Lecture
Games in Automata and Logic

Winter 2026/2027

Prof. Dr. Karin Quaas

Institut für Informatik
Universität Leipzig

Lecture 1 - Church's Problem

Welcome

- Games in Automata and Logic is a course in the field of Theoretical Computer Science
- Bachelor Module *Spiele in Automatentheorie und Logik*, zz-xxx-yyy
- Prerequisites: Discrete Structures, Logics, Automata and Languages, and Computability/Complexity
- Lecture every Tuesday 3-5 pm in Felix-Klein-Hörsaal
- Exercises every Monday 11-1 pm in SG 3-13
- Written exam (60 Minutes)
- Teaching Language: English or German

How to approach this course?

- Attending the lectures and exercise is not necessary but recommended
- Lecture presentation, exercise solutions and extra proofs are in the respective Moodle course:
  `https://moodle2.uni-leipzig.de/course/view.php?id=62794`
- Please also bring some electronic device if possible (for electronic quizzes and peer instruction)
- Weekly exercises with exercise sheets that I recommend to do before the exercise
- Group works in lectures and exercises
- Bonus point policy:
  - In the exam, you can achieve 60 points
  - You can obtain 6 bonus points during the semester:
  - 6 mini tests (one bonus point each) scattered randomly over the semester
  - So please bring a pen!
- You can always ask questions!

Let's play

We play a game with two players, Alice and Eve. The rules of the game are:

- Alice picks a bit in $\{0, 1\}$
- Eve responds with a bit in $\{0, 1\}$
- this goes on forever

For example, a play of this game may look as follows:

Player \ Round	1	2	3	4	5	...
Alice	1	0	1	1	0	...
Eve	0	1	1	0	1	...

A play is an infinite sequence $(a_1, e_1)(a_2, e_2)(a_3, e_3)\ldots$ of bits

Winning Condition I

Eve wins a play $(a_1, e_1)(a_2, e_2)(a_3, e_3)\ldots \iff e_i = 1 - a_i$ for all $i \geq 1$.

Question: Does Eve have a winning strategy?

Answer: Yes, Eve wins if she plays according to the following program:

```
1: for every round i = 1, 2, 3, ... do
2:     read a_i
3:     if a_i = 0 then
4:         output e_i = 1
5:     else
6:         output e_i = 0
7:     end if
8: end for
```

This program runs forever and produces Eve's bit e_i in every round.
The resulting play satisfies the winning condition.

Winning Strategies

- A program computes a function
- A strategy for Eve can be seen a function of the form

$$f : (\{0,1\} \times \{0,1\})^* \to (\{0,1\} \to \{0,1\})$$

  where the input is the history of the play so far; the output is a function that, for the next Alice-bit, outputs some Eve-bit
- If the output Eve-bit generates an infinite play that satisfies the winning condition (no matter what Alice-bit), it is a winning strategy
- For winning condition I, the strategy is

$$f(h)(a) = 1 - a$$

  for all h and a, that is, we do not use the history h at all
- Such a strategy is called memoryless: it does not depend on the history h, it only depends on Alice's current input bit

Winning Condition II

Eve wins a play $(a_1, e_1)(a_2, e_2)(a_3, e_3)\ldots \iff e_i = a_{i-4}$ for all $i \geq 5$.

Question: Does Eve have a winning strategy?

Answer: Yes, Eve can win if she plays according to

```
1: for every round i = 1, 2, 3, ... do
2:     if i ≤ 4 then
3:         output e_i = 0
4:     else
5:         output e_i = a_{i-4}
6:     end if
7: end for
```

This strategy is not memoryless: to produce the next output, the program needs to remember the input bit from 4 rounds ago.

Winning Condition II

- For winning condition II, the strategy $f : (\{0,1\} \times \{0,1\})^* \to (\{0,1\} \to \{0,1\})$ is defined by

$$f(h)(a) = \begin{cases} 0 & \text{if } |h| \leq 4 \\ a_{i-4} & \text{if } |h| \geq 5 \end{cases}$$

  for every finite history $h = (a_1, e_1) \dots (a_n, e_n)$ and Alice-bit a.
- The strategy depends on h, but the memory needed is fixed
- We call such strategies finite-memory (or: finite-state): the strategy requires memory from the history but it is bounded by some fixed finite amount
- Otherwise a strategy has unbounded memory

Memoryless and Finite-Memory Strategies

- A strategy is memoryless (or positional) if its next move depends only on the current position of the game, and not on the history of the play.
- That is $f(h)(a) = f(h')(a)$ for all histories h and h'
- A strategy is finite-memory if it can be implemented using a fixed finite set of memory states.
- Thus, a finite-memory strategy may remember information about the history, but only through a finite (bounded) summary.

$$\text{memoryless} \subseteq \text{finite-memory} \subseteq \text{arbitrary strategies}$$

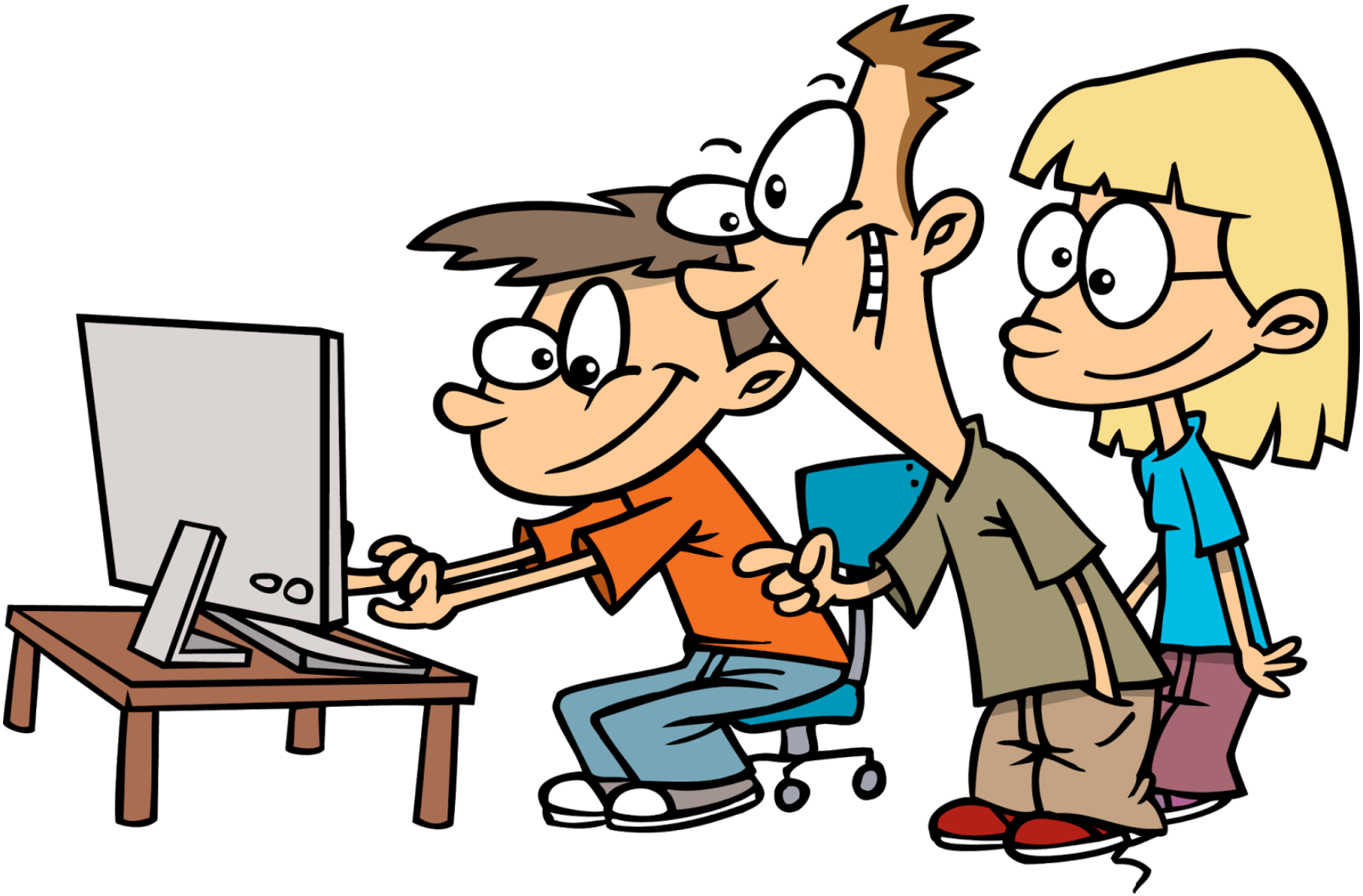
Winning Condition III

Eve wins a play $(a_1, e_1)(a_2, e_2)(a_3, e_3)\ldots \iff e_i = a_{i+1}$ for all $i \geq 1$.

Question: Does Eve have a winning strategy?

Answer: No. Eve must respond immediately, bit by bit, and cannot look into the future (no look-ahead).

Groupwork



Groupwork: Winning Condition IV

Eve wins a play $(a_1, e_1)(a_2, e_2)(a_3, e_3)\ldots \iff$ the following three conditions are satisfied:

1. **Safety:** for all $i \geq 1$, if $a_i = 1$ then $e_i = 1$,
2. **No two consecutive zeros:** there exists no $i \geq 1$ such that $e_i = e_{i+1} = 0$, and
3. **Infinitely often zeros:** if there are infinitely many $i \geq 1$ with $a_i = 0$, then there are infinitely many $i \geq 1$ with $e_i = 0$.

Question: Does Eve have a winning strategy?

Groupwork: Winning Condition IV - Solution

Answer: Yes. Eve can win if she responds according to

1: $e_0 = 0$
2: **for** every round $i = 1, 2, 3, \ldots$ **do**
3: read a_i
4: **if** $a_i = 1$ **then**
5: output $e_i = 1$
6: **else**
7: **if** $e_{i-1} = 0$ **then**
8: output $e_i = 1$
9: **else**
10: output $e_i = 0$
11: **end if**
12: **end if**
13: **end for**

Proof

1. **Safety:** for all $i \geq 1$, if $a_i = 1$ then $e_i = 1$: Lines 4 and 5.
2. **No two consecutive zeros:** there exists no $i \geq 1$ such that $e_i = e_{i+1} = 0$: Lines 6-11. Whenever $a_i = 0$, the strategy sets $e_i = 0$ only if $e_{i-1} = 1$. Hence two consecutive 0's in Eve's sequence are impossible.
3. **Infinitely often zeros:** suppose $i_1 < i_2 < i_3 < \ldots$ is an increasing sequence of indices such that $a_{i_j} = 0$. By Lines 6-11, we have $e_{i_j-1} = 0$ or $e_{i_j} = 0$ for all $j \geq 1$.

Winning Conditions as Specifications

A winning condition specifies which plays are winning for Eve.

- A play is an infinite word over

$$\{0,1\} \times \{0,1\}.$$

- Hence, a winning condition is a set

$$\Omega \subseteq \big(\{0,1\} \times \{0,1\}\big)^{\omega}.$$

- Equivalently, we can see Ω as a relation $\Omega \subseteq \{0,1\}^{\omega} \times \{0,1\}^{\omega}$ between input and output bit streams

Question: How can we describe such sets Ω?

Formalizing Specifications

- We would like to describe winning conditions by finite descriptions, such as logics and automata
- In Church's original formulation, specifications are given in monadic second order logic of one successor (S1S)
- S1S extends first-order logic (FO, for short) with second-order variables ranging over monadic predicates, and over which one can universally and existentially quantify
- An S1S formula φ defines a winning condition Ω_φ, that is, a relation between input and output streams
- There are other formalisms for expressing winning conditions including temporal logics and automata

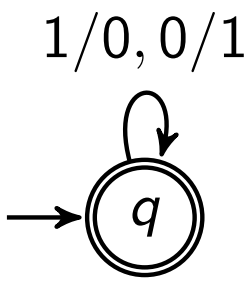
Formalizing Strategies

- A strategy is a function $f : (\{0,1\} \times \{0,1\})^* \to (\{0,1\} \to \{0,1\})$
- In Church's original problem formulation, such functions are restricted:
  - Eve's bit has to be produced without delay after receipt of Alice's bit, that is, e_i can only depend on the finite history $(a_1, e_1) \ldots (a_{i-1}, e_{i-1})$ and a_i
  - the computation of e_i must be realizable with a fixed finite memory (independent of i)
- A strategy can be represented by a finite-state transducer
- A finite-state transducer extends finite-state automaton:
  - it reads an input symbol,
  - changes its state, and
  - produces an output symbol
- The state stores the relevant information about the history.

Transducer Example Winning Condition I

Eve wins a play $(a_1, e_1)(a_2, e_2)(a_3, e_3)\ldots \iff e_i = 1 - a_i$ for all $i \geq 1$.

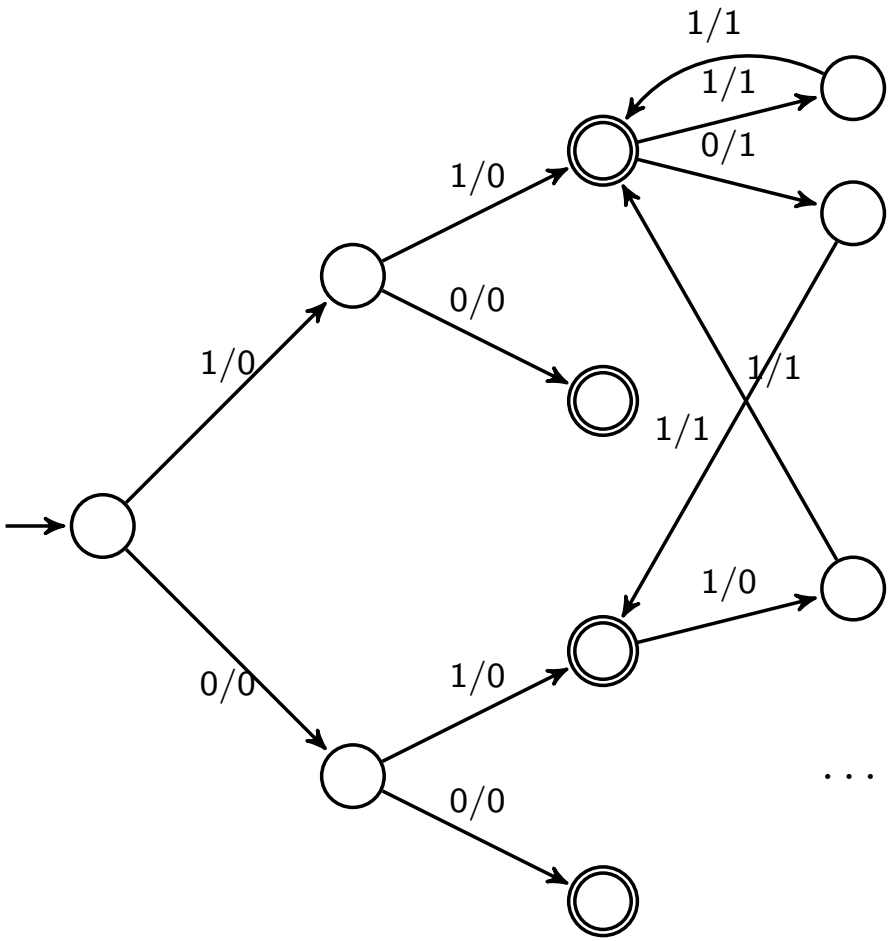
The following transducer computes Eve's winning strategy



Transducer Example (Simplified) Winning Condition II

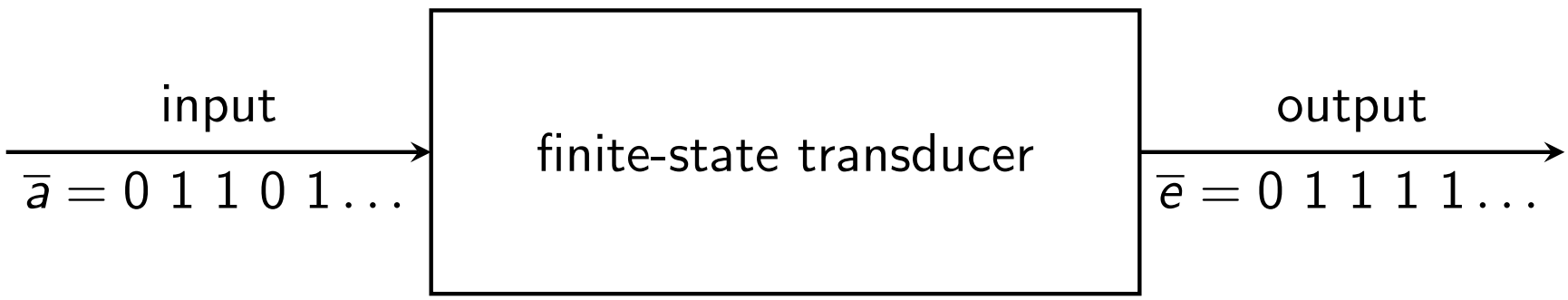
Eve wins a play $(a_1, e_1)(a_2, e_2)(a_3, e_3)\ldots \iff e_i = a_{i-2}$ for all $i \geq 3$.

The following (incomplete) transducer computes a winning strategy for Eve



Church's Problem

An infinite input bit stream $\overline{a}$ is to be transformed, bit by bit, into an infinite output bit stream $\overline{e}$ according to a given specification φ specifying the relation between $\overline{a}$ and $\overline{e}$

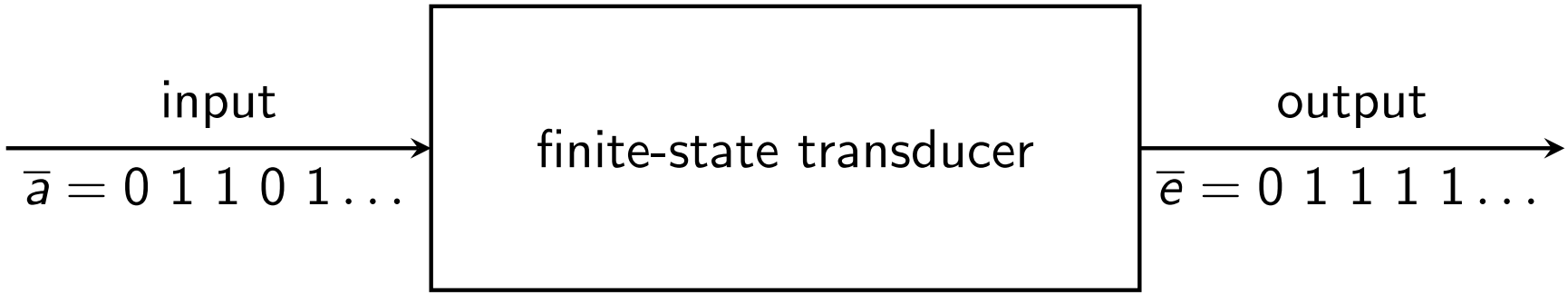


Church's problem:
Given a S1S-specification φ, can one automatically synthesize a finite-state transducer T that satisfies φ?

This is also known as synthesis problem

Church's Problem: Practical Relevance/Motivation

An infinite input bit stream $\overline{a}$ is to be transformed, bit by bit, into an infinite output bit stream $\overline{e}$ according to a given specification φ specifying the relation between $\overline{a}$ and $\overline{e}$



- input corresponds to the behaviour of the environment
- finite-state transducer corresponds to the controller
- controller must produce output for all kinds of behaviours of the environment

Alternative View to Church's Problem: Games

- We started with a game:
  - Alice produces the input stream $\overline{a} = 0\ 1\ 1\ 0\ 1\ldots$
  - Eve produces the output stream $\overline{e} = 0\ 1\ 1\ 1\ 1\ldots$
  - Eve wins if the resulting play satisfies the winning condition Ω_φ
- We can view Church's problem as a question about winning strategies:

  Does Eve have a winning strategy?
- A winning strategy tells us how to transform every possible input stream into an output stream satisfying the specification
- Asking whether Eve has a winning strategy is another way of asking whether one can synthesize a transducer

Historical Background

- **Church (1957):** formulates the synthesis problem: construct a finite-state transducer realizing a given specification.
- **Büchi (1960s):** develops the connection between S1S and automata on infinite words.

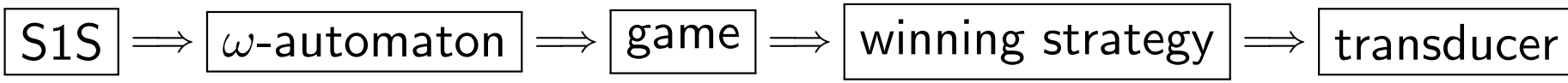
$$\boxed{\text{S1S}} \Longrightarrow \boxed{\omega\text{-automaton}}$$

- **Büchi and Landweber (1969):** solve Church's problem for S1S specifications.

$$\boxed{\text{game}} \Longrightarrow \boxed{\text{winning strategy}} \Longrightarrow \boxed{\text{transducer}}$$

Büchi-Landweber Theorem

- Main ideas of Büchi-Landweber:
  - convert an S1S formula φ into an equivalent ω-automaton $\mathcal{A}_\varphi$
  - view $\mathcal{A}_\varphi$ as a 2-player game between Alice and Eve
  - decide which player has a winning strategy
  - if Eve wins, construct a finite-state winning strategy, represented by a finite-state transducer.



Another Solution: Rabin

- **Rabin (1972)** gave a different solution to Church's problem.
- Instead of viewing the problem as a game, he uses automata on infinite trees.
- The key idea:
  - all possible input streams form a tree;
  - a strategy selects an output for every possible input history;
  - hence, a strategy can be represented as a labelled infinite tree.
- The specification can be transformed into a tree automaton accepting exactly the trees that represent strategies satisfying the specification.
- The existence of a suitable strategy can then be reduced to the non-emptiness problem for tree automata.



Why is Church's Problem so Beautiful?

- We start with a very simple question:

  Can we transform every input stream into an output stream satisfying a specification?

- The specification may be given in a very expressive logic (S1S).
- The specification describes infinitely many behaviours
- Yet, if a solution exists, there is a finite-state solution.
- And this solution can be found automatically.
- The proof brings together three seemingly different worlds:

Logic Automata Games

A logical specification can be turned into an algorithmic controller.

Schedule

	Monday 11-13	Tuesday 15-17
12.10.-18.10.		Lecture 1: Church's Problem
19.10.-25.10.	Exercise 1	Lecture 2: Games on Finite Graphs
26.10.-1.11.	Exercise 2	Lecture 3: Reachability Games
2.11.-8.11.	Exercise 3	Lecture 4: Safety / Büchi Games
9.11.-15.11.	Exercise 4	Lecture 5: Büchi / co-Büchi / generalized Büchi
16.11.-22.11.	No Exercise	No Lecture
23.11.-29.11.	Exercise 5	Lecture 6: Parity Games
30.11.-6.12.	Exercise 6	Lecture 7: Parity Games
7.12.-13.12.	Exercise 7	Lecture 8: S1S
14.12.-20.12.	Exercise 8	Lecture 9:
4.1.-10.1.	Exercise 10	Lecture 11:
11.1.-17.1.	Exercise 11	Lecture 12:
18.1.-24.1.	Exercise 12	Lecture 13:
25.1.-31.1.	Exercise 13 :	Lecture 14:
1.2.-7.2.	Exercise 14	Written Exam ?