling of the model with p under which φ holds.

- $\mathbf{E}\,\mathbf{F}\,\bigcirc \wedge \forall p.\ \big[\mathbf{E}\,\mathbf{F}(p \wedge \bigcirc) \Rightarrow \mathbf{A}\,\mathbf{G}(\bigcirc \Rightarrow p)\big]$



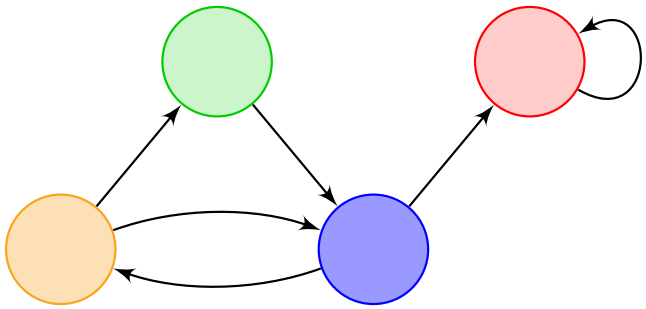
[ES84] Emerson and Sistla. Deciding Full Branching Time Logic. Information & Control, 1984.
[Kup95] Kupferman. Augmenting Branching Temporal Logics with Existential Quantification... CAV, 1995.
[Fre01] French. Decidability of Quantifed Propositional Branching Time Logics. AJCAI, 2001.

Quantified CTL [ES84,Kup95,Fre01]

QCTL extends CTL with propositional quantifiers

$\exists p.\ \varphi$ means that there exists a labelling of the model with p under which φ holds.

- $\mathbf{E}\,\mathbf{F}\,\bigcirc \wedge \forall p.\ \big[\mathbf{E}\,\mathbf{F}(p \wedge \bigcirc) \Rightarrow \mathbf{A}\,\mathbf{G}(\bigcirc \Rightarrow p)\big] \equiv \mathrm{uniq}(\bigcirc)$



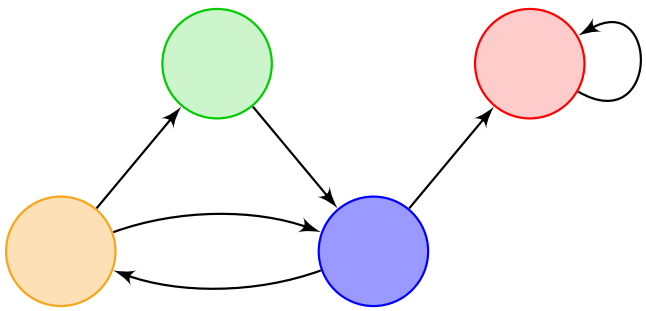
[ES84] Emerson and Sistla. Deciding Full Branching Time Logic. Information & Control, 1984.
[Kup95] Kupferman. Augmenting Branching Temporal Logics with Existential Quantification... CAV, 1995.
[Fre01] French. Decidability of Quantifed Propositional Branching Time Logics. AJCAI, 2001.

Quantified CTL [ES84,Kup95,Fre01]

QCTL extends CTL with propositional quantifiers

$\exists p.\ \varphi$ means that there exists a labelling of the model with p under which φ holds.

- $\mathbf{E}\,\mathbf{F}\,\bigcirc \wedge \forall p.\ \big[\mathbf{E}\,\mathbf{F}(p \wedge \bigcirc) \Rightarrow \mathbf{A}\,\mathbf{G}(\bigcirc \Rightarrow p)\big] \equiv \mathrm{uniq}(\bigcirc)$



$\rightsquigarrow$ true if we label the Kripke structure;

$\rightsquigarrow$ false if we label the computation tree;

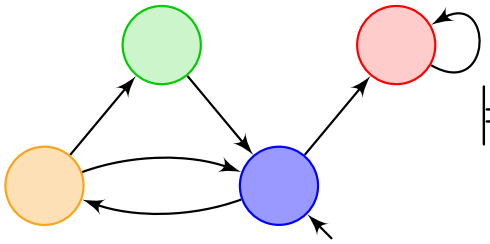
[ES84] Emerson and Sistla. Deciding Full Branching Time Logic. Information & Control, 1984.

[Kup95] Kupferman. Augmenting Branching Temporal Logics with Existential Quantification... CAV, 1995.

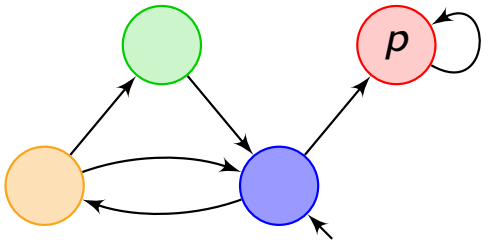
[Fre01] French. Decidability of Quantifed Propositional Branching Time Logics. AJCAI, 2001.

Semantics of QCTL

- structure semantics:



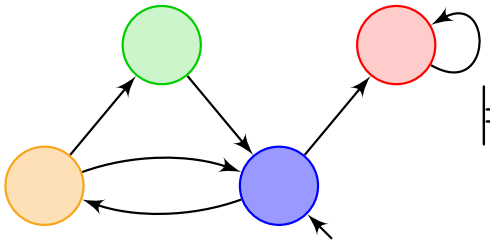
$\models_s \exists p.\varphi \quad \Leftrightarrow$



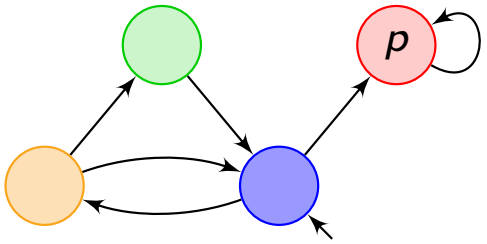
$\models \varphi$

Semantics of QCTL

- structure semantics:

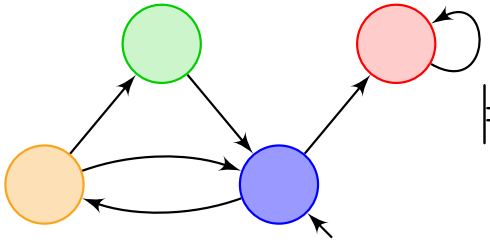


$\models_s \exists p.\varphi \quad \Leftrightarrow$

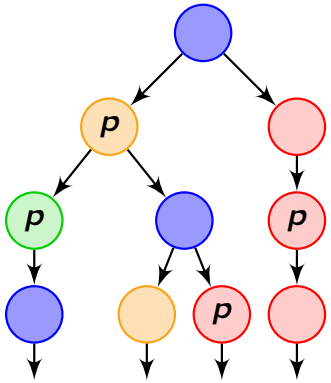


$\models \varphi$

- tree semantics:



$\models_t \exists p.\varphi \quad \Leftrightarrow$



$\models \varphi$

Expressiveness of QCTL

- QCTL can "count":

$$\mathbf{E}\,\mathbf{X}_1\,\varphi \equiv \mathbf{E}\,\mathbf{X}\,\varphi \wedge \forall p.\ [\mathbf{E}\,\mathbf{X}(p \wedge \varphi) \Rightarrow \mathbf{A}\,\mathbf{X}(\varphi \Rightarrow p)]$$
$$\mathbf{E}\,\mathbf{X}_2\,\varphi \equiv \exists q.\ [\mathbf{E}\,\mathbf{X}_1(\varphi \wedge q) \wedge \mathbf{E}\,\mathbf{X}_1(\varphi \wedge \neg q)]$$

[DLM12] Da Costa, Laroussinie, Markey. Quantified CTL: ... CONCUR, 2012.

Expressiveness of QCTL

- QCTL can "count":

$$\mathbf{E}\,\mathbf{X}_1\,\varphi \equiv \mathbf{E}\,\mathbf{X}\,\varphi \wedge \forall p.\ [\mathbf{E}\,\mathbf{X}(p \wedge \varphi) \Rightarrow \mathbf{A}\,\mathbf{X}(\varphi \Rightarrow p)]$$
$$\mathbf{E}\,\mathbf{X}_2\,\varphi \equiv \exists q.\ [\mathbf{E}\,\mathbf{X}_1(\varphi \wedge q) \wedge \mathbf{E}\,\mathbf{X}_1(\varphi \wedge \neg q)]$$

- QCTL can express (least or greatest) fixpoints:

$$\mu T.\varphi(T) \equiv \exists t.\ [\mathbf{A}\,\mathbf{G}(t \iff \varphi(t)) \wedge (\forall t.'(\mathbf{A}\,\mathbf{G}(t' \iff \varphi(t')) \Rightarrow \mathbf{A}\,\mathbf{G}(t \Rightarrow t')))]$$

[DLM12] Da Costa, Laroussinie, Markey. Quantified CTL: ... CONCUR, 2012.

Expressiveness of QCTL

- QCTL can "count":

$$\mathbf{E}\,\mathbf{X}_1\,\varphi \equiv \mathbf{E}\,\mathbf{X}\,\varphi \wedge \forall p.\ [\mathbf{E}\,\mathbf{X}(p \wedge \varphi) \Rightarrow \mathbf{A}\,\mathbf{X}(\varphi \Rightarrow p)]$$
$$\mathbf{E}\,\mathbf{X}_2\,\varphi \equiv \exists q.\ [\mathbf{E}\,\mathbf{X}_1(\varphi \wedge q) \wedge \mathbf{E}\,\mathbf{X}_1(\varphi \wedge \neg q)]$$

- QCTL can express (least or greatest) fixpoints:

$$\mu T.\varphi(T) \equiv \exists t.\ [\mathbf{A}\,\mathbf{G}(t \iff \varphi(t)) \wedge (\forall t.'(\mathbf{A}\,\mathbf{G}(t' \iff \varphi(t')) \Rightarrow \mathbf{A}\,\mathbf{G}(t \Rightarrow t')))]$$

Theorem

QCTL, QCTL* and MSO are equally expressive (under both semantics).

[DLM12] Da Costa, Laroussinie, Markey. Quantified CTL: ... CONCUR, 2012.

QCTL with structure semantics

Theorem

Model checking QCTL for the structure semantics is PSPACE-*complete.*

[DLM12] Da Costa, Laroussinie, Markey. Quantified CTL: ... CONCUR, 2012.

QCTL with structure semantics

Theorem

Model checking QCTL for the structure semantics is PSPACE-*complete.*

Proof

Membership: labelling algorithm.

Iteratively
- (nondeterministically) pick a labelling,
- check the subformula.

Hardness:
QBF is a special case (without even using temporal modalities).

[DLM12] Da Costa, Laroussinie, Markey. Quantified CTL: ... CONCUR, 2012.

QCTL with structure semantics

Theorem

Model checking QCTL for the structure semantics is PSPACE-*complete.*

Proof

Membership: labelling algorithm.

Iteratively
- (nondeterministically) pick a labelling,
- check the subformula.

Hardness:
QBF is a special case (without even using temporal modalities).

Theorem

QCTL satisfiability for the structure semantics is undecidable.

[DLM12] Da Costa, Laroussinie, Markey. Quantified CTL: ... CONCUR, 2012.

QCTL with tree semantics

Theorem

- *Model checking QCTL with k quantifiers in the tree semantics is* k-EXPTIME-*complete.*
- *Satisfiability of QCTL with k quantifiers in the tree semantics is* (k+1)-EXPTIME-*complete.*

[DLM12] Da Costa, Laroussinie, Markey. Quantified CTL: ... CONCUR, 2012.

QCTL with tree semantics

Theorem

- *Model checking QCTL with k quantifiers in the tree semantics is* k-EXPTIME-*complete.*
- *Satisfiability of QCTL with k quantifiers in the tree semantics is* (k+1)-EXPTIME-*complete.*

Proof

Using (alternating) parity tree automata:

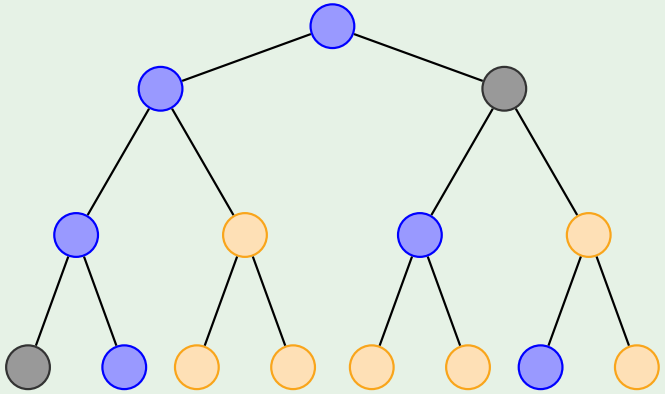
QCTL with tree semantics

Theorem

- *Model checking QCTL with k quantifiers in the tree semantics is* k-EXPTIME-*complete.*
- *Satisfiability of QCTL with k quantifiers in the tree semantics is* (k+1)-EXPTIME-*complete.*

Proof

Using (alternating) parity tree automata:



QCTL with tree semantics

Theorem

- *Model checking QCTL with k quantifiers in the tree semantics is* k-EXPTIME-*complete.*
- *Satisfiability of QCTL with k quantifiers in the tree semantics is* (k+1)-EXPTIME-*complete.*

Proof

Using (alternating) parity tree automata:

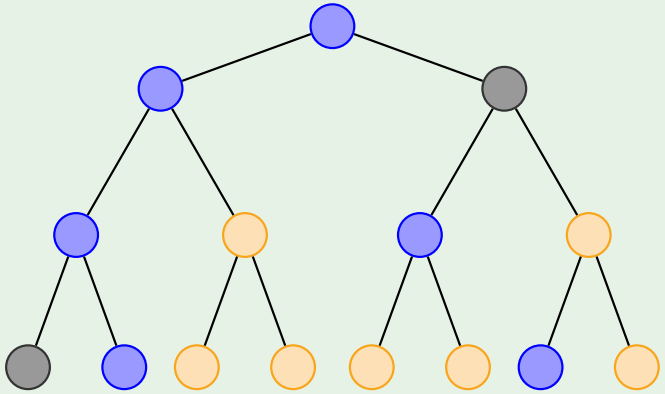
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

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

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

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

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



QCTL with tree semantics

Theorem

- *Model checking QCTL with k quantifiers in the tree semantics is* k-EXPTIME-*complete.*
- *Satisfiability of QCTL with k quantifiers in the tree semantics is* (k+1)-EXPTIME-*complete.*

Proof

Using (alternating) parity tree automata:

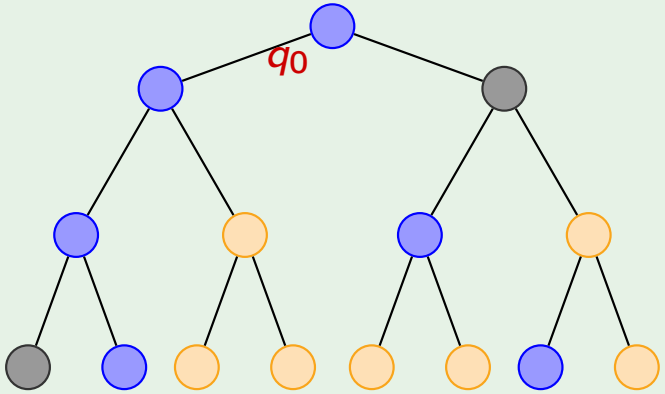
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

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

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

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

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



QCTL with tree semantics

Theorem

- *Model checking QCTL with k quantifiers in the tree semantics is* k-EXPTIME-*complete.*
- *Satisfiability of QCTL with k quantifiers in the tree semantics is* (k+1)-EXPTIME-*complete.*

Proof

Using (alternating) parity tree automata:

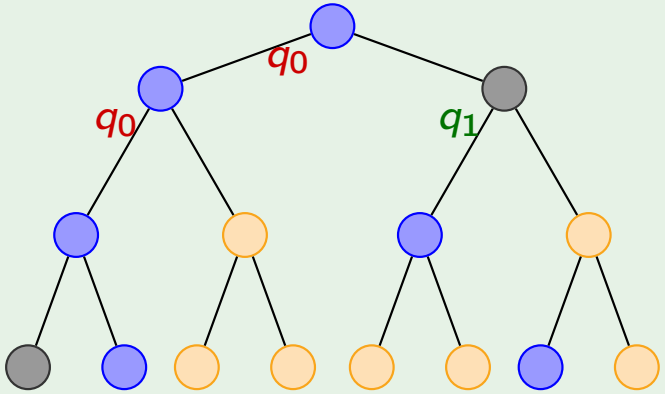
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

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

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

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

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



QCTL with tree semantics

Theorem

- *Model checking QCTL with k quantifiers in the tree semantics is* k-EXPTIME-*complete.*
- *Satisfiability of QCTL with k quantifiers in the tree semantics is* (k+1)-EXPTIME-*complete.*

Proof

Using (alternating) parity tree automata:

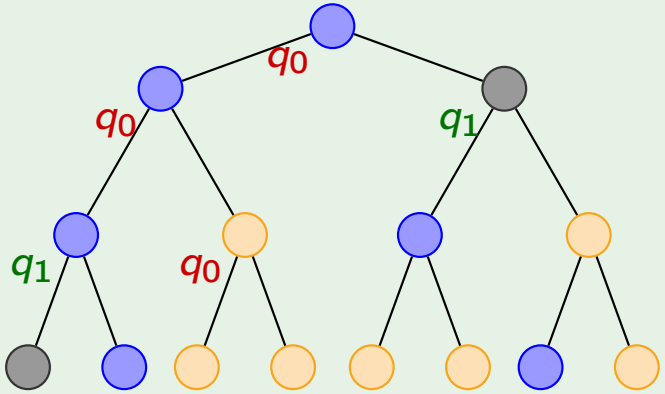
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

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

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

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

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



QCTL with tree semantics

Theorem

- *Model checking QCTL with k quantifiers in the tree semantics is* k-EXPTIME-*complete.*
- *Satisfiability of QCTL with k quantifiers in the tree semantics is* (k+1)-EXPTIME-*complete.*

Proof

Using (alternating) parity tree automata:

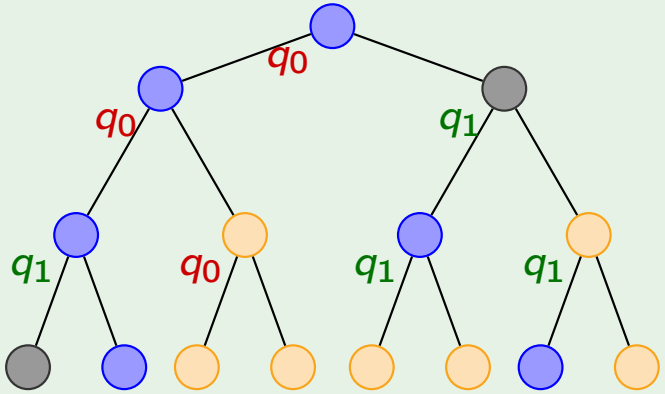
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

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

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

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

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



QCTL with tree semantics

Theorem

- *Model checking QCTL with k quantifiers in the tree semantics is* k-EXPTIME-*complete.*
- *Satisfiability of QCTL with k quantifiers in the tree semantics is* (k+1)-EXPTIME-*complete.*

Proof

Using (alternating) parity tree automata:

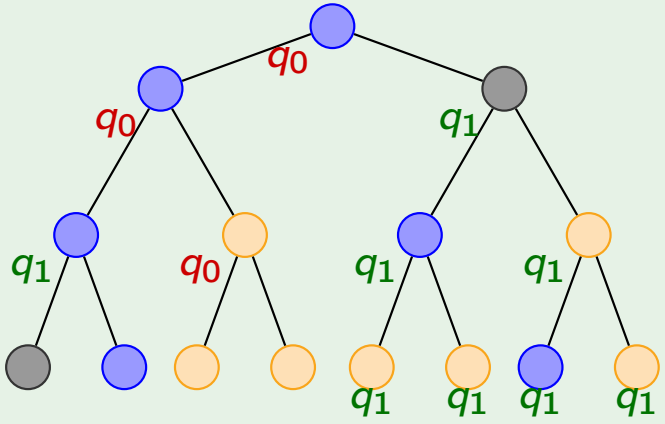
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

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

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

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

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



QCTL with tree semantics

Theorem

- *Model checking QCTL with k quantifiers in the tree semantics is* k-EXPTIME-*complete.*
- *Satisfiability of QCTL with k quantifiers in the tree semantics is* (k+1)-EXPTIME-*complete.*

Proof

Using (alternating) parity tree automata:

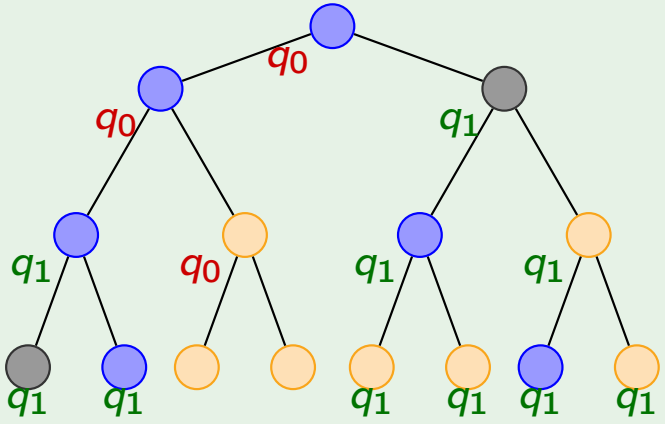
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

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

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

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

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



QCTL with tree semantics

Theorem

- *Model checking QCTL with k quantifiers in the tree semantics is* k-EXPTIME-*complete.*
- *Satisfiability of QCTL with k quantifiers in the tree semantics is* (k+1)-EXPTIME-*complete.*

Proof

Using (alternating) parity tree automata:

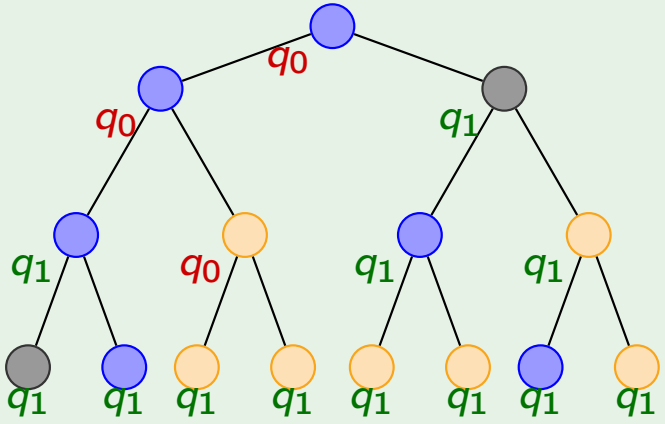
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

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

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

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

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



QCTL with tree semantics

Theorem

- *Model checking QCTL with k quantifiers in the tree semantics is* k-EXPTIME-*complete.*
- *Satisfiability of QCTL with k quantifiers in the tree semantics is* (k+1)-EXPTIME-*complete.*

Proof

Using (alternating) parity tree automata:

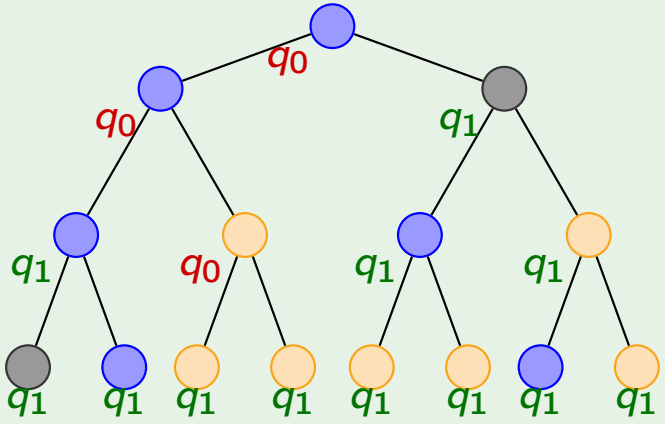
$\delta(q_0, \bullet) = (q_0, q_1) \vee (q_1, q_0)$

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

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

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

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



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

QCTL with tree semantics

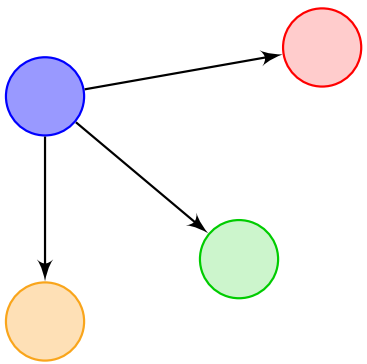
Theorem

- *Model checking QCTL with k quantifiers in the tree semantics is* k-EXPTIME-*complete.*
- *Satisfiability of QCTL with k quantifiers in the tree semantics is* (k+1)-EXPTIME-*complete.*

Proof

- polynomial-size tree automata for CTL;
- quantification is handled by projection, which first requires removing alternation (exponential blowup);
- an automaton equivalent to a QCTL formula can be built inductively;
- emptiness of an alternating parity tree automaton can be decided in exponential time.

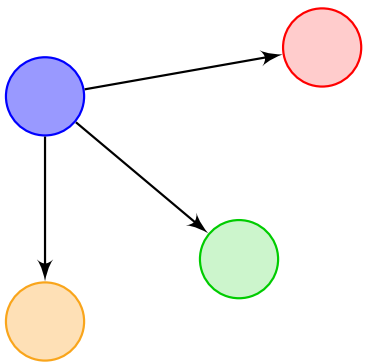
Translating ATL_{sc} into QCTL



- player A has moves m_1^A, ..., m_n^A;
- from the transition table, we can compute the set Next(, A, m_i^A) of states that can be reached from when player A plays m_i^A.

[DLM12] Da Costa, Laroussinie, Markey. Quantified CTL: ... CONCUR, 2012.

Translating ATL_{sc} into QCTL



- player A has moves m_1^A, ..., m_n^A;
- from the transition table, we can compute the set $\mathrm{Next}(\bullet, A, m_i^A)$ of states that can be reached from $\bullet$ when player A plays m_i^A.

$\langle \cdot A \cdot \rangle\, \varphi$ can be encoded as follows:

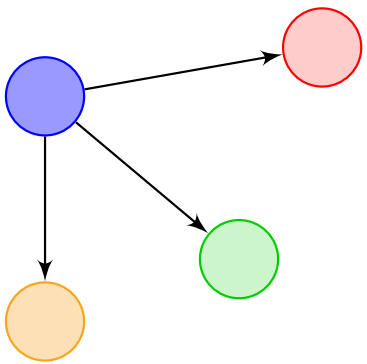
$\exists m_1^A.\ \exists m_2^A \ldots \exists m_n^A.$

- this corresponds to a strategy: $\mathbf{A}\,\mathbf{G}(m_i^A \Leftrightarrow \bigwedge \neg\, m_j^A)$;
- the outcomes all satisfy φ:

$$\mathbf{A}\big[\mathbf{G}(q \wedge m_i^A \Rightarrow \mathbf{X}\,\mathrm{Next}(q, A, m_i^A)) \Rightarrow \varphi\big].$$

[DLM12] Da Costa, Laroussinie, Markey. Quantified CTL: ... CONCUR, 2012.

Translating ATL_{sc} into QCTL



- player A has moves m_1^A, ..., m_n^A;
- from the transition table, we can compute the set Next(●, A, m_i^A) of states that can be reached from ● when player A plays m_i^A.

Corollary

*ATL_{sc} model checking is decidable, with non-elementary complexity (*TOWER-*complete).*

Corollary

ATL_{sc}^0 (quantification restricted to memoryless strategies) model checking is PSPACE-*complete.*

[DLM12] Da Costa, Laroussinie, Markey. Quantified CTL: ... CONCUR, 2012.

What about satisfiability?

Theorem

QCTL satisfiability is decidable (for the tree semantics).

What about satisfiability?

Theorem

QCTL satisfiability is decidable (for the tree semantics).

But

Theorem ([TW12])

ATL_{sc} satisfiability is undecidable.

[TW12] Troquard, Walther. On Satisfiability in ATL with Strategy Contexts. JELIA, 2012.

What about satisfiability?

Theorem

QCTL satisfiability is decidable (for the tree semantics).

But

Theorem ([TW12])

ATL_{sc} satisfiability is undecidable.

Why?

The translation from ATL_{sc} to QCTL assumes that the game structure is given!

[TW12] Troquard, Walther. On Satisfiability in ATL with Strategy Contexts. JELIA, 2012.

Satisfiability for turn-based games

Theorem ([LM13])

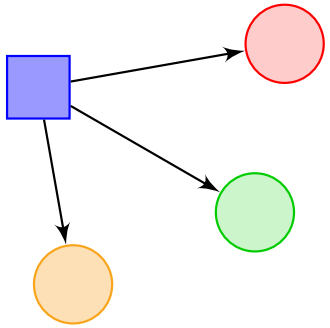
When restricted to turn-based games, ATL_{sc} satisfiability is decidable.

[LM13] Laroussinie, Markey. Satisfiability of ATL with strategy contexts. Gandalf, 2013.

Satisfiability for turn-based games

Theorem ([LM13])

When restricted to turn-based games, ATL_{sc} satisfiability is decidable.



- player $\square$ has moves ●, ● and ●.
- a strategy can be encoded by marking some of the nodes of the tree with proposition mov_A.

$\langle\cdot A\cdot\rangle\,\varphi$ can be encoded as follows:

$\exists \mathrm{mov}_A.$

- it corresponds to a strategy: $\mathbf{A}\,\mathbf{G}(\mathrm{turn}_A \Rightarrow \mathbf{E}\,\mathbf{X}_1\,\mathrm{mov}_A)$;
- the outcomes all satisfy φ: $\mathbf{A}\big[\mathbf{G}(\mathrm{turn}_A \wedge \mathbf{X}\,\mathrm{mov}_A) \Rightarrow \varphi\big]$.

[LM13] Laroussinie, Markey. Satisfiability of ATL with strategy contexts. Gandalf, 2013.

What about Strategy Logic? [CHP07,MMV10]

Strategy logic

Explicit quantification over strategies + strategy assignement

Example

$$\langle \cdot A \cdot \rangle \, \varphi \equiv \exists \sigma_1 . \mathsf{assign}(\sigma_1, A) . \varphi$$

Strategy logic can also be translated into QCTL.

Theorem

- *Strategy-logic model-checking is decidable.*
- *Strategy-logic satisfiability is decidable when restricted to turn-based games.*

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

Conclusions and future works

Conclusions

- ATL_{sc} is a very expressive, yet decidable extension of ATL;
- QCTL is a powerful extension of CTL;
- it is a nice tool to understand temporal logics for games (ATL_{sc}, Strategy Logic, ...);

Conclusions and future works

Conclusions

- ATL_{sc} is a very expressive, yet decidable extension of ATL;
- QCTL is a powerful extension of CTL;
- it is a nice tool to understand temporal logics for games (ATL_{sc}, Strategy Logic, ...);

Future directions

- Defining interesting fragments of those logics;
- Obtaining practicable algorithms.
- Considering partial observation;
- Considering randomised strategies.