Regular Sensing

Shaull Almagor[1] Denis Kuperberg[2]
Orna Kupferman[1] Yaron Velner[1]

[1]The Hebrew University, Jerusalem, Israel.

[2]IRIT/Onera, Toulouse

FSTTCS 2015+Submitted Work

Sensing - Motivation

- [redacted]
- [redacted]
- [redacted]
- [redacted]

Sensing - Motivation

- [redacted] si [redacted] ha [redacted] ts I [redacted] puts, resp [redacted]
- Con [redacted] ular I/O sp [redacted] $2^I \times 2^O$ [redacted] corresp [redacted] $\mathcal{T}$.
- [redacted] ading t [redacted] f each s [redacted] nsor.
- The [redacted] exity of $\mathcal{T}$ c [redacted] ber of se [redacted] s.

Sensing - Motivation

- Co[redacted]sider an al[redacted]habet $2^I \times 2^O$ [redacted]r some finite sets I an[redacted] of input[redacted]tputs, respect[redacted]
- Con[redacted]er a regular I/O spe[redacted]ication over $2^I \times 2^O$, an[redacted] corresp[redacted]ing transducer $\mathcal{T}$.
- [redacted]ading t[redacted] truth value of each s[redacted]nal in I requires activating [redacted]e sensor.
- The comp[redacted]exity of $\mathcal{T}$ can be measu[redacted]he expected n[redacted]ber of sensors $\mathcal{T}$ us[redacted]s.

Sensing - Motivation

- Consider an alphabet $2^I \times 2^O$ for some finite sets I and O of inputs and outputs, respectively.
- Consider a regular I/O specification over $2^I \times 2^O$, and a corresponding transducer $\mathcal{T}$.
- Reading the truth value of each signal in I requires activating some sensor.
- The complexity of $\mathcal{T}$ can be measured by the expected number of sensors $\mathcal{T}$ uses.

Related Work

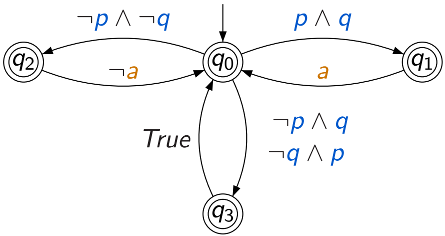
- Mean-payoff Games with Incomplete Information
  [P. Hunter, G. A. Pérez, and J.F. Raskin, 2013].
- Minimum attention controller synthesis for ω-regular objectives
  [K. Chatterjee and R. Majumdar, 2011].
- Controller synthesis with budget constraints
  [K. Chatterjee, R. Majumdar, and T. A. Henzinger, 2008].
- Synthesis with Incomplete Information
  [O. Kupferman and M.Y. Vardi, 97].

Example (1)

Let $I = \{p, q\}$ and $O = \{a\}$. Consider the specification:

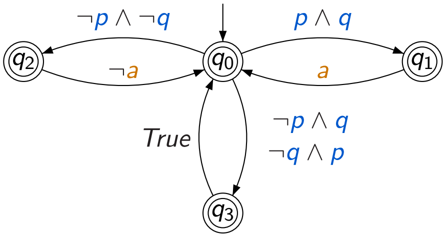
Example (1)

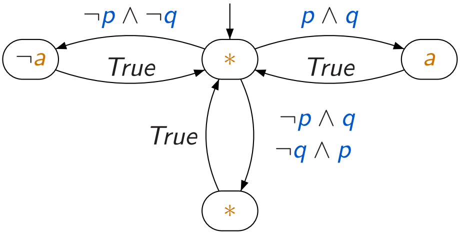
Let $I = \{p, q\}$ and $O = \{a\}$. Consider the specification:



Example (1)

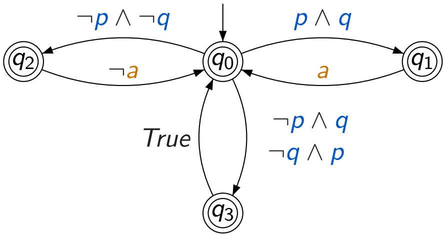
Let $I = \{p, q\}$ and $O = \{a\}$. Consider the specification:

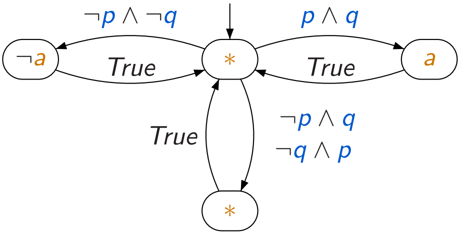




Example (1)

Let $I = \{p, q\}$ and $O = \{a\}$. Consider the specification:

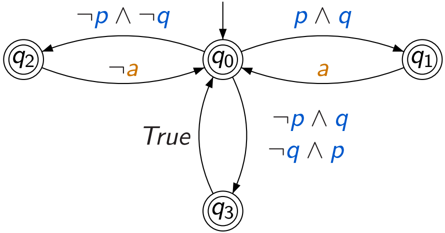


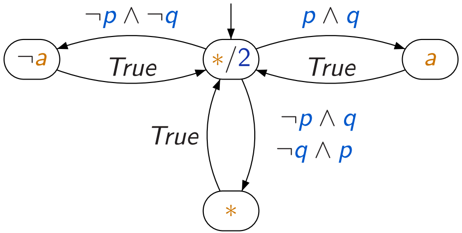


- Initial state senses $\{p, q\}$.

Example (1)

Let $I = \{p, q\}$ and $O = \{a\}$. Consider the specification:

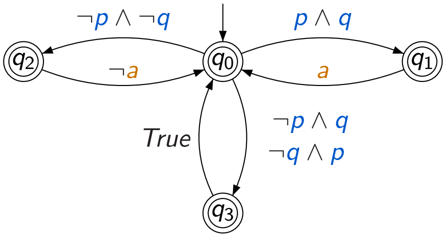


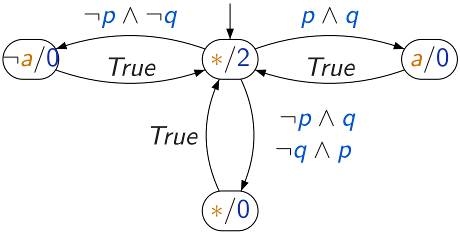


- Initial state senses $\{p, q\}$.
- Has sensing cost 2.

Example (1)

Let $I = \{p, q\}$ and $O = \{a\}$. Consider the specification:

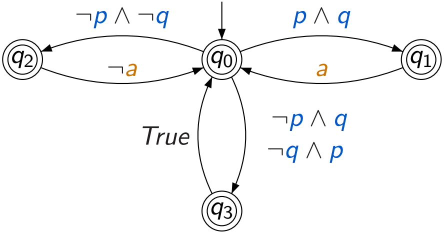


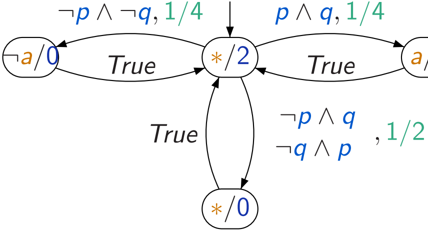


- Initial state senses $\{p, q\}$.
- Has sensing cost 2.
- Other states sense $\emptyset$.

Example (1)

Let $I = \{p, q\}$ and $O = \{a\}$. Consider the specification:





- Initial state senses $\{p, q\}$.
- Has sensing cost 2.
- Other states sense $\emptyset$.
- Sensing cost of $\mathcal{T}$ is 1.

Sensing - Definition

- Consider an I/O-*transducer* $\mathcal{T}$.

Sensing - Definition

- Consider an I/O-*transducer* $\mathcal{T}$.
- A state q of $\mathcal{T}$ is said to sense a signal $p \in I$ if there exists $i \subseteq I$ such that $\delta(q, i) \neq \delta(q, i \cup \{p\})$.

Sensing - Definition

- Consider an I/O-*transducer* $\mathcal{T}$.
- A state q of $\mathcal{T}$ is said to sense a signal $p \in I$ if there exists $i \subseteq I$ such that $\delta(q, i) \neq \delta(q, i \cup \{p\})$.
- The sensing cost of a state q is $\mathit{sensed}(q) = |\{p : q \text{ senses } p\}|$.

Sensing - Definition

- Consider an I/O-*transducer* $\mathcal{T}$.
- A state q of $\mathcal{T}$ is said to sense a signal $p \in I$ if there exists $i \subseteq I$ such that $\delta(q, i) \neq \delta(q, i \cup \{p\})$.
- The sensing cost of a state q is $sensed(q) = |\{p : q \text{ senses } p\}|$.
- The sensing cost of a word $\pi \in (2^I)^\omega$ is the average sensing cost along the run of $\mathcal{T}$ on π.

Sensing - Definition

- Consider an I/O-*transducer* $\mathcal{T}$.
- A state q of $\mathcal{T}$ is said to sense a signal $p \in I$ if there exists $i \subseteq I$ such that $\delta(q, i) \neq \delta(q, i \cup \{p\})$.
- The sensing cost of a state q is $sensed(q) = |\{p : q \text{ senses } p\}|$.
- The sensing cost of a word $\pi \in (2^I)^\omega$ is the average sensing cost along the run of $\mathcal{T}$ on π.
- The sensing cost of $\mathcal{T}$ is the expected average cost on a uniformly-random word.

Problem Formulation

- We are given a specification-automaton $\mathcal{A}$ over alphabet $2^{I} \times 2^{O}$.

Problem Formulation

- We are given a specification-automaton $\mathcal{A}$ over alphabet $2^{I} \times 2^{O}$.
- Output an I/O-*transducer* $\mathcal{T}$ that realizes $\mathcal{A}$, with minimal sensing.

Example (2)

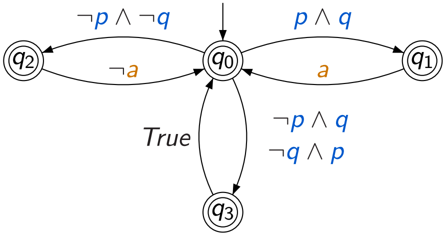
- It is well known that a DFA specification is realizable iff there is an "embodied" strategy.

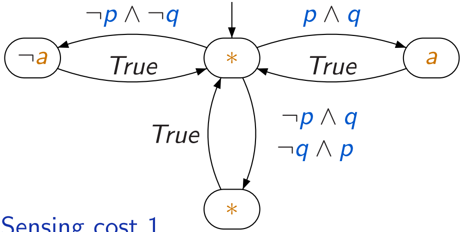
Example (2)

- It is well known that a DFA specification is realizable iff there is an "embodied" strategy.
- However, a minimal-sensing transducer does not always correspond to an embodied strategy.

Example (2)

Let $I = \{p, q\}$ and $O = \{a\}$. Consider the specification:

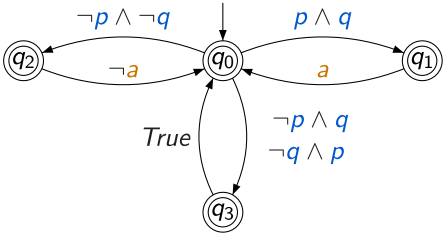


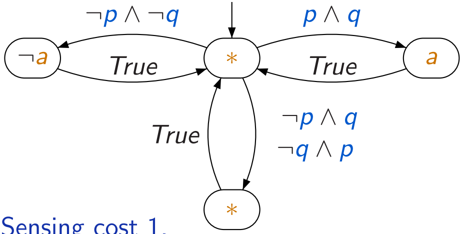


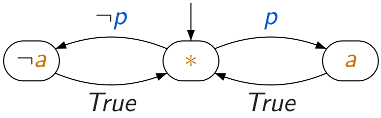
Sensing cost 1.

Example (2)

Let $I = \{p, q\}$ and $O = \{a\}$. Consider the specification:







Sensing cost 1.

Sensing cost $\frac{1}{2}$.

Example (2)

- It is well known that a DFA specification is realizable iff there is an "embodied" strategy.
- However, a minimal-sensing transducer does not always correspond to an embodied strategy.
- For infinite words, the minimal sensing might not be attained. For example: $\mathsf{GF}(a \iff \mathsf{X}b)$

Lower Bound

Theorem

Given a DFA specification $\mathcal{A}$ and a threshold v, deciding whether $scost(\mathcal{A}) \leq v$ is EXPTIME-hard.

This lower bound carries to safety properties on infinite words, and hence to all acceptance conditions.

Lower Bound

Theorem

Given a DFA specification $\mathcal{A}$ and a threshold v, deciding whether $scost(\mathcal{A}) \leq v$ is EXPTIME-hard.

This lower bound carries to safety properties on infinite words, and hence to all acceptance conditions.

We show a reduction from the problem of deciding the emptiness of the intersection of DFTs (EXPTIME-complete).

Lower Bound

Theorem

Given a DFA specification $\mathcal{A}$ and a threshold v, deciding whether $scost(\mathcal{A}) \leq v$ is EXPTIME-hard.

This lower bound carries to safety properties on infinite words, and hence to all acceptance conditions.

We show a reduction from the problem of deciding the emptiness of the intersection of DFTs (EXPTIME-complete).

We demonstrate the idea with DFAs.

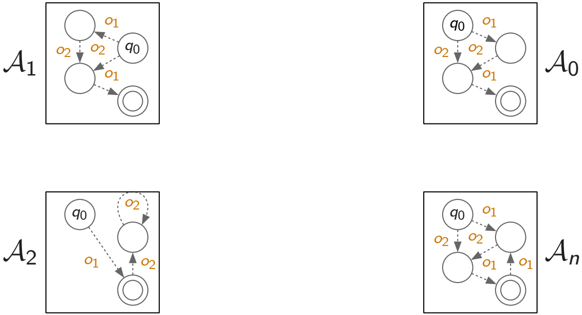
Lower Bound - Proof

Consider DFAs $\mathcal{A}_0, ..., \mathcal{A}_n$. We construct a specification:

Lower Bound - Proof

Consider DFAs $\mathcal{A}_0, \ldots, \mathcal{A}_n$. We construct a specification:

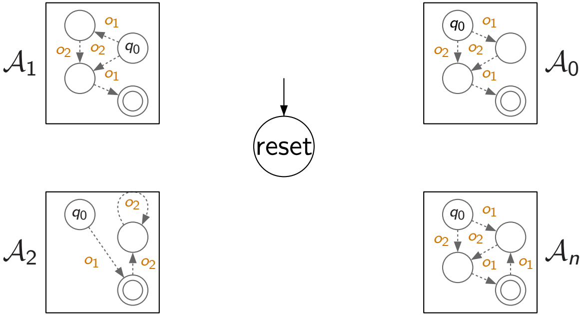
- The DFAs' alphabet are outputs.
-
-
-
-



Lower Bound - Proof

Consider DFAs $\mathcal{A}_0, \ldots, \mathcal{A}_n$. We construct a specification:

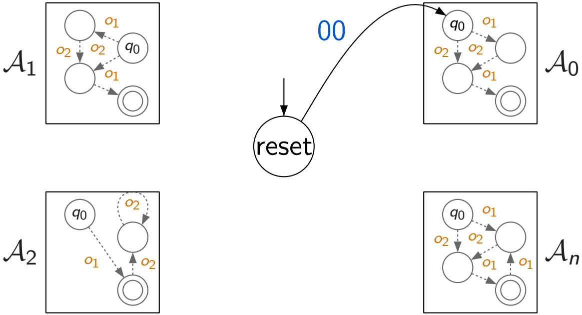
- The DFAs' alphabet are outputs.
- In *reset*, the input determines which DFA we choose.
-
-
-



Lower Bound - Proof

Consider DFAs $\mathcal{A}_0, \ldots, \mathcal{A}_n$. We construct a specification:

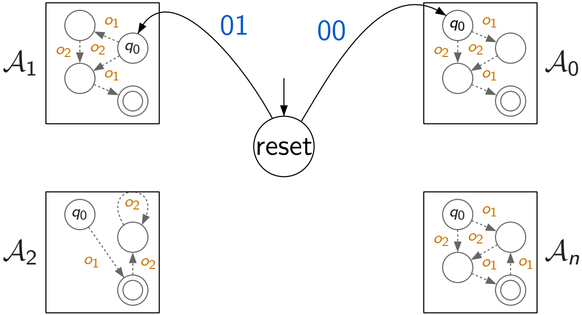
- The DFAs' alphabet are outputs.
- In *reset*, the input determines which DFA we choose.
-
-
-



Lower Bound - Proof

Consider DFAs $\mathcal{A}_0, ..., \mathcal{A}_n$. We construct a specification:

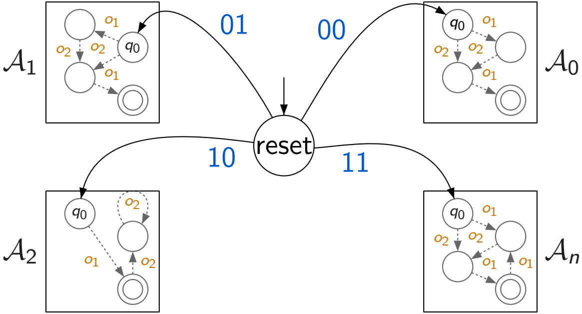
- The DFAs' alphabet are outputs.
- In *reset*, the input determines which DFA we choose.
-
-
-



Lower Bound - Proof

Consider DFAs $\mathcal{A}_0, \ldots, \mathcal{A}_n$. We construct a specification:

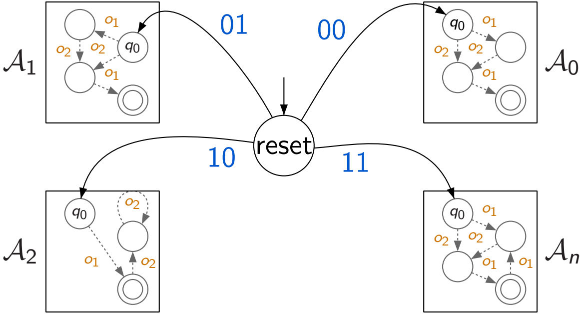
- The DFAs' alphabet are outputs.
- In *reset*, the input determines which DFA we choose.
-
-
-



Lower Bound - Proof

Consider DFAs $\mathcal{A}_0, \ldots, \mathcal{A}_n$. We construct a specification:

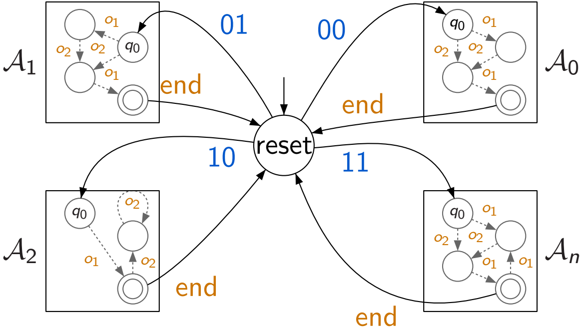
- The DFAs' alphabet are outputs.
- In *reset*, the input determines which DFA we choose.
- We force sensing of one bit in the DFAs.
-
-



Lower Bound - Proof

Consider DFAs $\mathcal{A}_0, \dots, \mathcal{A}_n$. We construct a specification:

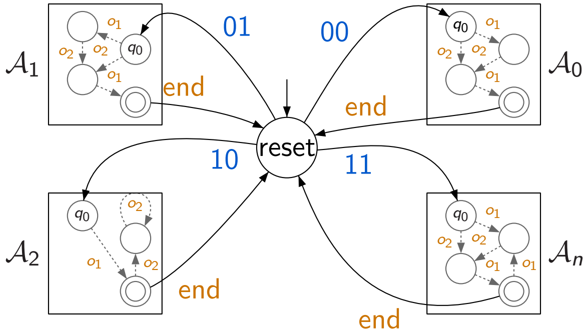
- The DFAs' alphabet are outputs.
- In *reset*, the input determines which DFA we choose.
- We force sensing of one bit in the DFAs.
- The output *end* signifies that the word is finished.
-



Lower Bound - Proof

Consider DFAs $\mathcal{A}_0, \ldots, \mathcal{A}_n$. We construct a specification:

- The DFAs' alphabet are outputs.
- In *reset*, the input determines which DFA we choose.
- We force sensing of one bit in the DFAs.
- The output *end* signifies that the word is finished.
- A transducer can ignore inputs in *reset* iff $\bigcap_i L(\mathcal{A}_i) \neq \emptyset$.



Upper Bound - Questions

- Given a parity automaton $\mathcal{A}$ over alphabet $2^{I} \times 2^{O}$, can we compute its infimum sensing cost?

Upper Bound - Questions

- Given a parity automaton $\mathcal{A}$ over alphabet $2^{I} \times 2^{O}$, can we compute its infimum sensing cost?
- Can we also output a transducer that attains/approximates this value?

Upper Bound - Questions

- Given a parity automaton $\mathcal{A}$ over alphabet $2^{I} \times 2^{O}$, can we compute its infimum sensing cost?
- Can we also output a transducer that attains/approximates this value?
- Do *finite* transducers suffice for an approximation?

From Sensing to Parity-MDPs

- Traditional synthesis from a parity automaton is solved by translating it to a parity game, and looking for a winning strategy.

From Sensing to Parity-MDPs

- Traditional synthesis from a parity automaton is solved by translating it to a parity game, and looking for a winning strategy.
- We reduce our problem to a variant of parity games combined with an MDP.

From Sensing to Parity-MDPs

Given a parity automaton $\mathcal{A}$ over alphabet $2^{I} \times 2^{O}$:

1. Construct a universal parity automaton $\mathcal{A}'$ where each state record the current state of $\mathcal{A}$ and which inputs are sensed. Then, given concrete inputs, universally choose an successor that agrees with the concrete inputs on the sensed inputs.

From Sensing to Parity-MDPs

Given a parity automaton $\mathcal{A}$ over alphabet $2^{I} \times 2^{O}$:

1. Construct a universal parity automaton $\mathcal{A}'$ where each state record the current state of $\mathcal{A}$ and which inputs are sensed. Then, given concrete inputs, universally choose an successor that agrees with the concrete inputs on the sensed inputs.
2. Determinize $\mathcal{A}'$ to a parity automaton $\mathcal{D}$.

From Sensing to Parity-MDPs

Given a parity automaton $\mathcal{A}$ over alphabet $2^{I} \times 2^{O}$:

1. Construct a universal parity automaton $\mathcal{A}'$ where each state record the current state of $\mathcal{A}$ and which inputs are sensed. Then, given concrete inputs, universally choose an successor that agrees with the concrete inputs on the sensed inputs.
2. Determinize $\mathcal{A}'$ to a parity automaton $\mathcal{D}$.
3. Construct from $\mathcal{D}$ a parity game $\mathcal{G}$, and assign a cost to each state according to the number of sensed inputs.

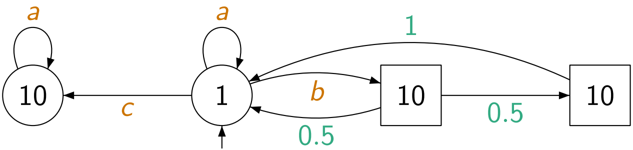
From Sensing to Parity-MDPs

Given a parity automaton $\mathcal{A}$ over alphabet $2^{I} \times 2^{O}$:

1. Construct a universal parity automaton $\mathcal{A}'$ where each state record the current state of $\mathcal{A}$ and which inputs are sensed. Then, given concrete inputs, universally choose an successor that agrees with the concrete inputs on the sensed inputs.
2. Determinize $\mathcal{A}'$ to a parity automaton $\mathcal{D}$.
3. Construct from $\mathcal{D}$ a parity game $\mathcal{G}$, and assign a cost to each state according to the number of sensed inputs.
4. A winning strategy in $\mathcal{G}$ realizes the specification, and its expected mean-payoff against a stochastic environment is its sensing cost.

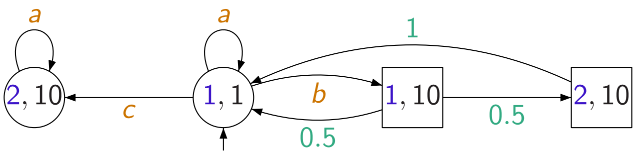
Parity MDPs

- A parity MDP is an MDP...



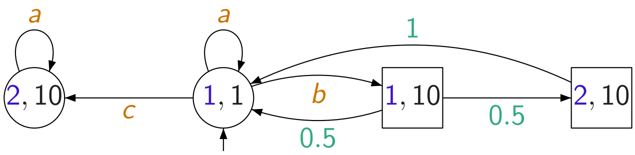
Parity MDPs

- A parity MDP is an MDP… equipped with a parity winning condition.



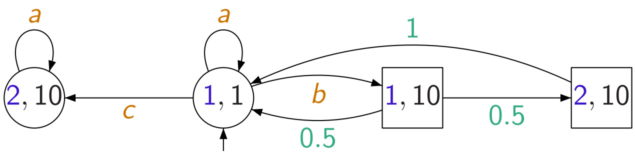
Parity MDPs

- A parity MDP is an MDP... equipped with a parity winning condition.
- The value of a sure-winning Player 1 strategy is the expected cost against the stochastic environment.



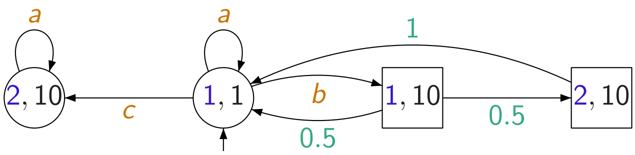
Parity MDPs

- A parity MDP is an MDP... equipped with a parity winning condition.
- The value of a sure-winning Player 1 strategy is the expected cost against the stochastic environment.
- Thus, Player 1 needs to surely win against an adversarial environment, while minimizing the expected cost.



Parity MDPs

- A parity MDP is an MDP... equipped with a parity winning condition.
- The value of a sure-winning Player 1 strategy is the expected cost against the stochastic environment.
- Thus, Player 1 needs to surely win against an adversarial environment, while minimizing the expected cost.
- Optimal strategy may not exist:



Solving Parity MDPs

- A *Good End Component* (GEC) in $\mathcal{G}$ is an end component whose maximal parity rank is even.

Solving Parity MDPs

- A *Good End Component* (GEC) in $\mathcal{G}$ is an end component whose maximal parity rank is even.
- Easy: every winning strategy reaches a GEC w.p. 1.

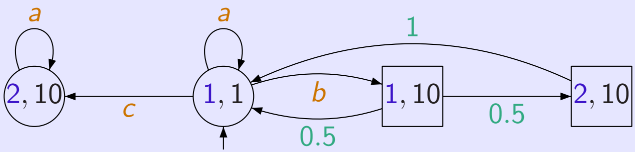
Solving Parity MDPs

- A *Good End Component* (GEC) in $\mathcal{G}$ is an end component whose maximal parity rank is even.
- Easy: every winning strategy reaches a GEC w.p. 1.
- We show that within a GEC, we can approximate the optimal mean-payoff with an infinite-memory strategy.

Solving Parity MDPs

- A *Good End Component* (GEC) in $\mathcal{G}$ is an end component whose maximal parity rank is even.
- Easy: every winning strategy reaches a GEC w.p. 1.
- We show that within a GEC, we can approximate the optimal mean-payoff with an infinite-memory strategy.

Proof Idea:

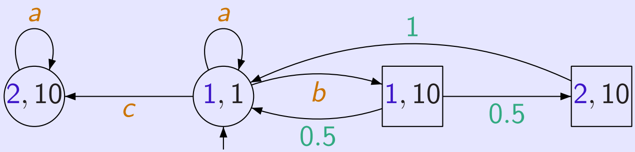


Solving Parity MDPs

- A *Good End Component* (GEC) in $\mathcal{G}$ is an end component whose maximal parity rank is even.
- Easy: every winning strategy reaches a GEC w.p. 1.
- We show that within a GEC, we can approximate the optimal mean-payoff with an infinite-memory strategy.

Proof Idea:

1. Play a for a long time.

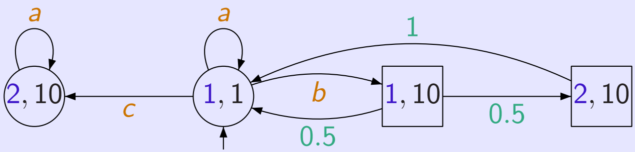


Solving Parity MDPs

- A *Good End Component* (GEC) in $\mathcal{G}$ is an end component whose maximal parity rank is even.
- Easy: every winning strategy reaches a GEC w.p. 1.
- We show that within a GEC, we can approximate the optimal mean-payoff with an infinite-memory strategy.

Proof Idea:

1. Play a for a long time.
2. Play b for a while, try to reach $2, 10$.

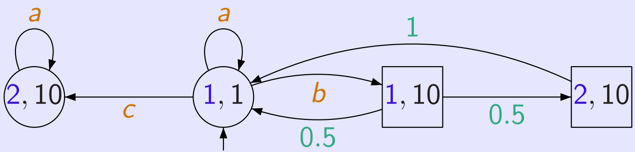


Solving Parity MDPs

- A *Good End Component* (GEC) in $\mathcal{G}$ is an end component whose maximal parity rank is even.
- Easy: every winning strategy reaches a GEC w.p. 1.
- We show that within a GEC, we can approximate the optimal mean-payoff with an infinite-memory strategy.

Proof Idea:

1. Play a for a long time.
2. Play b for a while, try to reach $2, 10$.
3. If $2, 10$ reached, goto 1 with a longer counter.

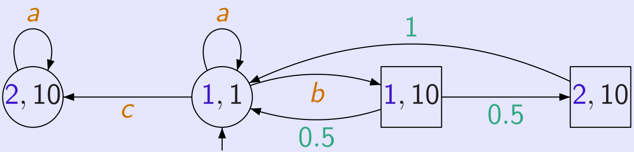


Solving Parity MDPs

- A *Good End Component* (GEC) in $\mathcal{G}$ is an end component whose maximal parity rank is even.
- Easy: every winning strategy reaches a GEC w.p. 1.
- We show that within a GEC, we can approximate the optimal mean-payoff with an infinite-memory strategy.

Proof Idea:

1. Play *a* for a long time.
2. Play *b* for a while, try to reach 2, 10.
3. If 2, 10 reached, goto 1 with a longer counter.
4. Otherwise, play *c* ("give up").



Summary of Results

- We show how to compute the value of Parity-MDPs in NP∩coNP.

Summary of Results

- We show how to compute the value of Parity-MDPs in NP∩coNP.
- Can also find an approximating infinite-memory strategy.

Summary of Results

- We show how to compute the value of Parity-MDPs in NP∩coNP.
- Can also find an approximating infinite-memory strategy.
- We show how to compute the value of Parity-MDPs under *finite-memory* strategies in NP∩coNP.

Summary of Results

- We show how to compute the value of Parity-MDPs in NP∩coNP.
- Can also find an approximating infinite-memory strategy.
- We show how to compute the value of Parity-MDPs under *finite-memory* strategies in NP∩coNP.
- Can also find an approximating finite-memory strategy for the latter.

Summary of Results

- We show how to compute the value of Parity-MDPs in NP∩coNP.
- Can also find an approximating infinite-memory strategy.
- We show how to compute the value of Parity-MDPs under *finite-memory* strategies in NP∩coNP.
- Can also find an approximating finite-memory strategy for the latter.
- Enables us to find a minimally-sensing transducer (or an approximating one) in EXPTIME, matching the lower bound.

Summary of Results

- We show how to compute the value of Parity-MDPs in NP∩coNP.
- Can also find an approximating infinite-memory strategy.
- We show how to compute the value of Parity-MDPs under *finite-memory* strategies in NP∩coNP.
- Can also find an approximating finite-memory strategy for the latter.
- Enables us to find a minimally-sensing transducer (or an approximating one) in EXPTIME, matching the lower bound.
- Parity MDPs are a useful tool in modeling combinations of quantitative and Boolean properties.

[redacted]k y[redacted]u!

Th[redacted]k y[redacted]u!