Automata Theoretic Foundations of Infinite Games

Tutorial, Amsterdam, February 2007

Wolfgang Thomas

Chair of Computer Science 7
RWTH Aachen University

thomas@informatik.rwth-aachen.de

RWTH AACHEN

Plan

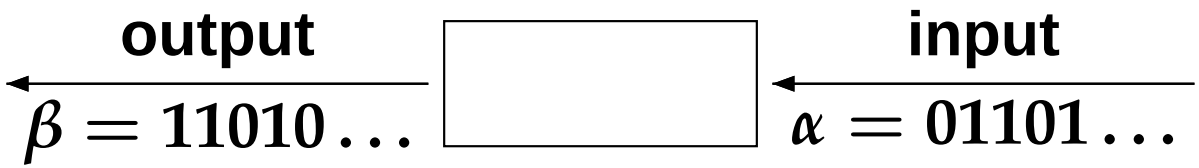
1. Church's Problem
2. Basic Terminology and Main Results
3. Reachability Games and Weak Muller Games
4. Muller Games and Parity Games
5. Infinite Game Graphs
6. Tree Automata
7. Selected Sources

Part 1

Church's Problem

Basic Question

Consider a bit-by-bit transformation of bit streams



Church's Problem:

For a given I/O specification fill the box!

Given a logical specification of the input-output relation R

find a circuit C mapping $\alpha \mapsto C(\alpha)$

such that $(\alpha, C(\alpha)) \in R$ for all α

Aim: Presentation of a solution of Church's Problem

More Detail

Church posed three questions:

1. **Solution:**
   Given C and a definition of R, decide whether C realizes R
2. **Solvability:**
   Given a definition of R, decide whether there is a C which realizes R
3. **Synthesis (to be treated here):**
   Given a definition of R, if possible construct C which realizes R

Natural view: Infinite game between two players:

A supplies input bit, B supplies output bit, etc. in alternation

Example

Consider R defined by the conjunction of three conditions on the input-output stream (α, β):

1. $\forall t : \alpha(t) = 1 \;\rightarrow\; \beta(t) = 1$
2. $\neg\exists t : \beta(t) = \beta(t+1) = 0$
3. $\exists^{\omega} t \alpha(t) = 0 \;\rightarrow\; \exists^{\omega} t \beta(t) = 0$

S1S formula $\varphi(X, Y)$:

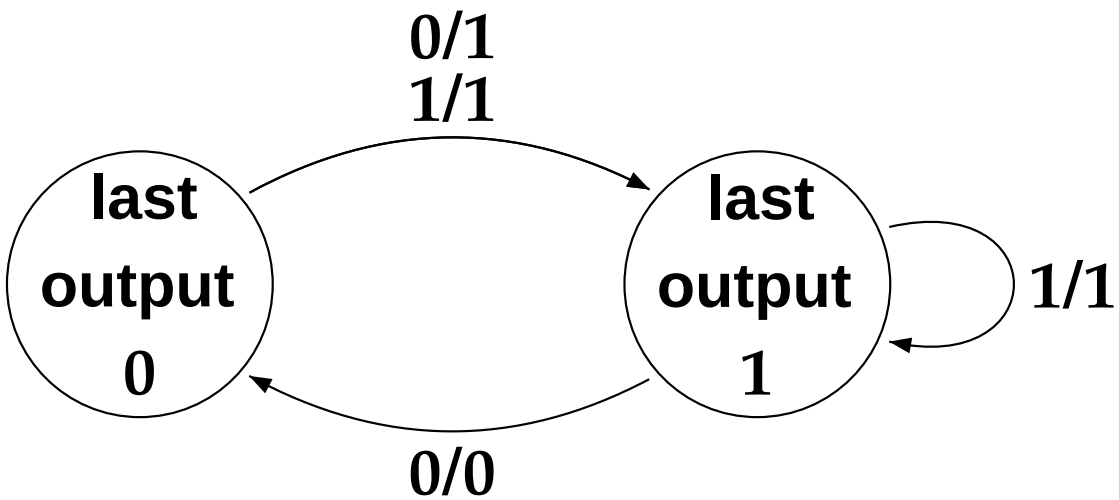
$$\forall t(X(t) \rightarrow Y(t))$$

$$\wedge\neg\exists t(\neg Y(t) \wedge \neg Y(t+1))$$

$$\wedge(\forall s \exists t > s \neg X(t) \;\rightarrow\; \forall u \exists v > u \neg Y(v))$$

Common-Sense Solution

- for input 1 produce output 1
- for input 0 produce
  - output 1 if last output was 0
  - output 0 if last output was 1



This is a finite-state strategy

Overview of Synthesis Method

1. **Convert the logical specification into a Muller automaton**
2. **Solve the synthesis problem in the automata theoretic framework**

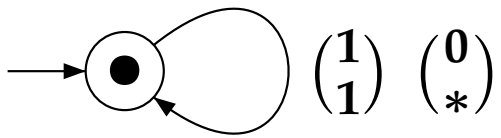
For item 1 we use the Büchi-McNaughton theory of ω-automata

Here we concentrate on the game theoretical part.

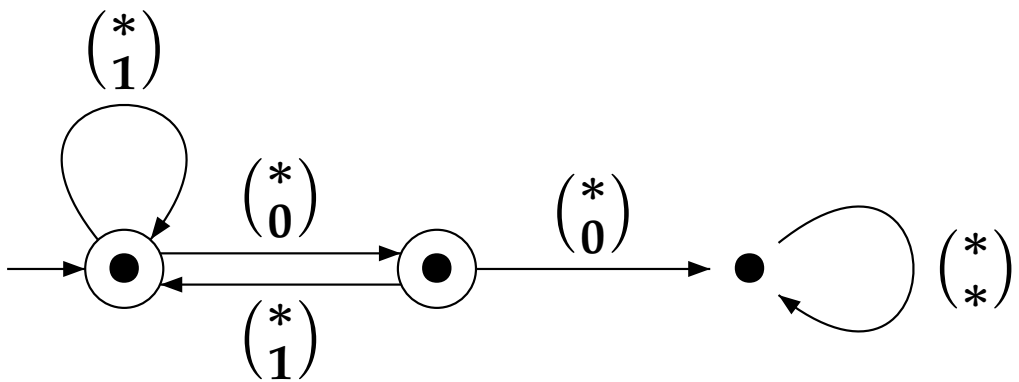
Büchi Automata

Büchi automaton checks sequence of bit pairs $(\alpha(t), \beta(t))$.

1. $\forall t : \alpha(t) = 1 \;\rightarrow\; \beta(t) = 1$



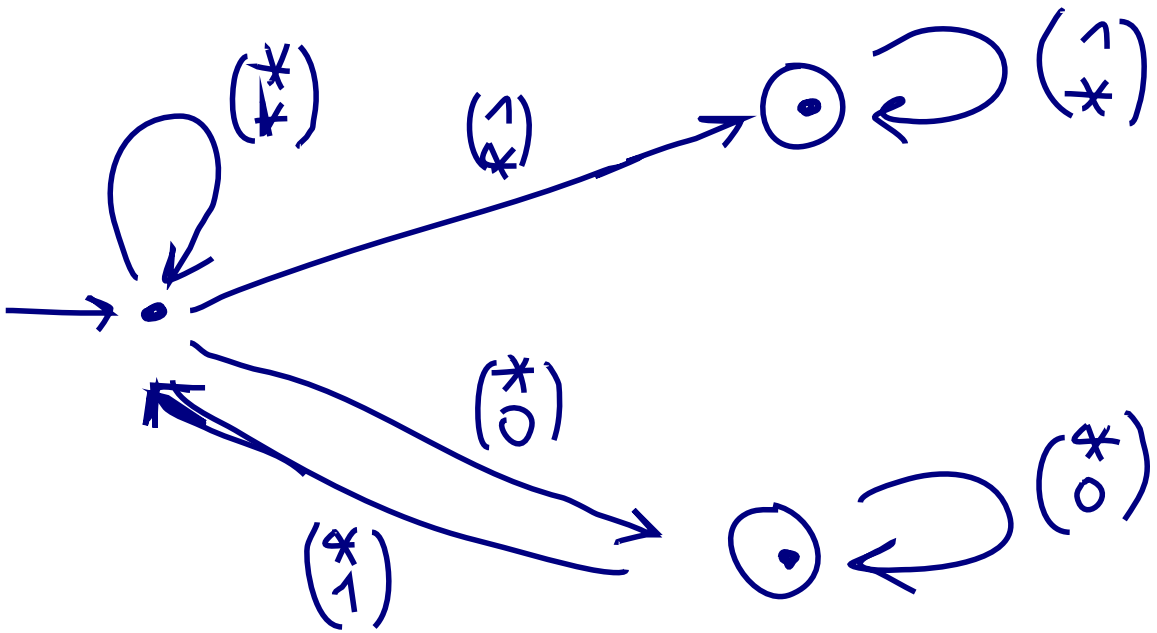
2. $\neg\exists t : \beta(t) = \beta(t+1) = 0$



Büchi Automata Continued

3. $\exists^{\omega} t \alpha(t) = 0 \quad \rightarrow \quad \beta(t) = 0$

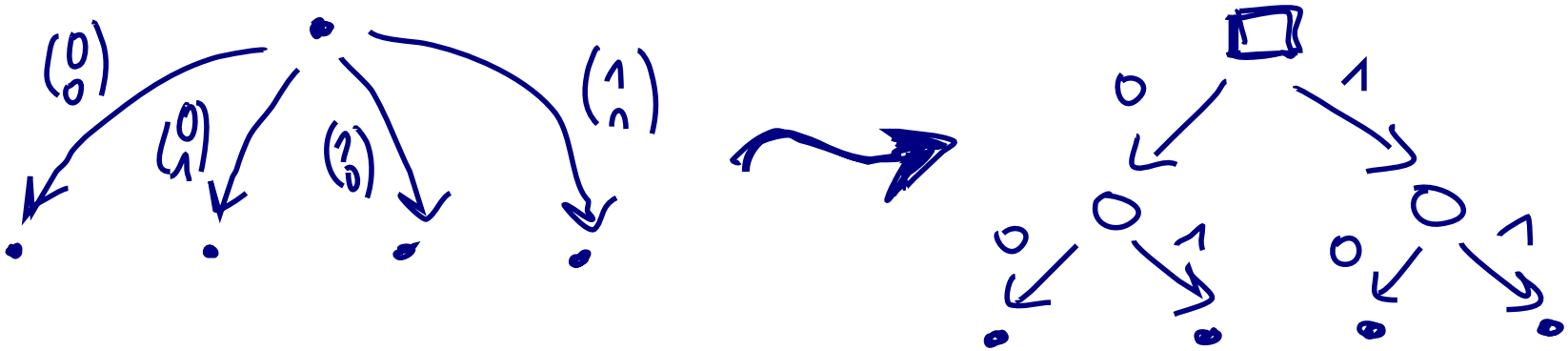
   $\exists^{<\omega} t \alpha(t) = 0 \quad \vee \quad \exists^{\omega} t \beta(t) = 0$



Towards Game Graphs

We transform the Büchi automaton to a deterministic Muller automaton.

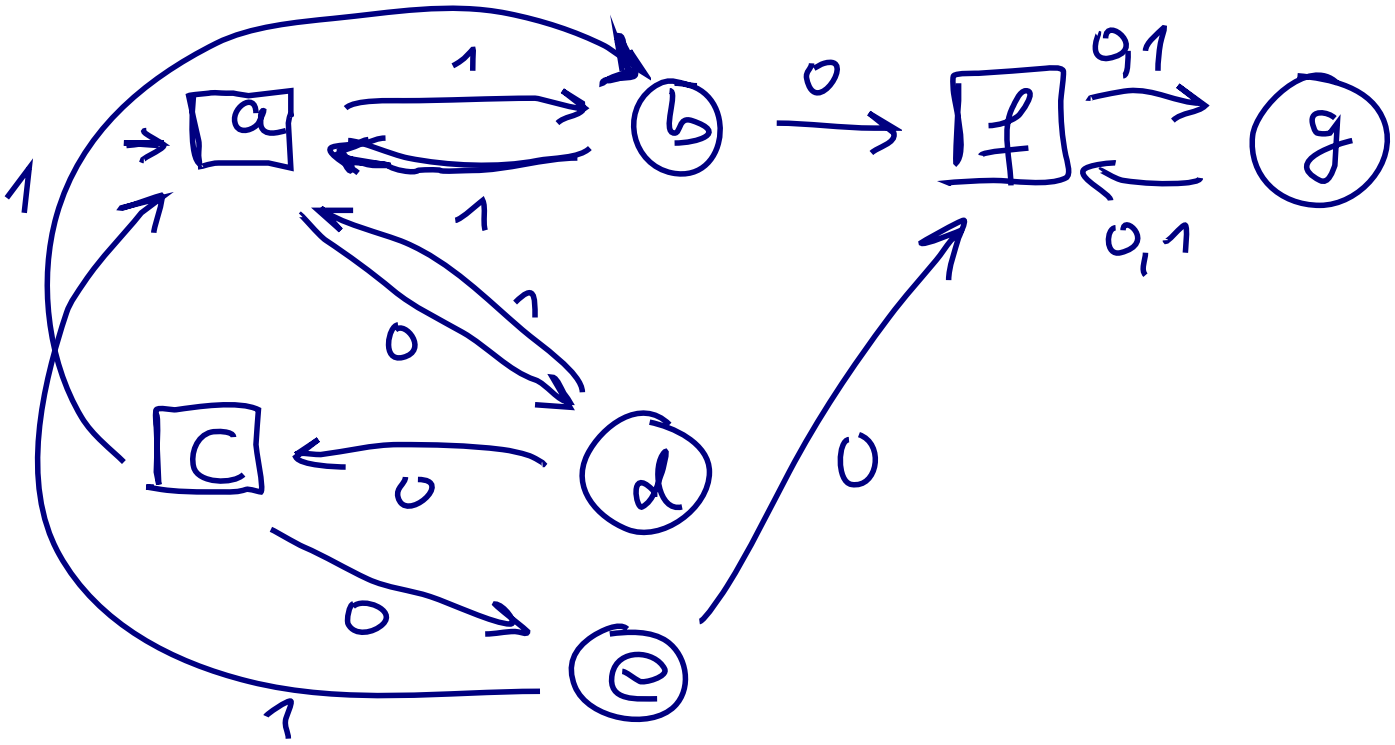
We separate the two contributions “input” and “output” of an automaton transition.



The result is a game graph with a Muller winning condition.

We solve Church’s Problem in this framework.

Muller Automaton



Acceptance: If c or d are visited infinitely often, then so is e.

The set of states visited infinitely often can be

$\{a, b\}$ or $\{a, c, d, e\}$ or $\{a, b, c, d, e\}$.

Part 2

Basic Terminology and Main Results

Formal Framework

We consider two-person games between players A and B.

An infinite game is a pair $\Gamma = (G, \varphi)$ of two components:

- a transition graph G (arena of the game), so that the infinite paths through G are the plays of the game,
- a winning condition φ on plays, describing those plays which are won by player 0.

For the second item φ one can use the Muller condition:

a collection $\mathcal{F} = \{F_1, \ldots, F_k\}$ of state sets such that

player B wins the play ρ if the states visited infinitely often in ρ forms a set in $\mathcal{F}$.

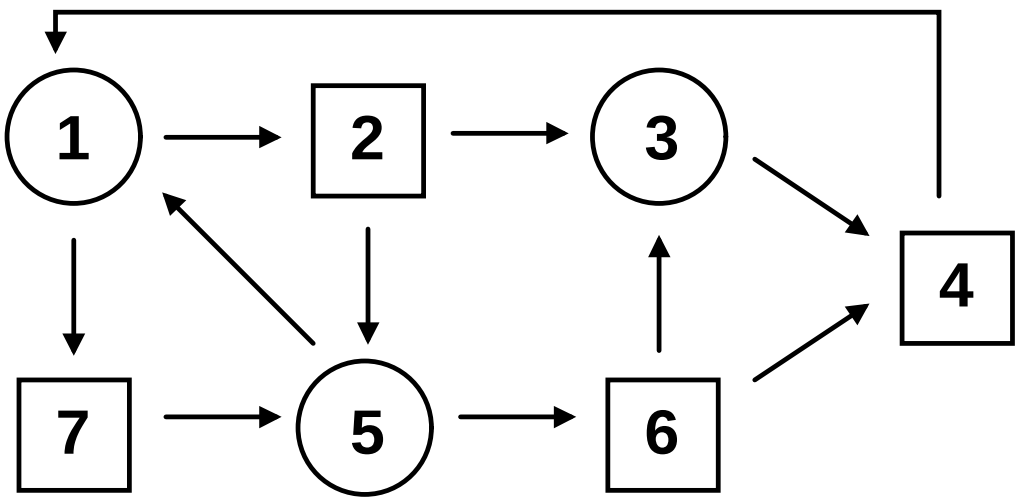
Game Graphs

A game graph (or arena) has the form $G = (Q, Q_A, E)$ where $Q_A \subseteq Q$ and $E \subseteq Q \times Q$ is the transition relation, satisfying

$$\forall q \in Q : qE \neq \emptyset \quad \text{(i.e. } \forall q \exists q' : (q, q') \in E\text{)}$$

We set $Q_B := Q \setminus Q_A$

Example:



A play is a sequence $\rho = r_0 r_1 r_2 \ldots$ with $(r_i, r_{i+1}) \in E$

Weak and Strong Muller Conditions

We use two fundamental forms of winning conditions:

- **A weak Muller condition refers to the visited states:**
  $\mathrm{Occ}(\rho) = \{q \in Q \mid \exists i \rho(i) = q\}$
  For a collection $\mathcal{F}$ of state-sets
  play ρ is won by player B iff $\mathrm{Occ}(\rho) \in \mathcal{F}$
- **A (strong) Muller condition refers to the infinitely often visited states:**
  $\mathrm{Inf}(\rho) = \{q \in Q \mid \exists^{\omega} i \rho(i) = q\}$
  For a collection $\mathcal{F}$ of state-sets
  play ρ is won by player B iff $\mathrm{Inf}(\rho) \in \mathcal{F}$

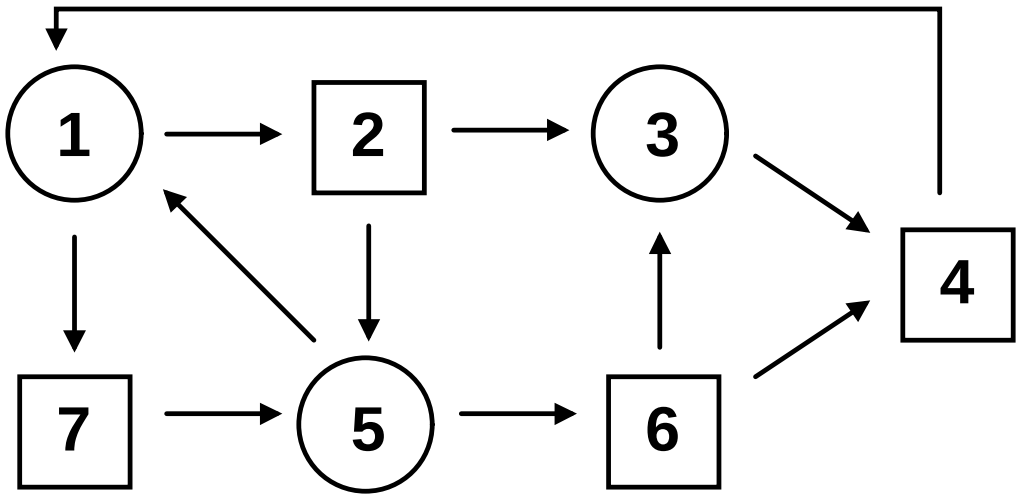
One speaks of weak Muller games and of Muller games.

Reachability-, Safety-, Recurrence Games

Special cases:

- **Reachability game is specified by state set F**
  Play ρ is won by B if $\exists i \rho(i) \in F$
  in other words: $\mathrm{Occ}(\rho) \cap F \neq \emptyset$
- **Safety game is specified by state set S:**
  Play ρ is won by B if $\forall i \rho(i) \in S$
  in other words: $\mathrm{Occ}(\rho) \subseteq S$
- **Recurrence game (or Büchi game) is specified by set R:**
  Play ρ is won by B if $\exists^{\omega} i \rho(i) \in R$
  in other words: $\mathrm{Inf}(\rho) \cap R \neq \emptyset$

Example



Winning condition 1: $3 \in \mathrm{Occ}(\rho)$

$W_B = \{3\}, \quad W_A = \{1,2,4,5,6,7\}$

Winning condition 2: $\{2,7\} \subseteq \mathrm{Inf}(\rho)$

$W_B = \{1,2,3,4,5,6,7\} \;\; W_A = \emptyset$

Strategy requires memory of one bit:

For example, from vertex 1 pass to 2 and 7 in alternation.

Strategies

A **strategy** for player B from q is a function $f : Q^+ \to Q$, specifying for any play prefix $q_0 \dots q_k$ with $q_0 = q$ and $q_k \in Q_B$ some vertex $r \in Q$ with $(q_k, r) \in E$

A play $\rho = q_0 q_1 \dots$ from $q_0 = q$ is **played according to strategy** f if for each $q_i \in Q_B$ we have $q_{i+1} = f(q_0 \dots q_i)$

A strategy f for player B from q is called **winning strategy for player B from** q if any play from q which is played according to f is won by player B (according to the winning condition).

In the analogous way, one introduces strategies and winning strategies for player A.

Player B (resp. A) **wins from** q if s/he has a winning strategy from q

Winning Regions and Determinacy

For a game $\Gamma = (G, \varphi)$ with $G = (Q, Q_A, E)$, the winning regions of players A and B are the sets

$$W_A := \{q \in Q \mid \text{player A wins from } q\}$$

$$W_B := \{q \in Q \mid \text{player B wins from } q\}$$

Remark: **Each vertex q belongs at most to W_A or W_B.**

So the winning regions W_A, W_B are disjoint.

A game is called determined if $W_A \cup W_B = Q$, i.e. from each vertex one of the two players has a winning strategy.

On Determinacy

1. There are (exotic) games which are not determined.
2. In descriptive set theory one investigates which abstract winning conditions define determined games.
3. Martin's Theorem (1975): If an ω-language is a Borel set, then the associated game is determined.
4. The games based on Occ- and Inf-conditions are Borel (and thus determined).

Synthesis Problem Restated

Given a game $\Gamma = (G, \varphi)$, $G = (Q, Q_A, E)$

1. **Decide for each $q \in Q$ whether $q \in W_B$ (i.e. whether player B wins from q)**
2. **In this case:**
   Construct a suitable winning strategy from q (in the form of a program computing an appropriate function $f_B : Q^+ \to Q$)
3. **Optimize the construction of the winning strategy (e.g., time complexity) or optimize parameters of the winning strategy (e.g., size of memory, quality of satisfaction of winning condition).**

Solving a game means to provide algorithms for 1. and 2.

Special Strategies

If Q is finite, then a strategy is a word function $f : Q^+ \to Q$

There are three basic types of strategies:

1. **A natural requirement: f is computable (computable or recursive strategy)**
2. **More restrictive requirement: f is a finite-state strategy or computable by a finite automaton (automaton strategy, which allows to use "bounded information" from the past)**
3. **Simplest kind of strategy: Current vertex determines choice of next vertex (positional strategy)**

Other types: pushdown strategy, counter strategy etc.

Positional Strategies

Given $G = (Q, Q_A, E)$

A strategy $f : Q^+ \to Q$ (say for player B) is called positional (sometimes also called local or memoryless) if the value $f(q_0 \dots q_k)$ only depends on $q_k \in Q_B$

A positional strategy for B is representable as

1. **a function $F : Q_B \to Q$ or**
2. **a set E_B of edges of the game graph, containing for every $q \in Q_B$ precisely one edge with source vertex q (and for $q \in Q_A$ all edges from E with source vertex q).**

Finite-State Strategies

A finite strategy automaton for player 0 over $G = (Q, E)$ is a Mealy-automaton $\mathcal{A} = (S, Q, s_0, \sigma, \tau)$ with

- **finite memory S, input alphabet Q**
- **initial memory content $s_0 \in S$**
- **memory update function $\sigma : S \times Q \to S$**
- **transition choice function $\tau : S \times Q_0 \to Q$**

Thus a Mealy-automaton represents a finite-state controller.

Application Scenario I: Reactive Systems

- Application areas: communication protocols, control systems
- Use game graph to model reactive system
- Use temporal logics or modal μ-calculus as specification logic for the winning condition

History: Early contributions by

- McNaughton, Büchi, Landweber, Rabin in the 1960's
- Gurevich, Harrington, Emerson, Jutla, Mostowski, Pnueli in the 1980's and 1990's
- More recent work: See sources at end of tutorial

Application Scenario II: Model-Checking

Question:

Given Kripke structure $\mathcal{K}$ and formula φ (e.g. of modal μ-calculus), decide whether $\mathcal{K} \models \varphi$

Combine $\mathcal{K}$ and φ into a "product game graph" $G_{\mathcal{K},\varphi}$ with the so-called "parity winning condition" such that

$\mathcal{K} \models \varphi$ iff

from a designated vertex in $G_{\mathcal{K},\varphi}$, player B has a (positional) winning strategy.

This transformation is due to Emerson, Jutla, Sistla (CAV 1995)

Application Scenario III: Tree Automata Theory

Nondeterministic tree automaton (e.g. Rabin tree automaton) generates run on input tree and accepts if all paths of the run satisfy acceptance condition.

Existence of successful run corresponds to existence of winning strategy in a game.

Complementation of (e.g.) Rabin tree automata can be solved with a determinacy theorem on this game:

Task: From non-existence of accepting run of given automaton conclude the existence of accepting run for the complement automaton.

Determinacy is useful here.

Main Theorems of Tutorial

- For a finite game graph with n states and a weak Muller winning condition, one can effectively determine the winning regions and construct a finite-state winning strategy for the respective winning player (using 2^n internal states).

- For a finite game graph with n states and a Muller winning condition, one can effectively determine the winning regions and construct a finite-state winning strategy for the respective winning player (using $n! \cdot n$ internal states).

Part 3

Reachability Games and Weak Muller Games

Reachability Games

Given a finite game graph $G = (Q, Q_A, E)$, $F \subseteq Q$ with the winning condition

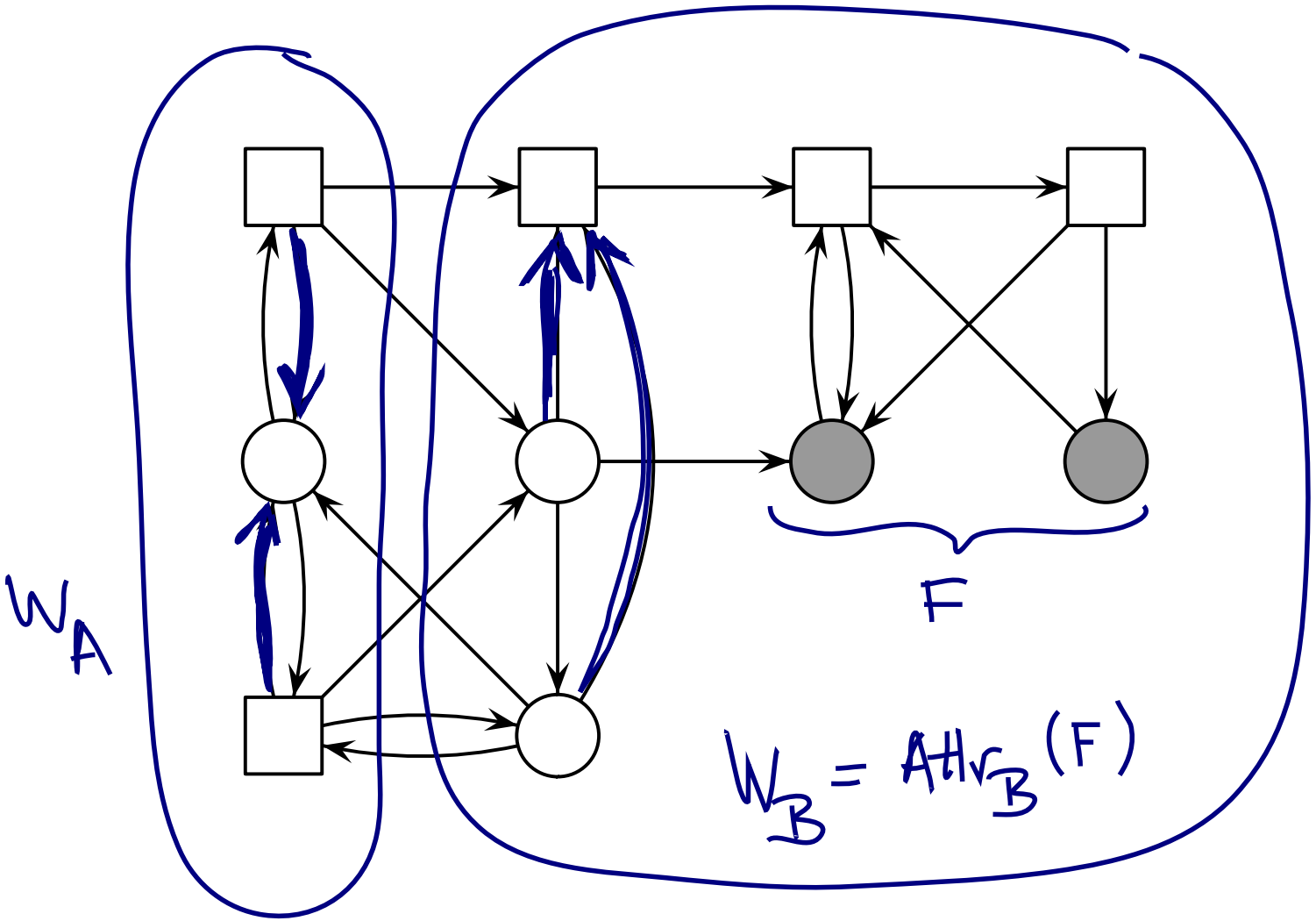
player B wins $\rho :\Leftrightarrow \exists i\ \rho(i) \in F$

Then the winning regions W_A, W_B of players A and B are computable, as well as corresponding positional winning strategies.

Proof: **Define**

$$\mathrm{Attr}_B^i(F) := \{q \in Q \mid \text{from } q \text{ player B can force a visit of } F \text{ in} \leq i \text{ moves}\}$$

Example



Computing the Attractor

Inductive construction of $\mathbf{Attr}_B^i(F)$:

$$\begin{aligned}\mathbf{Attr}_B^0(F) &= F,\\ \mathbf{Attr}_B^{i+1}(F) &= \mathbf{Attr}_B^i(F)\\ &\quad \cup \{q \in Q_B \mid \exists (q,r) \in E : r \in \mathbf{Attr}_B^i(F)\}\\ &\quad \cup \{q \in Q_A \mid \forall (q,r) \in E : r \in \mathbf{Attr}_B^i(F)\}\end{aligned}$$

Then

$$\mathbf{Attr}_B^0(F) \subseteq \mathbf{Attr}_B^1(F) \subseteq \mathbf{Attr}_B^2(F) \subseteq \ldots$$

Since Q is finite, for some $l \leq |Q|$: $\mathbf{Attr}_0^l(F) = \mathbf{Attr}_0^{l+1}(F)$

so the sequence becomes stationary at
$\mathbf{Attr}_B(F) := \bigcup_{i=0}^{|Q|} \mathbf{Attr}_B^i(F)$

Easy: $W_B = \mathbf{Attr}_B(F)$ and $W_A = Q \setminus \mathbf{Attr}_B(F)$

Weak Muller Games

The winning condition for player B (on the play ρ) is given by a family $\mathcal{F}$ of state-sets,

and B wins play ρ if $\mathrm{Occ}(\rho) \in \mathcal{F}$

Remark: Weak Muller conditions coincide with the Boolean combinations of reachability conditions.

One also speaks of Staiger-Wagner games (or obligation games).

The Idea of the Construction

We shall see: As memory in a finite-state winning strategy, the set of previously visited states suffices.

Given a weak Muller game over $G = (Q, Q_A, E)$ with $\mathcal{F} = \{F_1, \ldots, F_k\}$; $F_i \subseteq Q$

and the winning condition

- **player B wins $\rho \Leftrightarrow \mathrm{Occ}(\rho) \in \mathcal{F}$**

we define a new game over $2^Q \times Q$ such that a play ρ over Q induces a play ρ' over $2^Q \times Q$.

Record of Visited States

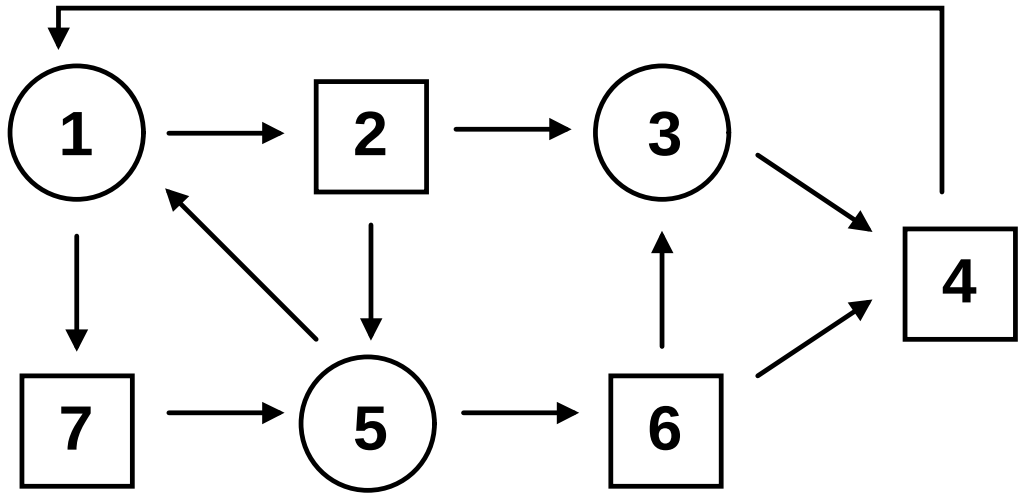
Construction:

- Instead of $\rho(0)$ take $\rho'(0) := (\{\emptyset, \rho(0))$
- If the play arrives at (P, p) and over Q a move $p \longrightarrow q$ is taken, now take $(P, p) \longrightarrow (P \cup \{p\}, q)$ in the new game.

Memory component records the previously visited Q-states.

Example

Game graph G:



Winning condition (for player 0): $\mathrm{Occ}(\rho) \supseteq \{2, 6\}$

ρ : 1, 2, 5, ...

ρ' : $(\emptyset, 1)$ $(\{1\}, 2)$ $(\{1, 2\}, 5)$ ...

General Observation

If $\mathrm{Occ}(\rho) = F$, then ρ' in its memory component reaches somewhere the value F (which then stays forever).

Define a coloring of states (P, q) by the memory component P

$$c((P,q)) = \begin{cases} 2 \cdot |P| & \text{for } P \in \mathcal{F} \\ 2 \cdot |P| - 1 & \text{for } P \notin \mathcal{F} \end{cases}$$

Then during the play ρ' the colors increase and finally stay

- at even color if $P \in \mathcal{F}$
- at odd color if $P \notin \mathcal{F}$

We obtain a “weak parity game”.

We shall show: In weak parity games, positional strategies suffice!

Idea of Game Reduction

We transform a game (G, φ) into a game (G', φ') such that

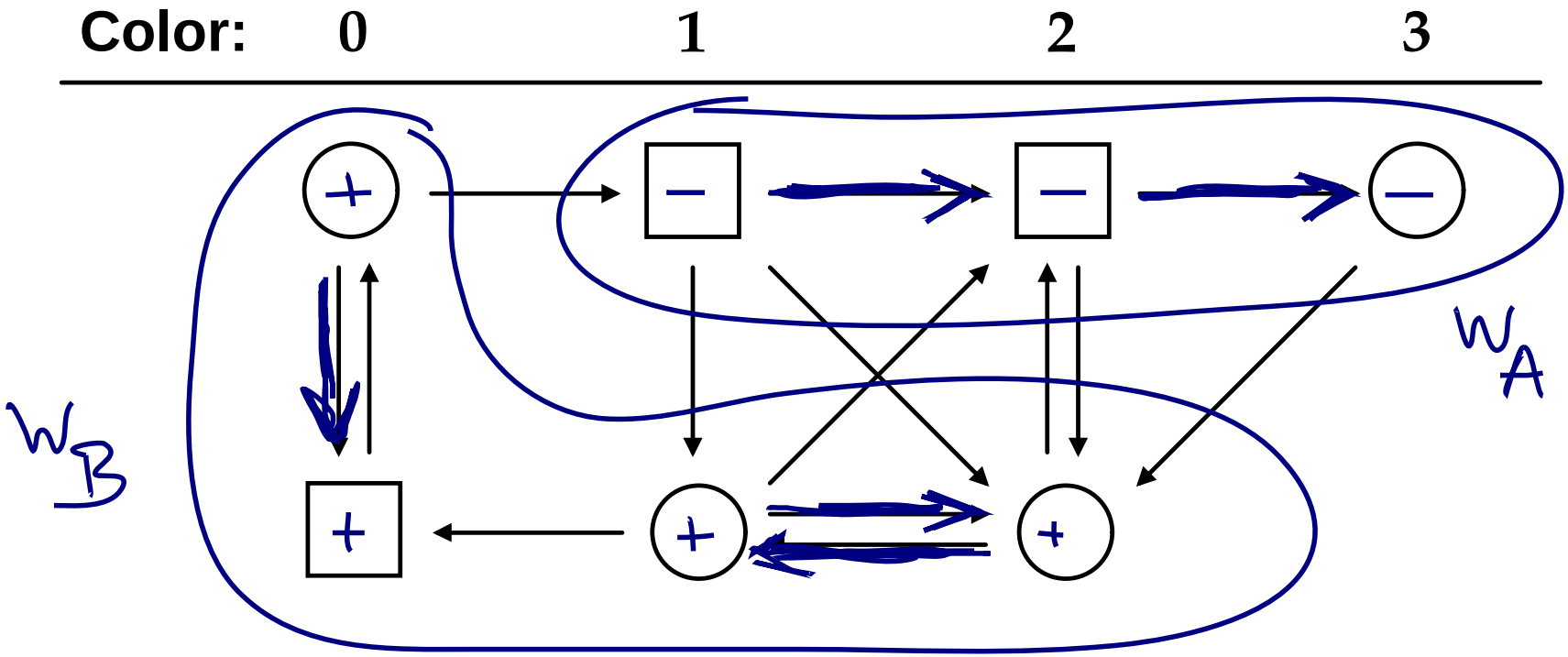
- **G' is larger than G**
- **φ' is "simpler" than φ**
- **from a solution of (G', φ') (using positional winning strategies) we can construct a solution of (G, φ) (with finite-state winning strategies)**

Weak Parity Games

A **weak parity game** is a pair $\Gamma = (G, \varphi)$ where

- the game graph $G = (Q, Q_A, E)$ is equipped with a coloring $c : Q \to \{0, \ldots, k\}$,
- the winning condition φ is the weak parity condition: player B wins ρ iff the maximal color occurring in $c(\rho)$ is even:
  $\max(\mathrm{Occ}(c(\rho)))$ is even

Example



- From the vertices marked $+$, player 0 has a winning strategy.
- From the vertices marked $-$, player 1 has a winning strategy.

The winning strategies are positional.

Theorem on Weak Parity Games

Theorem

For a weak parity game one can compute the winning regions W_A, W_B and also construct corresponding positional winning strategies.

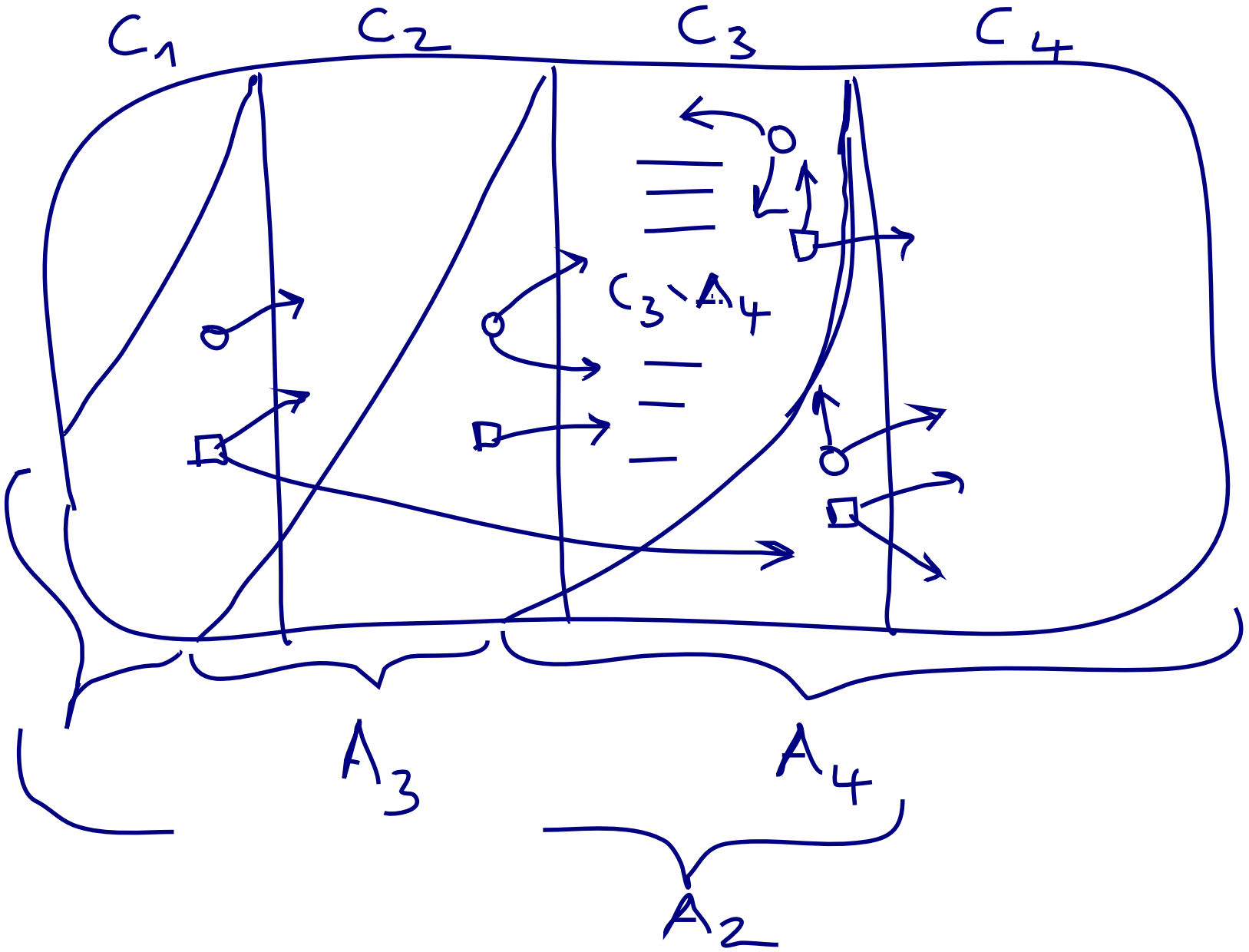
Proof: **Let $G = (Q, Q_A, E)$ be a game graph, $c : Q \to \{1, \ldots, k\}$ a coloring (w.l.o.g. k even).**

Set $C_i = \{q \in Q \mid c(q) = i\}$

First steps:
We first compute $A_k := \mathrm{Attr}_B(C_k)$; clearly from here player B wins.

Then we compute $\mathrm{Attr}_A(C_{k-1} \setminus A_k)$; from here player A wins.

General Construction: Figure



General Construction: Definitions

Aim: Compute larger and larger sets $A_k, A_{k-1}, \ldots, A_1$ such that $A_1 = W_A$ and $A_2 = W_B$

Let G_i be the game graph restricted to $Q \setminus (A_{i+1} \cup \ldots \cup A_k)$.

$\mathbf{Attr}_B^{G_i}(M)$ is the B-attractor of M in the subgraph induced by G_i, similarly for $\mathbf{Attr}_A^{G_i}(M)$

$$A_k := \mathbf{Attr}_B(C_k)$$
$$A_{k-1} := \mathbf{Attr}_A(C_{k-1} \setminus A_k)$$

for $i < k-1$:

$$A_i := \begin{cases} \mathbf{Attr}_B((C_i \setminus A_{i+1}) \cup A_{i+2}) & \text{if } i \text{ even} \\ \mathbf{Attr}_A((C_i \setminus A_{i+1}) \cup A_{i+2}) & \text{if } i \text{ odd} \end{cases}$$

Back to Weak Muller Games

Given a weak Muller game over Q

construct the weak parity game over $2^Q \times Q$

where transitions realize the "visited states update" in the memory component.

Each play over ρ induces a play ρ' over $2^Q \times Q$

such that

ρ satisfies the weak Muller condition iff ρ' satisfies the weak parity condition.

Consequence: A positional winning strategy over 2^Q induces a finite-state winning strategy over Q.

Game Simulation

The game (G, φ) is simulated by the game (G', φ') if in case $G = (Q, Q_A, E)$

there is a finite set S and two functions $f : q \mapsto (s_0, q)$ and $\delta : S \times Q \to S$ such that

- $G' = (S \times Q, S \times Q_A, E')$
- E' **has the edges** $((s, p), (\delta(s, p), q)$ **where** $(p, q) \in E$

such that

a play ρ over G satisfies φ
iff the play ρ' over G' induced by ρ via f and δ satisfies φ'.

Note: f supplies the initialization of memory and δ the update.

Application of Game Simulation

Suppose (G,φ) is simulated by (G',φ') via the finite set S and the functions f,δ.

Given a positional winning strategy for player B in (G',φ') from $f(q)$,

one can construct a finite-state winning strategy for player B in (G,φ) from q

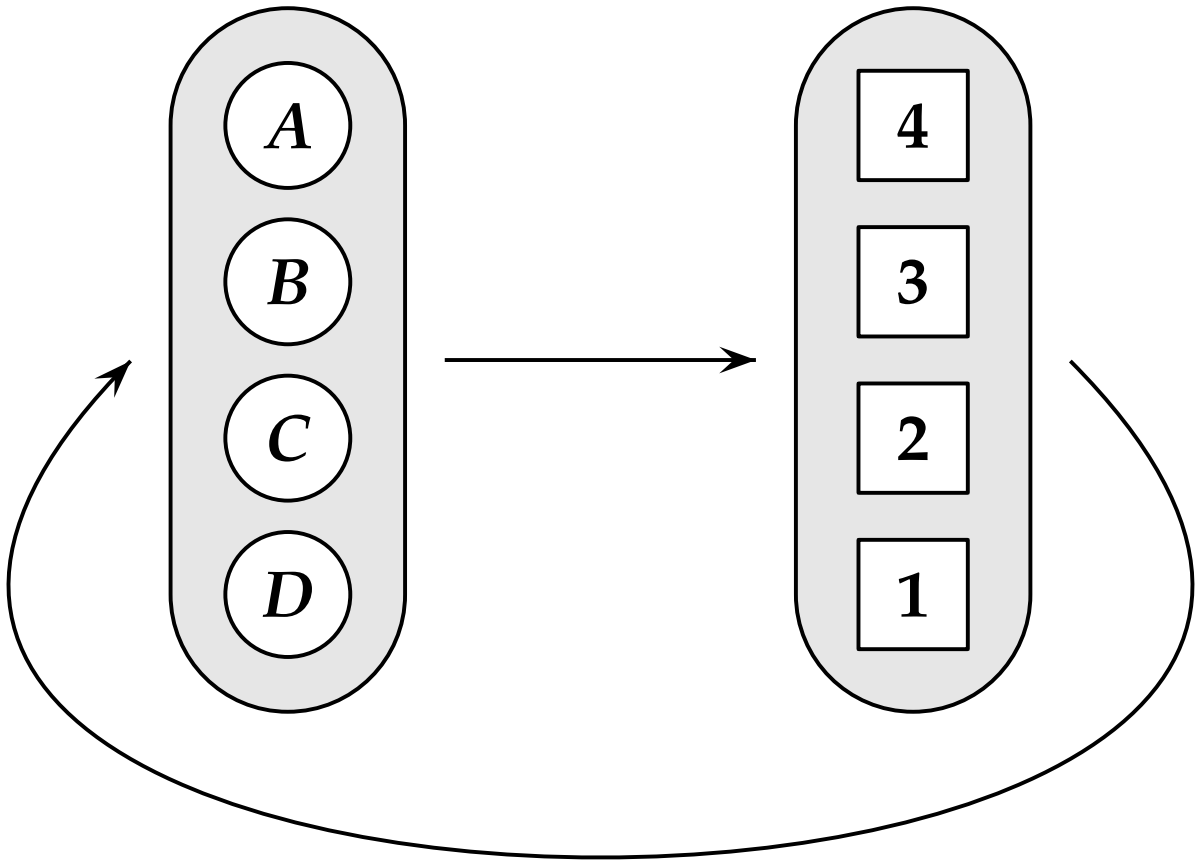
Take the strategy automaton with state-set S and initial state s_0 (assuming $f(q) = (s_0, q)$), use δ for memory update, and the given positional strategy for transition choice.

Part 4

Muller Games and Parity Games

How to Win Muller Games: The DJW Game

invented by Dziembowski, Jurdzinski and Walukiewicz (1997)



Winning condition:

$$|\mathrm{Inf}(\rho) \cap \{A, B, C, D\}| \;=\; \max(\mathrm{Inf}(\rho) \cap \{1, 2, 3, 4\})$$

Formal Definition of Muller Game

Player B wins ρ iff

the number of vertices A, B, C, D visited infinitely often

= highest number of Q_1 visited infinitely often

Formally take all vertex sets where the greatest number present is the same as the number of letter vertices.

Data structure for player B: Keep record of last visited (letter-) states, and while updating underline the position from where the current state was taken.

Latest Appearance Record

Visited letter	LAR
A	$ABCD$
C	$CA\underline{B}D$
C	$\underline{C}ABD$
D	$DCA\underline{B}$
B	$BDC\underline{A}$
D	$D\underline{B}CA$
C	$CD\underline{B}A$
D	$D\underline{C}BA$
D	$\underline{D}CBA$

Example Scenario

Assume the states C and D are repeated infinitely often.

Then:

- **the states A and B eventually arrive at the last two positions and are not touched any more; so finally underlinings appear at most on positions 1 and 2**
- **position 2 is underlined again and again; if only position 1 is underlined from some point onwards, only the same letter would be chosen from there onwards (and not two states C and D as assumed)**

Solution of the DJW-Game

LAR-strategy for player B:

During play, update and use the LAR as follows:

- **shift the current letter vertex to the front**
  underline the position from where the current letter was taken
- **move to the number vertex given by underlined position**

These are the two items performed by the strategy:

- **update of memory**
- **choice of next step (“output”)**

Result: Finite-state winning strategy with $n! \cdot n$ states for a game graph with $2n$ vertices

Analyzing the Winning Strategy

Call the underlined position the hit.

The states of a LAR up to the hit are called the recent states.

Consider the maximal hit $h_{\max}$ occurring infinitely often.

From some time onwards, only hits $\leq h_{\max}$ occur.

From this time, the recent states for $h_{\max}$ will always form the same set (but maybe listed in different order).

This set coincides with $\mathrm{Inf}(\rho)$

So $\mathrm{Inf}(\rho)$ is detectable from the sequence of LAR's

$\mathrm{Inf}(\rho)$ is the set of recent states occurring eventually for the highest hit position that occurs infinitely often.

In the DJW game, this was $\{C, D\}$, for $h_{\max} = 2$.

Introducing the Parity Condition

The Muller winning condition amounts to the following:

The set recent states which eventually is assumed for $h_{\max}$ belongs to $\mathcal{F}$.

Merge the information about hit value h and status of recent states by attaching a color to the LAR:

- **color $2h$ if recent states form set in $\mathcal{F}$**
- **color $2h-1$ otherwise**

So the Muller winning condition says:

The highest LAR-color occurring infinitely often is even

We abstract from the LAR structure, call a LAR just ”state“ and keep its color as the only information.

Parity Condition

We assume a coloring $c : Q \to \{1, \ldots, k\}$ of the game graph.

A play $\rho \in Q^\omega$ satisfies the parity condition iff the maximal color occurring infinitely often in ρ is even.

The parity condition says:

$$\bigvee_{j \text{ even}} (\exists^\omega i : c(\rho(i)) = j \ \wedge \neg \exists^\omega i : c(\rho(i)) > j)$$

A parity game is given by a game graph with finite coloring and the parity condition as winning condition for player B.

The LAR structure allows us to simulate a Muller game by a parity game.

The memory component now contains LAR's instead of sets.

From Muller to Parity Games

We summarize the lin between Muller games an parity games:

Theorem (Simulation Theorem):

For a game graph $G = (Q, Q_A, E)$ and Muller winning condition w.r.t. the set $\mathcal{F} \subseteq 2^Q$),

there is a parity game over a graph $G' = (Q', Q'_A, E')$ with a coloring c of Q'
such that a play ρ over G induces a play ρ' over G' and ρ satisfies the Muller condition iff ρ' satisfies the parity condition.

Proof: **Assume $Q = \{1, \ldots, n\}$.**
Define $Q' := Q \times \mathrm{LAR}(Q)$ ad define E' according to E with the respective LAR-update.

Application of Game Reduction

We shall show: The parity game over G' can be solved, with a positional winning strategy.

Consequence:

If player B wins over G' from $f(q)$ by a positional winning strategy, then player B wins the Muller game over G from q by a finite-state strategy.

(State $f(q)$ gives q with the initial LAR in G')

Just use the latest appearance records as states in a strategy automaton.

We now solve parity games by positional winning strategies.

Parity Games: Main Result

Theorem (Emerson-Jutla, Mostowski 1991)

- **Parity games are determined (i.e., each vertex belongs to W_A or W_B), and the winner from a given vertex has a positional winning strategy.**
- **Over finite graphs, the winning regions and winning strategies of the two players can be computed in (at most) exponential time in the number of vertices of the game graph.**

Whether polynomial time suffices in the second item, is (probably) a major open problem.

Positional Determinacy of Parity Games

here for finite G

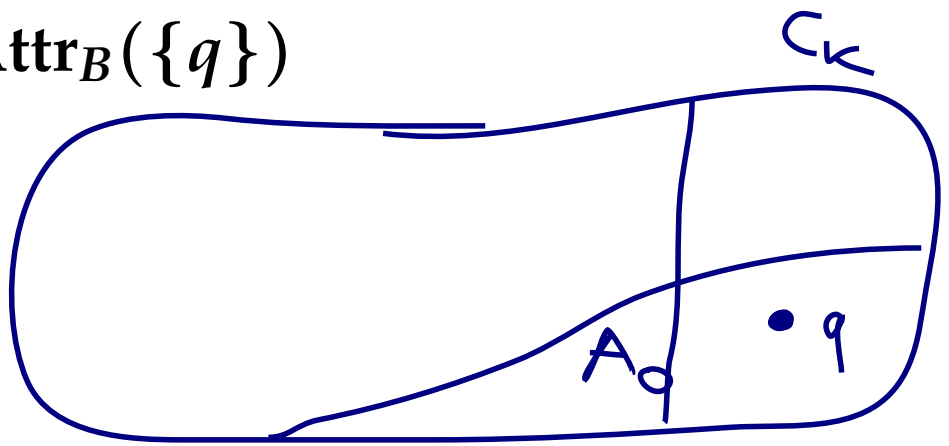
Given $G = (Q, Q_0, E)$ with coloring $c : Q \to \{0, \ldots, k\}$

We proceed by induction on the number of vertices of G

In the induction step assume that the maximal color k is even (otherwise switch the roles of players A and B below).

Let q be a vertex of the highest (even) color k.

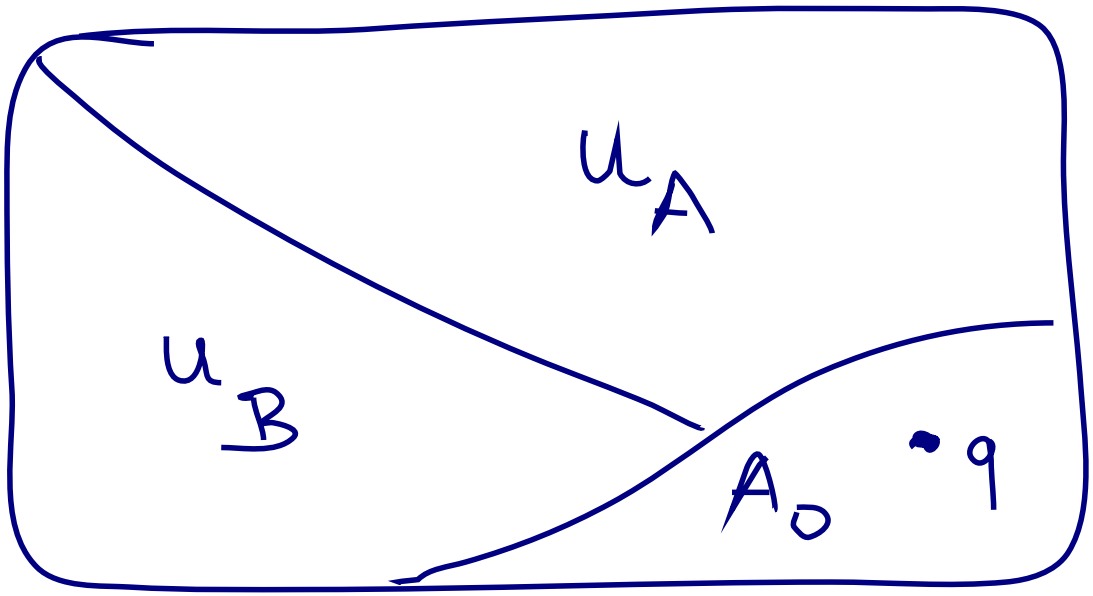
Define $A_0 = \mathrm{Attr}_B(\{q\})$



Induction Step

The set $Q \setminus A_0$ defines a subgame.

Induction hypothesis ensures the partition of $Q \setminus A_0$ into the winning regions U_A, U_B of the two players (with corresponding positional winning strategies)



A Case Distinction

Case 1: From q, player B can ensure to be in $U_B \cup A_0$ in the next step.

If not Case 1 and $q \in Q_B$: All transitions go to U_A

If not Case 1 and $q \in Q_A$: Some transition goes to U_A

So the complement of Case 1 is the following:

Case 2: From q, player 1 can ensure to be in U_A in one step.

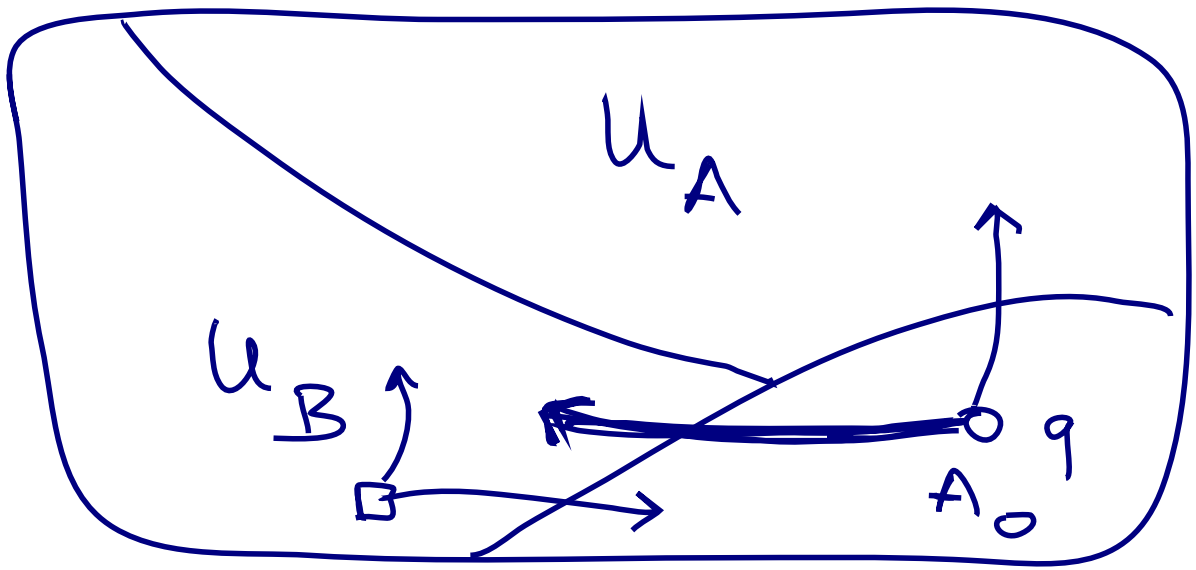
Illustration:

Induction Step Completed: Case 1

In Case 1 one can show $W_B = U_B \cup \mathrm{Attr}_B(\{q\})$ and $W_A = U_A$.

A play in the first set either remains in U_B from some point onwards, whence Player B wins by induction hypothesis,

or it visits (by choice of player A) the attractor and hence q again and again, so that player B wins by seeing the highest color (even!) repeatedly.

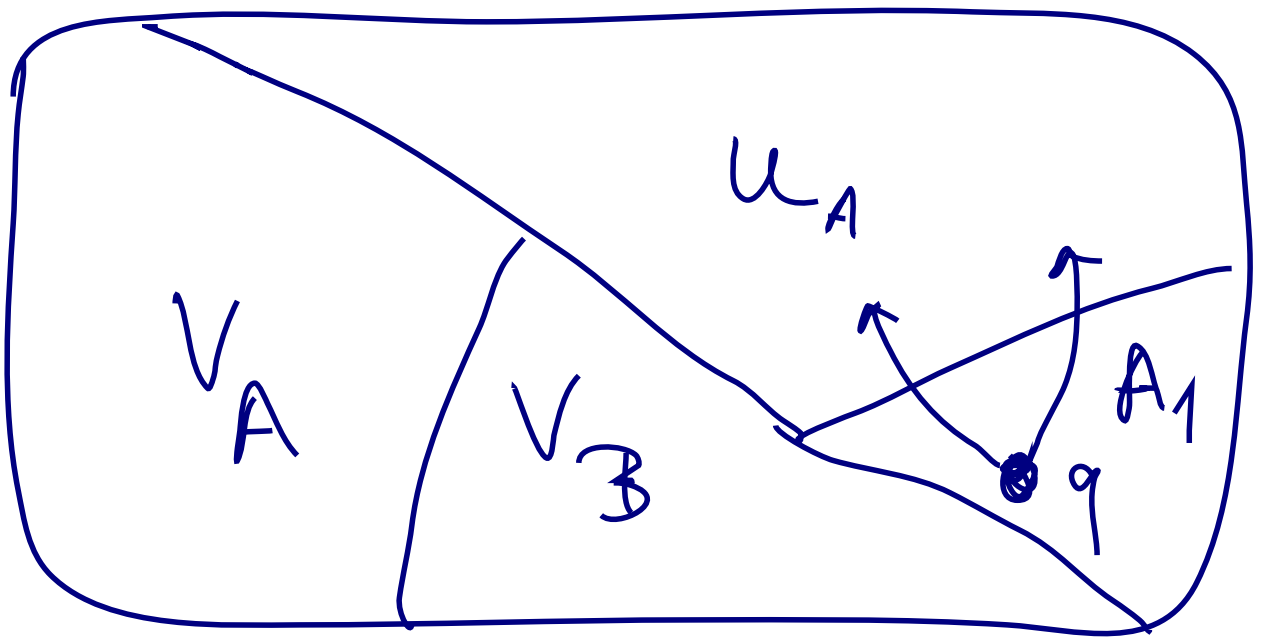


Case 2

In Case 2 one forms $A_1 = \mathrm{Attr}_A(U_A \cup \{q\})$ and applies induction hypothesis to the domain $Q \setminus A_1$

Obtain a partition of this domain into V_A, V_B.

Then $W_B = V_B$ and $W_A = V_A \cup A_1$



Towards an Algorithmic Solution

We use a nondeterministic algorithm:

1. **Guess W_A and W_B and positional strategies given by edge sets: E_A with one out-edge from Q_A-vertices, and E_B with one out-edge from Q_B-vertices.**
2. **Check that E_A is a uniform winning strategy from each $q \in W_A$ and that E_B is a uniform winning strategy from each $q \in W_B$**

Step 1: is done in nondeterministic polynomial time (or by exhaustive search of all $2^{|Q|} \cdot 2^{|Q|^2}$ possibilities)

Step 2: Check whether a given positional strategy is a winning strategy for player 0 from q.

Step 2: Checking Strategies

Remark: For a fixed positional strategy f of player B one can decide in polynomial time for any $q \in Q$, whether f is a winning strategy from q

Proof:

Consider graph $G(f)$ given by the positional strategy f:

Each Q_B-vertex has precisely one out-edge given by f. We have essentially a one-player game (of player 1).

Check whether in $G(f)$ there is a path from q to a loop whose highest color is odd.

For this, check for reachable SCC's (strongly connected components in the restriction of G_f to colors $\leq m$ (for odd m).

Decision Problem “Parity Game”

Given: A finite game graph G with coloring, $q \in Q$

Question: Does player B win the parity game from q?
(Short: “$q \in W_B$ in the corresponding parity game?”)

Theorem: The Problem “Parity Game” is in the complexity class NP $\cap$ co-NP

Proof: The above nondeterministic procedure shows that the problem is in NP.

It remains to show that the complementary problem
“$q \notin W_B$?” is also in NP.

This problem means “$q \in W_A$?”. It is solvable in the same way as “$q \in W_B$?”, hence in NP.

Intriguing: Open problem: Is ”parity game” in P?

Summary and Outlook

Given a game graph and a “regular” winning condition (formalized in LTL, in S1S, or by a nondeterministic Büchi automaton)

we can decide for any q whether B has a winning strategy, and – if yes – construct a finite automaton executing it.

Use game simulations:

- **from formulas to Muller games**
  (use transformation from logic to Büchi automata and determinization into Muller automata)
- **from Muller games to parity games**

Part 5

Infinite Game Graphs

Infinite-State Games: Pushdown Games

A pushdown game has as vertices the configurations of a pushdown automaton (of the form (p, w) consisting of control state and stack content)

For solution of reachability games use a classical result of Büchi (with new constructions):

Given a regular set C of configurations, the set $P(C)$ configurations from which one can reach C is again regular.

Extension to parity games and Muller games is possible (Walukiewicz).

Further extensions: higher-order pushdown systems

Beyond Pushdown Graphs: Recursive Graphs

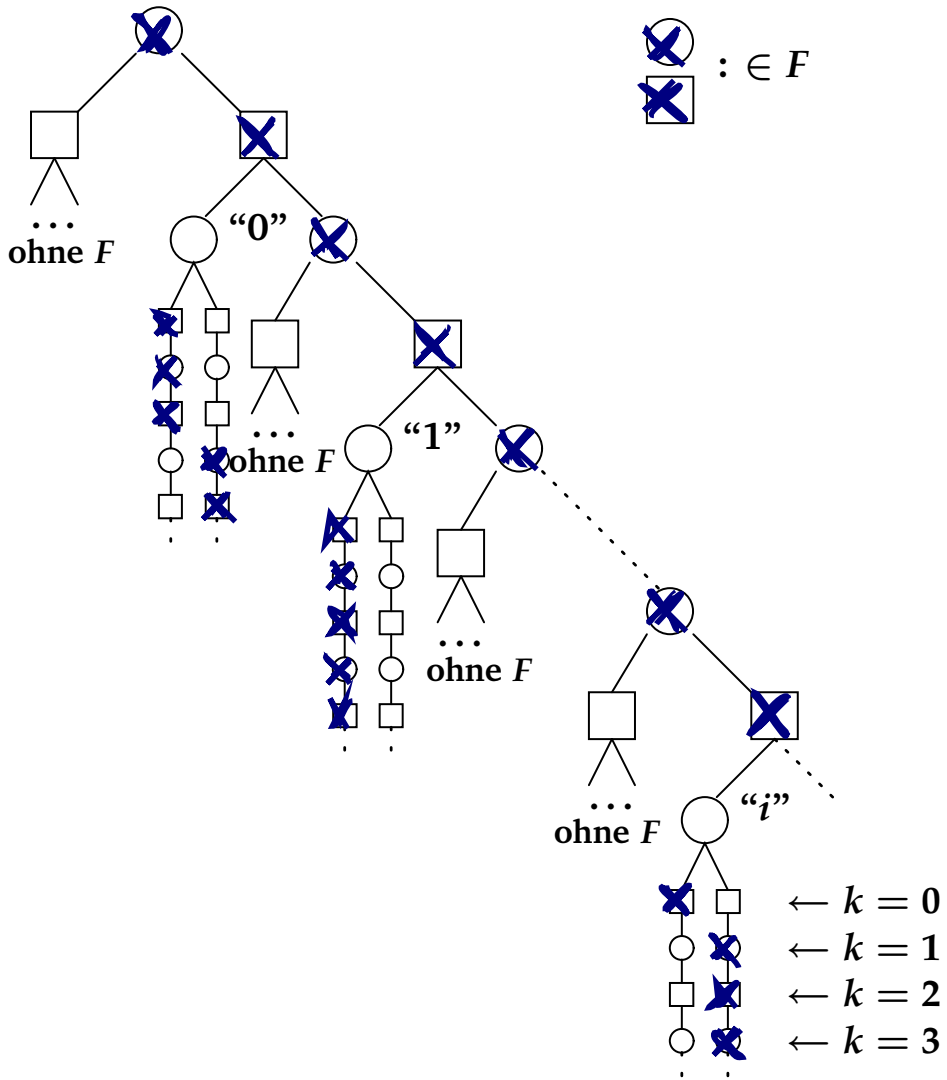
Open problem: Find classes of infinite graphs where interesting games are algorithmically solvable.

Recursive graphs (effectively presented graphs) are much too general.

Also one has to distinguish the effectiveness of the winning region from the effectiveness of the winning strategy.

We present an example.

A Recursive Infinite Graph



Winning Strategy is Not Computable

Convention: Below node i the membership in F switches at level k iff the Turing machine M_i stops on the empty tape at step k

Claim: Player B has (en a positional) winning strategy in the Büchi game over G_0 from the root, but he has no computable winning strategy.

Proof: Assume B has computable winning strategy f

Let $\mathcal{A}_f$ be an algorithm computing f

Use $\mathcal{A}_f$ to solve the halting problem for Turing machines:

Given TM M_i, consider the play where player 1 branches off the rightmost branch to vertex i

Apply $\mathcal{A}_f$ to this play prefix and consider the value “branch left”, respectively “branch right”.

Value is “branch right” iff M_i stops, started on empty tape.

Part 6

Sources

Some Sources

- **Section 6 of the survey:**
  W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, Handbook of Formal Languages, volume III, pages 389–455. Springer, New York, 1997.
- **Tutorial volume**
  E. Grädel, W. Thomas, Th. Wilke (eds.): Automata, Logics, and Infinite Games, Springer LNCS 2500.
- **Recordings of the course “Automata and Reactive Systems”, winter term 2002/03 at RWTH Aachen, accessible via**
  `www-i7.informatik.rwth-aachen.de/d/teaching/ws0203/ars/`