Cycle Detection in Computation-Tree Logic

Gaelle Fontaine[1] Fabio Mogavero[2] Aniello Murano[3] **Giuseppe Perelli**[2]
Loredana Sorrentino[3]

[1]Universidad de Chile, [2]University of Oxford, [3]University of Naples

GandALF 2016
Catania, September 16th

Everybody knows Model Checking

How to check system correctness.

- System represented as Mathematical Structure $\mathcal{K}$ (e.g., Kripke structure, Labeled transition system);
- Desired behavior represented as Logic Formula φ (e.g., Modal Logic, LTL, CTL, CTL*);
- The systems meets the behavior if (and only if) $\mathcal{K} \models \varphi$.

Everybody knows Model Checking

How to check system correctness.

- System represented as Mathematical Structure $\mathcal{K}$ (e.g., Kripke structure, Labeled transition system);
- Desired behavior represented as Logic Formula φ (e.g., Modal Logic, LTL, CTL, CTL*);
- The systems meets the behavior if (and only if) $\mathcal{K} \models \varphi$.

The ability of checking a behaviour depends on the expressiveness of the logics.
Solution techniques often reduce to find cycle properties (e.g., lasso paths).

Everybody knows Model Checking

How to check system correctness.

- System represented as Mathematical Structure $\mathcal{K}$ (e.g., Kripke structure, Labeled transition system);
- Desired behavior represented as Logic Formula φ (e.g., Modal Logic, LTL, CTL, CTL*);
- The systems meets the behavior if (and only if) $\mathcal{K} \models \varphi$.

The ability of checking a behaviour depends on the expressiveness of the logics.
Solution techniques often reduce to find cycle properties (e.g., lasso paths).

Temporal logics usually don't have power to explicitly denote cyclic behaviour.

Everybody knows Model Checking

How to check system correctness.

- System represented as Mathematical Structure $\mathcal{K}$ (e.g., Kripke structure, Labeled transition system);
- Desired behavior represented as Logic Formula φ (e.g., Modal Logic, LTL, CTL, CTL*);
- The systems meets the behavior if (and only if) $\mathcal{K} \models \varphi$.

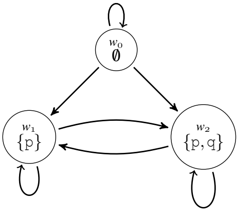
The ability of checking a behaviour depends on the expressiveness of the logics. Solution techniques often reduce to find cycle properties (e.g., lasso paths).

Temporal logics usually don't have power to explicitly denote cyclic behaviour.

Here: we introduce and investigate a temporal logic that can explicitly express the existence (or non existence!) of a cycle.

Kripke structures

A Kripke structure is a tuple $\mathcal{K} = \langle \mathrm{AP}, \mathrm{W}, \mathrm{R}, \mathsf{L}, w_I \rangle$ with:



- $\mathrm{AP} = \{\mathtt{p}, \mathtt{q}\}$;
- $\mathrm{W} = \{w_0, w_1, w_2\}$;
- $\mathrm{R} \subseteq \mathrm{W} \times \mathrm{W}$;
- $\mathsf{L} : \mathrm{W} \to 2^{\mathrm{AP}}$;
- $w_I \in \mathrm{W}$.

Basic notions

Path

A path in $\mathcal{K}$ is an infinite sequence $w_0, w_1, \ldots$ such that $(w_i, w_{i+1}) \in \mathrm{R}$, for all $i \in \mathbb{N}$.

Basic notions

Path

A path in $\mathcal{K}$ is an infinite sequence $w_0, w_1, \ldots$ such that $(w_i, w_{i+1}) \in \mathrm{R}$, for all $i \in \mathbb{N}$.

Cycle

A cycle in $\mathcal{K}$ is a path $w_0, w_1, \ldots$ such that w_0 occurs infinitely many times.

Outline

Outline

1. Computation-Tree Logic with Cycle Detection
2. Model-theoretic properties
3. Decision problems
4. Conclusions and future works

Computation-Tree Logic with Cycle Detection

Syntax

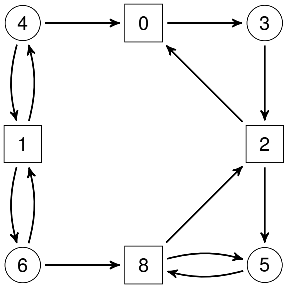
$\phi := p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathrm{E}\psi \mid \mathrm{A}\psi \mid \mathrm{E}^{\circlearrowleft}\psi \mid \mathrm{A}^{\circlearrowleft}\psi$
$\psi := \phi \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \mathrm{X}\psi \mid \psi \mathrm{U} \psi$

Two new path quantifiers $\mathrm{E}^{\circlearrowleft}$ and $\mathrm{A}^{\circlearrowleft}$, predicating over cycles.

Semantics

- $\mathcal{K}, w \models \mathrm{E}^{\circlearrowleft}\psi$ if there exists a cycle π starting from w such that $\mathcal{K}, \pi \models \psi$;
- $\mathcal{K}, w \models \mathrm{A}^{\circlearrowleft}\psi$ if, for all cycles π starting from w, it holds that $\mathcal{K}, \pi \models \psi$.

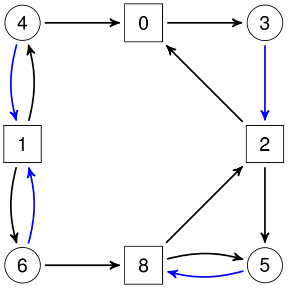
Example (Parity Games)



- V_0: nodes of player 0;
- V_1: nodes of player 1;
- $E \subseteq (V_0 \cup V_1) \times (V_0 \cup V_1)$;
- $p : (V_0 \cup V_1) \to \mathbb{N}$;
- An outcome of the game is a (infinite) path π determined by moves of the players in their respective nodes.

π is winning for Player 0 if the biggest priority occurring infinitely often is even.
π is prompt winning for Player 0 if every occurrence of an infinitely occurent odd number k is followed by a bigger even number n with bounded delay.

Example (Parity Games)

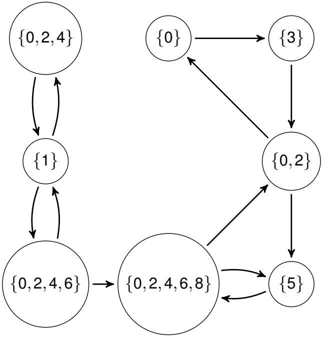


- V_0: nodes of player 0;
- V_1: nodes of player 1;
- $E \subseteq (V_0 \cup V_1) \times (V_0 \cup V_1)$;
- $p : (V_0 \cup V_1) \to \mathbb{N}$;
- An outcome of the game is a (infinite) path π determined by moves of the players in their respective nodes.

π is winning for Player 0 if the biggest priority occurring infinitely often is even.
π is prompt winning for Player 0 if every occurrence of an infinitely occurent odd number k is followed by a bigger even number n with bounded delay.

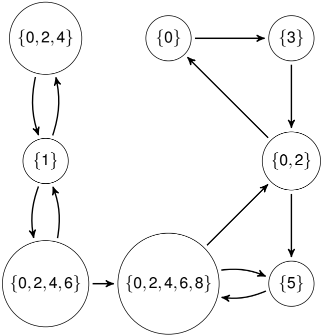
A strategy for Player 0 is a function mapping each node in V_0 to an outgoing arrow.
A strategy is winning if it enforces only winning paths.
A strategy is prompt winning if it enforces only prompt winning paths.

Strategy projection



- Strategy projection for Player 0 produces a Kripke structure;
- The paths in this structure are the ones enforced by Player 0;
- We can use $\mathrm{CTL}^{\star}$ and $\mathrm{CTL}^{\star}_{\circlearrowright}$ to represent properties of the strategy in the game.

Strategy projection

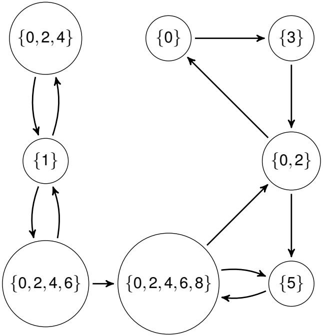


- Strategy projection for Player 0 produces a Kripke structure;
- The paths in this structure are the ones enforced by Player 0;
- We can use CTL* and CTL$^\star_\circlearrowleft$ to represent properties of the strategy in the game.

Winning strategy

$$\varphi^{par} = \mathtt{A}\bigwedge_{k\equiv_2 1}(\mathtt{GF}k \rightarrow \mathtt{GF}(k+1))$$

Strategy projection



- Strategy projection for Player 0 produces a Kripke structure;
- The paths in this structure are the ones enforced by Player 0;
- We can use CTL* and CTL$^\star_\circlearrowleft$ to represent properties of the strategy in the game.

Winning strategy

$$\varphi^{par} = \mathtt{A} \bigwedge_{k \equiv_2 1} (\mathtt{GF}\, k \rightarrow \mathtt{GF}(k+1))$$

Prompt winning strategy

$$\varphi^{par} \wedge \neg (\bigvee_{n \equiv_2 0} \mathtt{E} (\bigvee_{k<n, k \equiv_2 1} (\mathtt{GF}\, k \wedge \mathtt{G}(k \rightarrow \mathtt{G}\neg(K+1) \mathtt{U} \mathtt{E}^{\circlearrowleft} \mathtt{G} \neg (k+1)))))$$

Outline

Bisimulation?

Bisimulation?

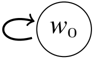


Figure: A Kripke structure $\mathcal{K}$

Bisimulation?

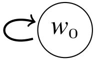


Figure: A Kripke structure $\mathcal{K}$

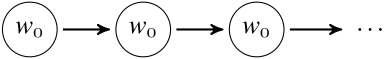


Figure: Tree unwinding $\mathcal{T}_{\mathcal{K}}$

Bisimulation?

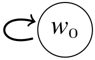


Figure: A Kripke structure $\mathcal{K}$

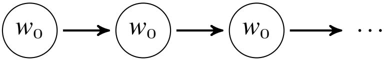


Figure: Tree unwinding $\mathcal{T}_{\mathcal{K}}$

$\mathcal{K}, w_0 \models \mathtt{E}^{\circlearrowleft}\top$
$\mathcal{T}_{\mathcal{K}}, w_0 \not\models \mathtt{E}^{\circlearrowleft}\top$

Bisimulation?

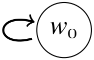


Figure: A Kripke structure $\mathcal{K}$

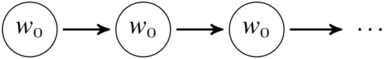


Figure: Tree unwinding $\mathcal{T}_{\mathcal{K}}$

$\mathcal{K}, w_0 \models \mathrm{E}^{\circlearrowright}\top$
$\mathcal{T}_{\mathcal{K}}, w_0 \not\models \mathrm{E}^{\circlearrowright}\top$

Theorem

- *$CTL^{\star}_{\circlearrowright}$ is not invariant under tree-unwinding*
- *$CTL^{\star}_{\circlearrowright}$ is not invariant under bisimulation*

Finite-Model Property?

Finite-Model Property?

$$\mathsf{AG}\neg\mathsf{E}^{\circlearrowleft}\top$$

For every path, at every time-point, there not exist a cycle.

Finite-Model Property?

$$\mathsf{AG}\neg\mathsf{E}^{\circlearrowleft}\top$$

For every path, at every time-point, there not exist a cycle.
There is no loop for every reachable state $\Rightarrow$ the formula is satisfiable only on infinite-state Kripke structures.

Finite-Model Property?

$$\mathtt{AG}\neg\mathtt{E}^{\circlearrowleft}\top$$

For every path, at every time-point, there not exist a cycle.
There is no loop for every reachable state $\Rightarrow$ the formula is satisfiable only on infinite-state Kripke structures.

Theorem

$CTL^{\star}_{\circlearrowleft}$ does not have the finite-model property.

Expressiveness

Theorem

$CTL^{\star}_{\circlearrowleft}$ is strictly more expressive than $CTL^{\star}$.

Expressiveness

Theorem

$CTL^\star_{\circlearrowleft}$ is strictly more expressive than $CTL^\star$.

Proof intuition

$CTL^\star$ is invariant under bisimulation.

Expressiveness

Theorem

$CTL^{\star}_{\circlearrowright}$ is strictly more expressive than $CTL^{\star}$.

Proof intuition

$\text{CTL}^{\star}$ is invariant under bisimulation.

Theorem

$CTL^{\star}_{\circlearrowright}$ is expressively incomparable with μCalculus .

Expressiveness

Theorem

$CTL^{\star}_{\circlearrowleft}$ is strictly more expressive than $CTL^{\star}$.

Proof intuition

CTL* is invariant under bisimulation.

Theorem

$CTL^{\star}_{\circlearrowleft}$ is expressively incomparable with μCalculus .

Proof intuition

μCalculus is invariant under bisimulation, but CTL$^{\star}_{\circlearrowleft}$ cannot express all regular expressions, e.g., the *counting-by-two*.

Cycle-Bisimulation

Definition

Cycle-Bisimulation

Two Kripke structures $\mathcal{K}_1$ and $\mathcal{K}_2$ are cycle-bisimilar if there exists a bisimulation relation $\mathrm{B} \subseteq \mathrm{W}_{\mathcal{K}_1} \times \mathrm{W}_{\mathcal{K}_2}$ on their states such that, for all $(w_1, w_2) \in \mathrm{B}$:

- For every cycle π_1 starting from w_1 there exists a cycle π_2 starting from w_2 bisimilar to π_1 *state-by-state*;
- For every cycle π_2 starting from w_2 there exists a cycle π_1 starting from w_1 bisimilar to π_2 *state-by-state*.

Cycle-Bisimulation

Definition

Cycle-Bisimulation

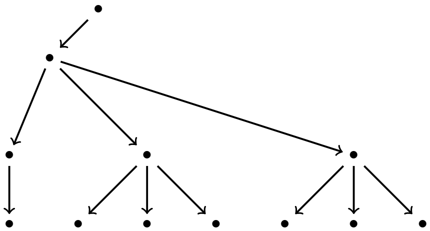
Two Kripke structures $\mathcal{K}_1$ and $\mathcal{K}_2$ are cycle-bisimilar if there exists a bisimulation relation $\mathrm{B} \subseteq \mathrm{W}_{\mathcal{K}_1} \times \mathrm{W}_{\mathcal{K}_2}$ on their states such that, for all $(w_1, w_2) \in \mathrm{B}$:

- For every cycle π_1 starting from w_1 there exists a cycle π_2 starting from w_2 bisimilar to π_1 *state-by-state*;
- For every cycle π_2 starting from w_2 there exists a cycle π_1 starting from w_1 bisimilar to π_2 *state-by-state*.

Theorem

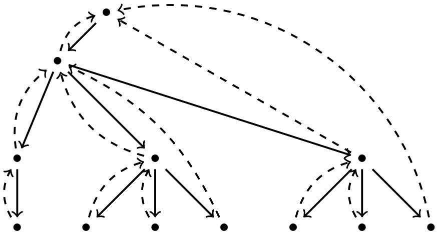
$CTL^{\star}_{\circlearrowleft}$ is invariant under cycle-bisimulation.

Tree with Back Edges



- A simple tree $\mathcal{T} = (\mathrm{V}, \mathsf{E})$

Tree with Back Edges



- A simple tree $\mathcal{T} = (\mathrm{V}, \mathsf{E})$
- Plus a partial mapping f from nodes to *back edges* such that:
  - $\mathsf{f}(w)$ is an ancestor of w;
  - Back edges do not overlap.

Tree-like unwinding

Theorem

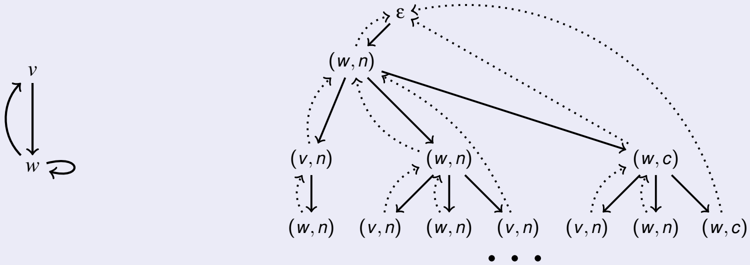
For every Kripke structure $\mathcal{K}$, there exists a tree with back edges $\mathcal{T}_{\mathcal{K}}$ which is cycle-bisimilar.

Tree-like unwinding

Theorem

For every Kripke structure $\mathcal{K}$, there exists a tree with back edges $\mathcal{T}_{\mathcal{K}}$ which is cycle-bisimilar.

An example



Outline

Model Checking

Theorem

The model-checking problem for $\mathrm{CTL}^{\star}_{\circlearrowright}$ *is PSpace-Complete w.r.t. the formula and NLogSpace-Complete w.r.t. the data complexity.*

Model Checking

Theorem

The model-checking problem for $\mathrm{CTL}^{\star}_{\circlearrowleft}$ is PSpace-Complete w.r.t. the formula and NLogSpace-Complete w.r.t. the data complexity.

Proof idea

Bottom-up automata-theoretic technique construction borrowed from $\mathrm{CTL}^{\star}$.
The cases $\mathtt{E}^{\circlearrowleft}\psi$ and $\mathtt{A}^{\circlearrowleft}\psi$ are similar to $\mathtt{E}\psi$ and $\mathtt{A}\psi$ with an additional check that the initial state occurs infinitely often. It suffices to intersect the automaton $\mathcal{N}_{\mathcal{K}}^{\mathtt{E}\psi}$ with a suitably defined Büchi word automaton for this additional check.

Satisfiability

Problem

The CTL* automata-technique cannot be borrowed as there is no tree-model property.

Satisfiability

Problem

The CTL* automata-technique cannot be borrowed as there is no tree-model property.

However

Exploiting the tree-like model property and cycle-bisimulation invariance we can build a two-way parity tree automaton $\mathcal{A}_\varphi$ that:

- Is of size double-exponential w.r.t. φ;

Satisfiability

Problem

The CTL* automata-technique cannot be borrowed as there is no tree-model property.

However

Exploiting the tree-like model property and cycle-bisimulation invariance we can build a two-way parity tree automaton $\mathcal{A}_\varphi$ that:

- Is of size double-exponential w.r.t. φ;
- Recognises all the tree structures that can be extended into the tree-like models of φ;

Satisfiability

Problem

The CTL* automata-technique cannot be borrowed as there is no tree-model property.

However

Exploiting the tree-like model property and cycle-bisimulation invariance we can build a two-way parity tree automaton $\mathcal{A}_\varphi$ that:

- Is of size double-exponential w.r.t. φ;
- Recognises all the tree structures that can be extended into the tree-like models of φ;
- The back-and-forth reading of the tree is used to correctly guess the back edge of the model.

Satisfiability

Problem

The CTL* automata-technique cannot be borrowed as there is no tree-model property.

However

Exploiting the tree-like model property and cycle-bisimulation invariance we can build a two-way parity tree automaton $\mathcal{A}_\varphi$ that:

- Is of size double-exponential w.r.t. φ;
- Recognises all the tree structures that can be extended into the tree-like models of φ;
- The back-and-forth reading of the tree is used to correctly guess the back edge of the model.

Satisfiability

Problem

The CTL* automata-technique cannot be borrowed as there is no tree-model property.

However

Exploiting the tree-like model property and cycle-bisimulation invariance we can build a two-way parity tree automaton $\mathcal{A}_\varphi$ that:

- Is of size double-exponential w.r.t. φ;
- Recognises all the tree structures that can be extended into the tree-like models of φ;
- The back-and-forth reading of the tree is used to correctly guess the back edge of the model.

Theorem

The satisfiability problem for CTL$^\star_\circlearrowleft$ can be solved in 3ExpTime and it is 2ExpTime-Hard.

Outline

Conclusion

In this paper we ...

- Introduced $\text{CTL}^{\star}_{\circlearrowleft}$ to explicitly quantify over cycles in structures;

Conclusion

In this paper we ...

- Introduced $\mathrm{CTL}^{\star}_{\circlearrowleft}$ to explicitly quantify over cycles in structures;
- Compared its expressive power with well known formalisms like $\mathrm{CTL}^{\star}$ and μCalculus ;

Conclusion

In this paper we ...

- Introduced $\mathrm{CTL}^{\star}_{\circlearrowleft}$ to explicitly quantify over cycles in structures;
- Compared its expressive power with well known formalisms like $\mathrm{CTL}^{\star}$ and μCalculus ;
- Investigated on its model properties like invariance under bisimulation;

Conclusion

In this paper we ...

- Introduced $\mathrm{CTL}^{\star}_{\circlearrowleft}$ to explicitly quantify over cycles in structures;
- Compared its expressive power with well known formalisms like $\mathrm{CTL}^{\star}$ and μCalculus ;
- Investigated on its model properties like invariance under bisimulation;
- Addressed both Model-Checking and Satisfiability problems.

Future works

- Fill the gap in complexity for the satisfiability problem;
- Further syntactic extension in which the cycle symbol is atomic (work in progress);
- Quantifying over cycles in logics for open systems (e.g., $\mathsf{ATL}^{\star}$, and SL).

Thank you!