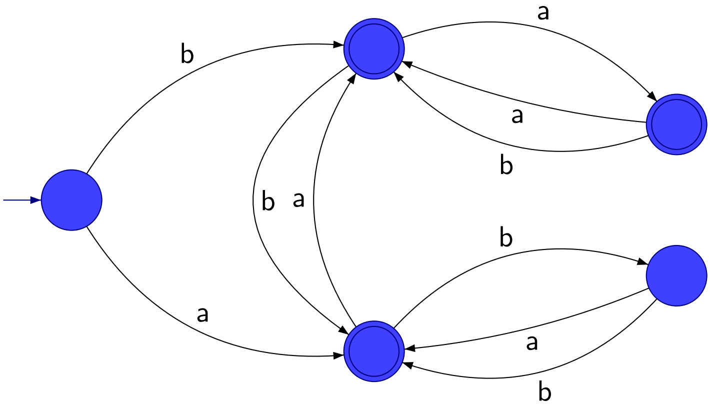
From deterministic finite automata to infinite games

Eryk Kopczyński

University of Warsaw

May 11, 2011

Deterministic finite automaton



used to recognize languages $L \subseteq \Sigma^*$, i.e., sets of words using letters from the set $\Sigma = \{a, b\}$

$$a \in L, ab \notin L, aba \in L, \ldots$$

Regular languages

Deterministic finite automaton = a simplest automaton which performs some computation

Regular languages = those that can be recognized by deterministic finite automata

Regular languages

Deterministic finite automaton = a simplest automaton which performs some computation
Regular languages = those that can be recognized by deterministic finite automata

Regular languages can be also defined

- algebraically (recognized by a finite monoid)

Regular languages

Deterministic finite automaton = a simplest automaton which performs some computation
Regular languages = those that can be recognized by deterministic finite automata

Regular languages can be also defined

- algebraically (recognized by a finite monoid)
- using regular expressions: regular languages can be obtained from very simple languages ($\emptyset$, a) using concatenation, union, or Kleene star

Regular languages

Deterministic finite automaton = a simplest automaton which performs some computation
Regular languages = those that can be recognized by deterministic finite automata

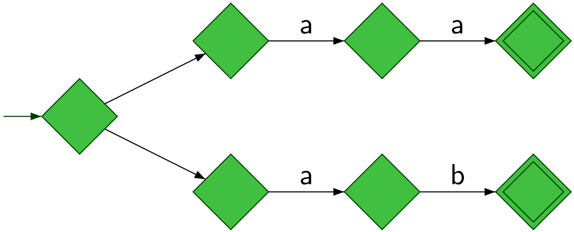
Regular languages can be also defined

- algebraically (recognized by a finite monoid)
- using regular expressions: regular languages can be obtained from very simple languages ($\emptyset$, a) using concatenation, union, or Kleene star
- ...

Non-determinism

In some states the automaton has a choice of where it will go

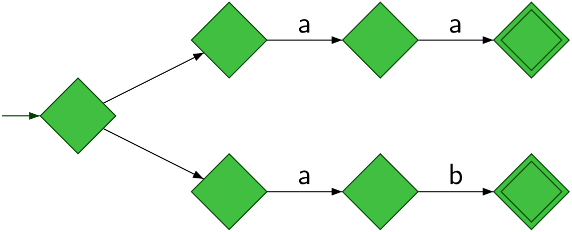
We assume that the automaton always makes the choice which leads to acceptance



Non-determinism

In some states the automaton has a choice of where it will go

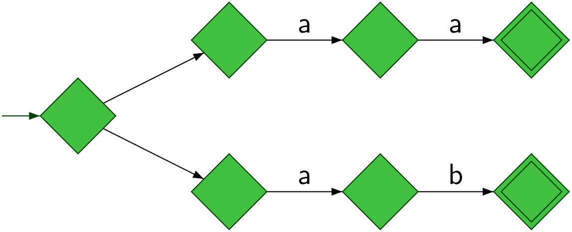
We assume that the automaton always makes the choice which leads to acceptance



Not very realistic, but theoretically very useful

Non-determinism

In some states the automaton has a choice of where it will go
We assume that the automaton always makes the choice which leads to acceptance



Not very realistic, but theoretically very useful
subset construction: n non-det states $\rightarrow 2^n$ det states

But what is non-determinism?

We can also define languages with logical formulae

- $\exists i \; i \in A$
  (there is a in the word: $\Sigma^* a \Sigma^*$)

But what is non-determinism?

We can also define languages with logical formulae

- $\exists i \; i \in A$
  (there is a in the word: $\Sigma^* a \Sigma^*$)
- $\exists i \; \exists j \; i \in A \wedge j \in B \wedge j > i$
  (there are letters a and b in the word, and b is after a)

But what is non-determinism?

We can also define languages with logical formulae

- $\exists i \; i \in A$
  (there is a in the word: $\Sigma^* a \Sigma^*$)
- $\exists i \; \exists j \; i \in A \wedge j \in B \wedge j > i$
  (there are letters a and b in the word, and b is after a)

In terms of logic, non-determinism corresponds to disjunction and existential quantification

In terms of regular expressions, non-determinism allows to express union, concatenation, and Kleene star easily

More about formulae

What languages can we express using first order logic (FO)?

$$\exists i \quad \forall i \quad i = j \quad i < j \quad i \in A \quad \neg \quad \vee \quad \wedge$$

More about formulae

What languages can we express using first order logic (FO)?

$$\exists i \quad \forall i \quad i = j \quad i < j \quad i \in A \quad \neg \quad \vee \quad \wedge$$

We can express for example $(oe)^*$

More about formulae

What languages can we express using first order logic (FO)?

$$\exists i \quad \forall i \quad i = j \quad i < j \quad i \in A \quad \neg \quad \vee \quad \wedge$$

We can express for example $(oe)^*$

Answer: starfree languages $\subsetneq$ regular languages

More about formulae

What languages can we express using first order logic (FO)?

$$\exists i \quad \forall i \quad i = j \quad i < j \quad i \in A \quad \neg \quad \vee \quad \wedge$$

We can express for example $(oe)^*$

Answer: starfree languages $\subsetneq$ regular languages

To get all regular languages, we also need $\exists O \exists E$ (MSO logic)

Non-deterministic automata: what about negation?

Non-deterministic automata can easily express $\exists$ and $\vee$.
But what about $\neg$, $\wedge$, and $\forall$?

Non-deterministic automata: what about negation?

Non-deterministic automata can easily express $\exists$ and $\vee$.
But what about $\neg$, $\wedge$, and $\forall$?

Impossible to do effectively: we have to determinize the automaton (thus 2^n states)

Non-deterministic automata: what about negation?

Non-deterministic automata can easily express $\exists$ and $\vee$.
But what about $\neg$, $\wedge$, and $\forall$?

Impossible to do effectively: we have to determinize the automaton (thus 2^n states)

Maybe we can do something to do it effectively?

Non-deterministic automata: what about negation?

Non-deterministic automata can easily express $\exists$ and $\vee$.
But what about $\neg$, $\wedge$, and $\forall$?

Impossible to do effectively: we have to determinize the automaton (thus 2^n states)

Maybe we can do something to do it effectively?

A try: **Accept all paths** – we can express $\wedge$, but not $\vee$

Solution: using games

The sequence (a_n) is convergent

$$\exists l\ \forall \epsilon\ \exists m\ \forall n\ (n < m \vee |a_n - l| < \epsilon)$$

Solution: using games

The sequence (a_n) is convergent

$$\exists l \; \forall \epsilon \; \exists m \; \forall n \; (n < m \vee |a_n - l| < \epsilon)$$

- Eva chooses l
- Adam chooses ϵ
- Eva chooses m
- Adam chooses n

Solution: using games

The sequence (a_n) is convergent

$$\exists l\ \forall \epsilon\ \exists m\ \forall n\ (n < m \vee |a_n - l| < \epsilon)$$

- Eva chooses l
- Adam chooses ϵ
- Eva chooses m
- Adam chooses n
- Eva chooses whether we test $n < m$ or $|a_n - l| < \epsilon$
- Eva wins if true, Adam wins if false

Solution: using games

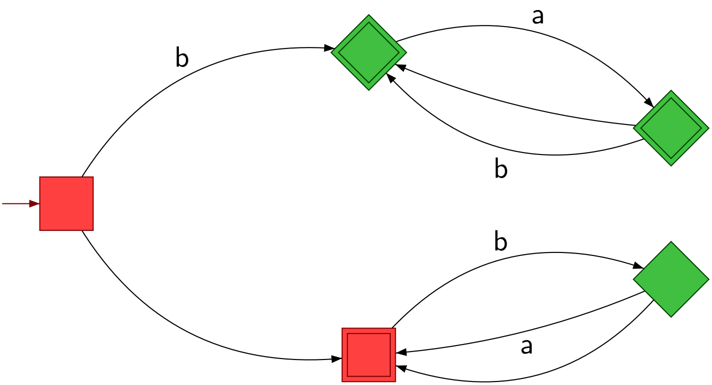
The sequence (a_n) is convergent

$$\exists l\ \forall \epsilon\ \exists m\ \forall n\ (n < m \vee |a_n - l| < \epsilon)$$

- Eva chooses l
- Adam chooses ϵ
- Eva chooses m
- Adam chooses n
- Eva chooses whether we test $n < m$ or $|a_n - l| < \epsilon$
- Eva wins if true, Adam wins if false

If both players play perfectly and Eva wins, then the sequence is convergent

Alternating automata



Alternation in complexity theory

What problems can be solved by a machine running in polynomial time?

deterministic	P

Alternation in complexity theory

What problems can be solved by a machine running in polynomial time?

deterministic	P
non-deterministic	NP

Alternation in complexity theory

What problems can be solved by a machine running in polynomial time?

deterministic	P
non-deterministic	NP
only universal states	co-NP

Alternation in complexity theory

What problems can be solved by a machine running in polynomial time?

deterministic	P
non-deterministic	NP
only universal states	co-NP
only existential, then only universal	Σ_2^P

Alternation in complexity theory

What problems can be solved by a machine running in polynomial time?

deterministic	P
non-deterministic	NP
only universal states	co-NP
only existential, then only universal	Σ_2^P
only universal, then only existential	Π_2^P

Alternation in complexity theory

What problems can be solved by a machine running in polynomial time?

deterministic	P
non-deterministic	NP
only universal states	co-NP
only existential, then only universal	Σ_2^P
only universal, then only existential	Π_2^P
a fixed number of alternations	PH

Alternation in complexity theory

What problems can be solved by a machine running in polynomial time?

deterministic	P
non-deterministic	NP
only universal states	co-NP
only existential, then only universal	Σ_2^P
only universal, then only existential	Π_2^P
a fixed number of alternations	PH
any number of alternations	AP $=$ PSPACE

The order of quantifiers matters!

Alternating automata

Not so useful in case of finite automata

- The order of quantifiers matters, and this makes some constructions (e.g. quantifiers, negation) not as easy as we would like

Alternating automata

Not so useful in case of finite automata

- The order of quantifiers matters, and this makes some constructions (e.g. quantifiers, negation) not as easy as we would like
- By using the subset construction we get a NFA with 2^n states, we have to determinize again to get a DFA with 2^{2^n} states
- Also we can obtain DFA running in reverse with 2^n states

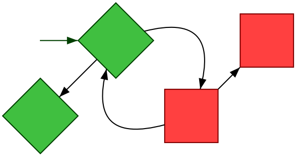
Alternating automata

Not so useful in case of finite automata

- The order of quantifiers matters, and this makes some constructions (e.g. quantifiers, negation) not as easy as we would like
- By using the subset construction we get a NFA with 2^n states, we have to determinize again to get a DFA with 2^{2^n} states
- Also we can obtain DFA running in reverse with 2^n states
- Still an useful notion

Infinite cycles

What happens if we get into an infinite cycle?



Usually we assume infinite computations to be non-accepting
But in terms of games this means Eva loses – that's **not fair!**

Infinite computations in nature

Maybe we **want** infinite computations?

Operating systems, control systems, and hardware run potentially forever

Infinite computations in nature

Maybe we **want** infinite computations?
Operating systems, control systems, and hardware run potentially forever

We actually **want** the computation to be infinite, but it is required to satisfy some property ϕ, for example:

Infinite computations in nature

Maybe we **want** infinite computations?
Operating systems, control systems, and hardware run potentially forever

We actually **want** the computation to be infinite, but it is required to satisfy some property ϕ, for example:

for every request there is a response
there is no response if it was not requested
no deadlocks, no starvation...

We want Eva to win the game iff ϕ is satisfied

Infinite games in nature

How do other games solve the problem?

- Chess – the game is considered a draw after 50 moves (without an irreversible action such as moving a pawn or capturing)
- Go – ko rule

But we don't want draws!

Infinite games in mathematics

Banach-Mazur game (1930)
Pick a set $Z \subseteq \mathbb{R}$

Infinite games in mathematics

Banach-Mazur game (1930)
Pick a set $Z \subseteq \mathbb{R}$

- Adam chooses an open interval $I_1 \subseteq \mathbb{R}$
- Eva chooses $I_2 \subseteq I_1$
- Adam chooses $I_3 \subseteq I_2$
- Eva chooses $I_4 \subseteq I_3$
- ...

Infinite games in mathematics

Banach-Mazur game (1930)
Pick a set $Z \subseteq \mathbb{R}$

- Adam chooses an open interval $I_1 \subseteq \mathbb{R}$
- Eva chooses $I_2 \subseteq I_1$
- Adam chooses $I_3 \subseteq I_2$
- Eva chooses $I_4 \subseteq I_3$
- ...

Eva wins if the intersection of all intervals is a subset of Z

Infinite games in mathematics

Banach-Mazur game (1930)
Pick a set $Z \subseteq \mathbb{R}$

- Adam chooses an open interval $I_1 \subseteq \mathbb{R}$
- Eva chooses $I_2 \subseteq I_1$
- Adam chooses $I_3 \subseteq I_2$
- Eva chooses $I_4 \subseteq I_3$
- ...

Eva wins if the intersection of all intervals is a subset of Z

Question: for which sets Z Eva has a winning strategy?

Infinite games in mathematics

Banach-Mazur game (1930)
Pick a set $Z \subseteq \mathbb{R}$

- Adam chooses an open interval $I_1 \subseteq \mathbb{R}$
- Eva chooses $I_2 \subseteq I_1$
- Adam chooses $I_3 \subseteq I_2$
- Eva chooses $I_4 \subseteq I_3$
- ...

Eva wins if the intersection of all intervals is a subset of Z

Question: for which sets Z Eva has a winning strategy?

Answer: Eva has a winning strategy iff $\mathbb{R} - Z$ is a meager set

Determinacy

What does it mean that a player has a winning strategy?

Determinacy

What does it mean that a player has a winning strategy?

A strategy = a decision procedure which tells which move a player should do in given situation (i.e. a function from situations to moves)

Determinacy

What does it mean that a player has a winning strategy?

A strategy = a decision procedure which tells which move a player should do in given situation (i.e. a function from situations to moves)

A winning strategy = a strategy such that the player using that strategy will always win, no matter what the opponent is doing

Determinacy

What does it mean that a player has a winning strategy?

A strategy = a decision procedure which tells which move a player should do in given situation (i.e. a function from situations to moves)

A winning strategy = a strategy such that the player using that strategy will always win, no matter what the opponent is doing

Impossible for both players to have winning strategies

Determinacy

What does it mean that a player has a winning strategy?

A strategy = a decision procedure which tells which move a player should do in given situation (i.e. a function from situations to moves)

A winning strategy = a strategy such that the player using that strategy will always win, no matter what the opponent is doing

Impossible for both players to have winning strategies

In finite games, one of the players will have a winning strategy (the game is determined)

Determinacy

What does it mean that a player has a winning strategy?

A strategy = a decision procedure which tells which move a player should do in given situation (i.e. a function from situations to moves)

A winning strategy = a strategy such that the player using that strategy will always win, no matter what the opponent is doing

Impossible for both players to have winning strategies

In finite games, one of the players will have a winning strategy (the game is determined)

We can use the axiom of choice to construct infinite games which are not determined

Determinacy

What does it mean that a player has a winning strategy?

A strategy = a decision procedure which tells which move a player should do in given situation (i.e. a function from situations to moves)

A winning strategy = a strategy such that the player using that strategy will always win, no matter what the opponent is doing

Impossible for both players to have winning strategies

In finite games, one of the players will have a winning strategy (the game is determined)

We can use the axiom of choice to construct infinite games which are not determined

But **reasonable** infinite games are determined (Martin '75 - Borel determinacy)

A non-determined game

XOR function:

$$X : \{0,1\}^* \to \{0,1\}$$

such that

$$X(0^*) = 0$$
$$X(u0v) \neq X(u1v)$$

A well known function

Non-determined game

Infinite XOR function:

$$X : \{0,1\}^\omega \to \{0,1\}$$

such that

$$X(0^\omega) = 0$$
$$X(u0v) \neq X(u1v)$$

Existence follows from the Axiom of Choice

Non-determined game

Infinite XOR game

- Eva chooses a finite sequence of bits w_1
- Adam chooses w_2
- Eva chooses w_3
- Adam chooses w_4
- ...
- Eva wins if $X(w_1 w_2 w_3 w_4 \ldots) = 1$

Non-determined game

Infinite XOR game

- Eva chooses a finite sequence of bits w_1
- Adam chooses w_2
- Eva chooses w_3
- Adam chooses w_4
- ...
- Eva wins if $X(w_1 w_2 w_3 w_4 \ldots) = 1$

- Eva chooses w_1
- Adam chooses $w_2' w_3$
- Eva chooses w_4
- Adam chooses w_5
- ...
- Adam will win!

Back to computations: how to express ϕ?

In terms of logic (FO, MSO): no problem ($\forall i$ now quantifies not over a finite set of positions in a word, but an infinite set of integers $\mathbb{N}$); there are also special logics for that (e.g. LTL)

Back to computations: how to express ϕ?

In terms of logic (FO, MSO): no problem ($\forall i$ now quantifies not over a finite set of positions in a word, but an infinite set of integers $\mathbb{N}$); there are also special logics for that (e.g. LTL)

In terms of ω-**languages**: we speak about subsets of Σ^ω instead of Σ^*

Back to computations: how to express ϕ?

In terms of logic (FO, MSO): no problem ($\forall i$ now quantifies not over a finite set of positions in a word, but an infinite set of integers $\mathbb{N}$); there are also special logics for that (e.g. LTL)

In terms of ω-**languages**: we speak about subsets of Σ^ω instead of Σ^*

In terms of ω-**regular expressions**: for $L \in \Sigma^*$ we add an operation L^ω

$$\Sigma^\omega \ni baabbaabb(ab)^\omega$$

What about automata?

ω-regular expressions and MSO logic express the same class of languages (called ω-**regular languages**)

But what about automata?

What about automata?

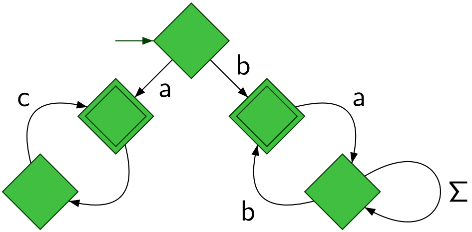
ω-regular expressions and MSO logic express the same class of languages (called ω-**regular languages**)
But what about automata?

The simplest approach: make all infinite computations false/losing (or true/winning) – not powerful enough, we cannot express $a^\omega \in \{a, b\}^\omega$
We need to use some acceptance condition (or winning condition) to tell which infinite runs are accepted

Büchi automata ('60)

We again use F - the set of accepting states, but now the infinite computation is accepting if it visits the states in F infinitely often (Büchi condition)

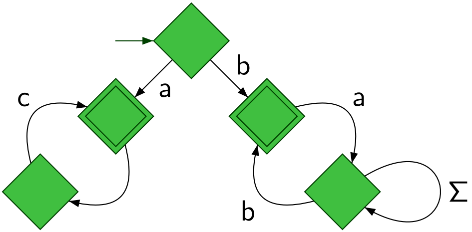
$$ac^{\omega} \cup (ba\Sigma^*)^{\omega}$$



Büchi automata ('60)

We again use F - the set of accepting states, but now the infinite computation is accepting if it visits the states in F infinitely often (Büchi condition)

$$ac^{\omega} \cup (ba\Sigma^*)^{\omega}$$



$$K_1 L_1^{\omega} \cup K_2 L_2^{\omega}$$

Büchi automata cont

- Languages recognized by non-deterministic Büchi automata are exactly ω-regular languages

Büchi automata cont

- Languages recognized by non-deterministic Büchi automata are exactly ω-regular languages
- Deterministic Büchi automata cannot recognize $(a+b)^* a^\omega$

Büchi automata cont

- Languages recognized by non-deterministic Büchi automata are exactly ω-regular languages
- Deterministic Büchi automata cannot recognize $(a+b)^*a^\omega$
- Negation is not straightforward

Muller automata ('63)

We use $\mathcal{F} \subseteq P(Q)$
Run is accepted iff the set of states appearing infinitely often during the play is in $\mathcal{F}$

Muller automata ('63)

We use $\mathcal{F} \subseteq P(Q)$
Run is accepted iff the set of states appearing infinitely often during the play is in $\mathcal{F}$

- Deterministic Muller automata recognize all ω-regular languages
- Negation is straightforward
- But the description is long (we have to define acceptance for each subset of Q)

Parity condition: motivation

Büchi conditions allows us to define a good thing that has to happen infinitely often in order to make Eva a winner.

Parity condition: motivation

Büchi conditions allows us to define a good thing that has to happen infinitely often in order to make Eva a winner.
In practice, both good and bad things could happen...

the program seems to do its job
the program uses too much resources
the program hangs
the program works as it should
we lose some money
we break our moral rules
we earn some money
we become rich
newspapers write about us
we go to jail

Parity condition: motivation

Büchi conditions allows us to define a good thing that has to happen infinitely often in order to make Eva a winner.
In practice, both good and bad things could happen...

the program seems to do its job	0
the program uses too much resources	1
the program hangs	1
the program works as it should	2
we lose some money	3
we break our moral rules	3
we earn some money	4
we become rich	4
newspapers write about us	4
we go to jail	5

Parity condition

We use $\mathrm{rank} : Q \to \mathbb{N}$
Run is accepted (Eva wins) if the greatest rank appearing infinitely often during the play is even, not accepted (Adam wins) if it is odd

Parity condition

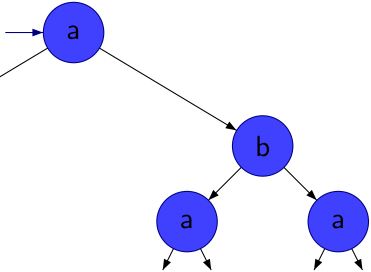
We use $\mathrm{rank} : Q \to \mathbb{N}$
Run is accepted (Eva wins) if the greatest rank appearing infinitely often during the play is even, not accepted (Adam wins) if it is odd

- Deterministic parity automata recognize all ω-regular languages (a nice translation from the Muller condition, LAR)
- Negation straightforward
- Effective description

More than words: ω-trees

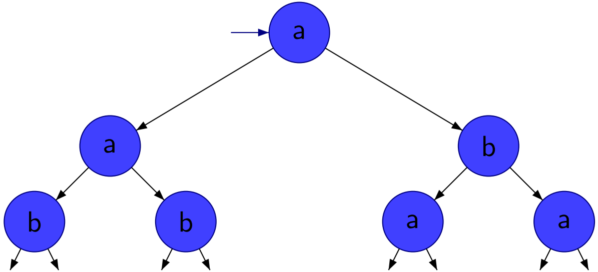
In an ω-word, each position has **one successor**

In an ω-tree, a position can have **many successors**



More than words: ω-trees

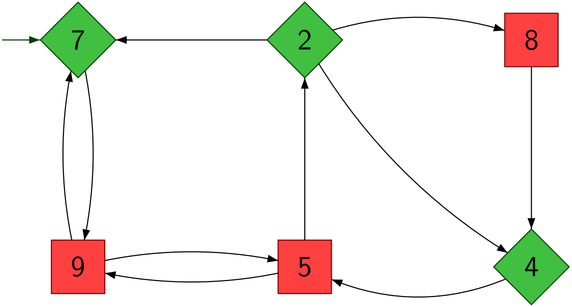
In an ω-word, each position has **one successor**
In an ω-tree, a position can have **many successors**



A tree can represent e.g. all possible runs of an operating system

Infinite game

Alternating automaton = a transition system, where transitions depends on decisions of two players and input
What happens if we remove the input?



Motivation I-II

- We can encode the input inside our automaton, getting an infinite game without input (for infinite "irregular" inputs this leads to an infinite transition system)

Motivation I-II

- We can encode the input inside our automaton, getting an infinite game without input (for infinite "irregular" inputs this leads to an infinite transition system)
- We cannot run automata on infinite input in practice; but we want to solve problems like:
  - Given ϕ - a property that we want our program to satisfy during its execution
  - Given M - a model of our program
  - Question: does M satisfy ϕ? (**model checking**)

For ϕ in modal μ calculus, this reduces to a parity game

Motivation III

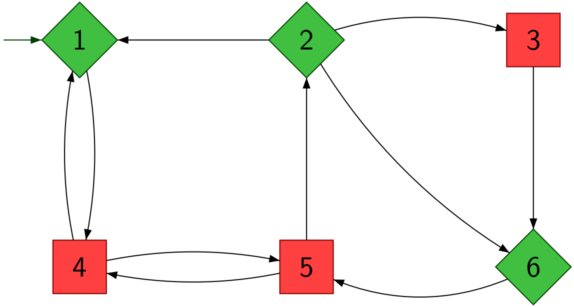
Our game models a system whose task is to provide outputs for given inputs

- States Q model possible states of our system
- Adam's moves model possible inputs
- Eva's moves model possible outputs
- The winning condition models whether Eva responded according to our needs (given by a formula ϕ)

If Eva wins such a game, then it is possible to implement a system which works according to ϕ

Parity games

Given: a parity game (an infinite game using the parity acceptance condition)



Question: who has a winning strategy?

Positional determinacy

A winning condition is determined if one of the players has a winning strategy

Reasonable winning conditions (parity, Muller, etc) are determined

Positional determinacy

A winning condition is determined if one of the players has a winning strategy
Reasonable winning conditions (parity, Muller, etc) are determined

A winning condition is positionally determined if one of the players has a positional strategy: always makes the same move in each of his positions
Parity condition is positionally determined
(Emerson, Jutla '91; Mostowski '91; McNaughton '93)

Positional determinacy

A winning condition is determined if one of the players has a winning strategy
Reasonable winning conditions (parity, Muller, etc) are determined

A winning condition is positionally determined if one of the players has a positional strategy: always makes the same move in each of his positions
Parity condition is positionally determined
(Emerson, Jutla '91; Mostowski '91; McNaughton '93)

Even on infinite arenas (useful theoretically, e.g. when complementing automata on ω-trees)

Parity games: algorithms

Let n - the number of states, d - the number of ranks in the parity condition

- Solving parity games is in NP (we guess Eva's positional strategy, and obtain a single player game, which is easy to solve)

Parity games: algorithms

Let n - the number of states, d - the number of ranks in the parity condition

- Solving parity games is in NP (we guess Eva's positional strategy, and obtain a single player game, which is easy to solve)
- Solving parity game is also in co-NP (we could also guess Adam's positional strategy)

Parity games: algorithms

Let n - the number of states, d - the number of ranks in the parity condition

- Solving parity games is in NP (we guess Eva's positional strategy, and obtain a single player game, which is easy to solve)
- Solving parity game is also in co-NP (we could also guess Adam's positional strategy)
- $O(n^{d/2})$ (Jurdziński '00)

Parity games: algorithms

Let n - the number of states, d - the number of ranks in the parity condition

- Solving parity games is in NP (we guess Eva's positional strategy, and obtain a single player game, which is easy to solve)
- Solving parity game is also in co-NP (we could also guess Adam's positional strategy)
- $O(n^{d/2})$ (Jurdziński '00)
- strategy improvement algorithms (Jurdziński, Vöge '00)

Parity games: algorithms

Let n - the number of states, d - the number of ranks in the parity condition

- Solving parity games is in NP (we guess Eva's positional strategy, and obtain a single player game, which is easy to solve)
- Solving parity game is also in co-NP (we could also guess Adam's positional strategy)
- $O(n^{d/2})$ (Jurdziński '00)
- strategy improvement algorithms (Jurdziński, Vöge '00)
- $n^{O(\sqrt{n})}$ (Jurdziński, Paterson, Zwick '06)

Parity games: algorithms

Let n - the number of states, d - the number of ranks in the parity condition

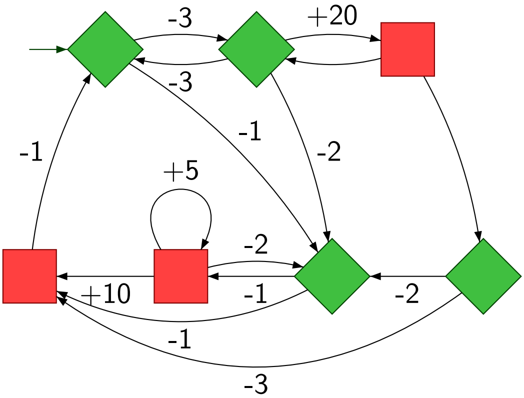
- Solving parity games is in NP (we guess Eva's positional strategy, and obtain a single player game, which is easy to solve)
- Solving parity game is also in co-NP (we could also guess Adam's positional strategy)
- $O(n^{d/2})$ (Jurdziński '00)
- strategy improvement algorithms (Jurdziński, Vöge '00)
- $n^{O(\sqrt{n})}$ (Jurdziński, Paterson, Zwick '06)
- $O(n^{d/3})$ (Schewe '07)

Parity games: algorithms

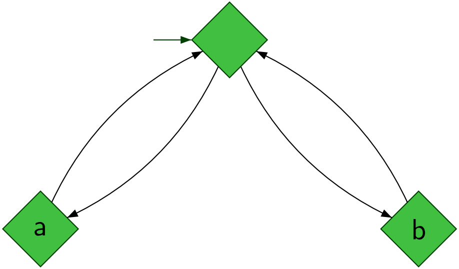
Let n - the number of states, d - the number of ranks in the parity condition

- Solving parity games is in NP (we guess Eva's positional strategy, and obtain a single player game, which is easy to solve)
- Solving parity game is also in co-NP (we could also guess Adam's positional strategy)
- $O(n^{d/2})$ (Jurdziński '00)
- strategy improvement algorithms (Jurdziński, Vöge '00)
- $n^{O(\sqrt{n})}$ (Jurdziński, Paterson, Zwick '06)
- $O(n^{d/3})$ (Schewe '07)
- No polynomial algorithm known yet

Mean payoff game

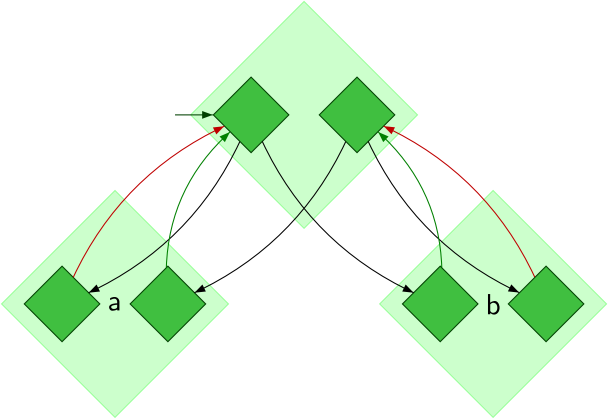


Muller game



Eva wants both a and b to appear infinitely often

Muller game



Eva wants both *a* and *b* to appear infinitely often

Muller game via LAR (DJW '97)

In Muller games we have winning strategies with finite memory:

Eva has a deterministic finite automaton which changes memory states depending on what happens in the game (i.e., the sequence of game states), and her move depends only on the current game state and the current memory state

Muller game via LAR (DJW '97)

In Muller games we have winning strategies with finite memory:

Eva has a deterministic finite automaton which changes memory states depending on what happens in the game (i.e., the sequence of game states), and her move depends only on the current game state and the current memory state

Strategies using a small amount of memory are good in practice (useful for automatic synthesis)

More infinite games

- each move is assigned a **color** from a finite subset C

More infinite games

- each move is assigned a **color** from a finite subset C
- the winning condition is given as an ω-**regular language** $W \subseteq C^\omega$

More infinite games

- each move is assigned a **color** from a finite subset C
- the winning condition is given as an ω-**regular language** $W \subseteq C^\omega$
- Question: for a winning condition, what is the smallest possible size of an automaton $\mathcal{M}$ over C such that whenever Eva can win a game using W as a winning condition, she can win using $\mathcal{M}$ as memory?

More infinite games

- each move is assigned a **color** from a finite subset C
- the winning condition is given as an ω-**regular language** $W \subseteq C^\omega$
- Question: for a winning condition, what is the smallest possible size of an automaton $\mathcal{M}$ over C such that whenever Eva can win a game using W as a winning condition, she can win using $\mathcal{M}$ as memory?
- three transition systems come into play

arena $\qquad W \qquad \mathcal{M}$

More infinite games

- each move is assigned a **color** from a finite subset C
- the winning condition is given as an ω-**regular language** $W \subseteq C^\omega$
- Question: for a winning condition, what is the smallest possible size of an automaton $\mathcal{M}$ over C such that whenever Eva can win a game using W as a winning condition, she can win using $\mathcal{M}$ as memory?
- three transition systems come into play

  arena $\quad W \quad \mathcal{M}$
- solvable in single polynomial time

Research problems

- Given a winning condition W, how to effectively decide who wins the game on given arena?
- Can the winner win using a simple strategy (positional, small memory)?
- Are there any characterizations which allow us to immediately tell that games using given winning condition are positionally determined?

Conclusion

Summary

- non-deterministic automata, FO and MSO logic
- alternating automata
- ω-regular languages
- infinite games
- parity games

thank you