Games with delay for automaton synthesis

Christof Löding

RWTH Aachen University, Germany

GandALF
Borca di Cadore, Dolomites, Italy
August 29–31, 2013

Outline

1. Automaton Synthesis from Specifications
   - Classical Setting
   - Delaying the Output
   - Beyond Finite Automata

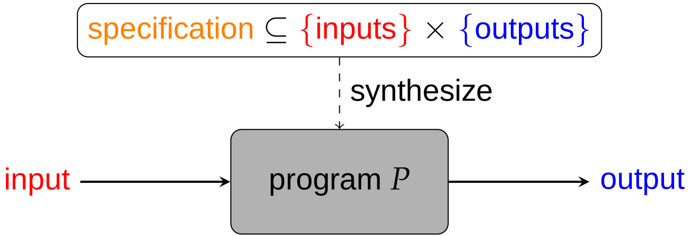
2. Synthesis of Lookahead Delegators

Outline

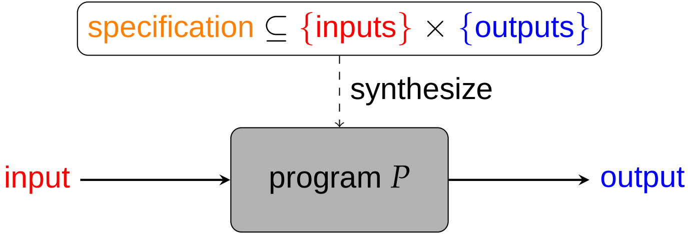
Motivation: Realizing Specifications

$$\boxed{\text{specification} \subseteq \{\text{inputs}\} \times \{\text{outputs}\}}$$

Motivation: Realizing Specifications



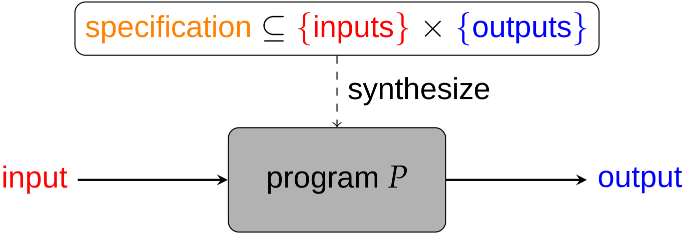
Motivation: Realizing Specifications



input

output

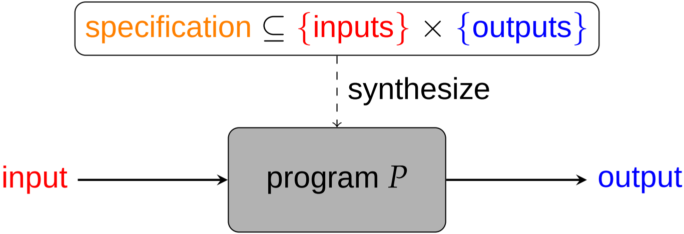
Motivation: Realizing Specifications



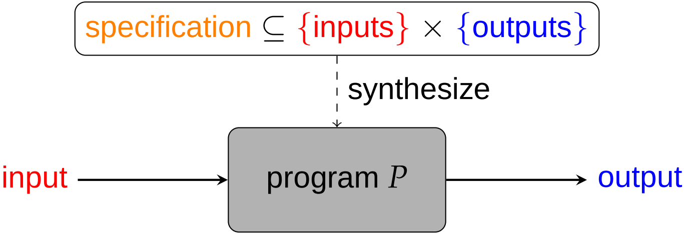
input a_0

output

Motivation: Realizing Specifications



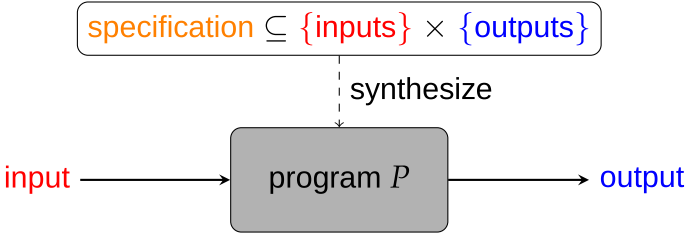
Motivation: Realizing Specifications



input $a_0\ a_1$

output b_0

Motivation: Realizing Specifications

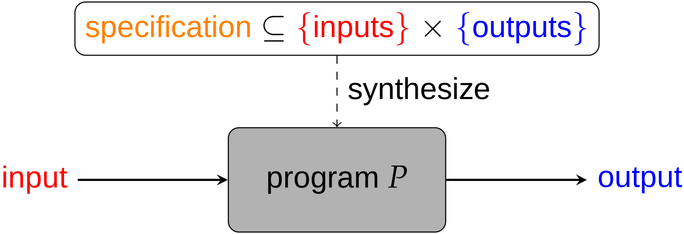


input $\boxed{a_0\ a_1}$

output $b_0\ b_1$

P

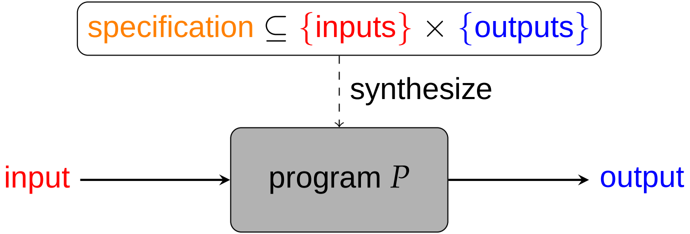
Motivation: Realizing Specifications



input $a_0\ a_1\ a_2$

output $b_0\ b_1$

Motivation: Realizing Specifications

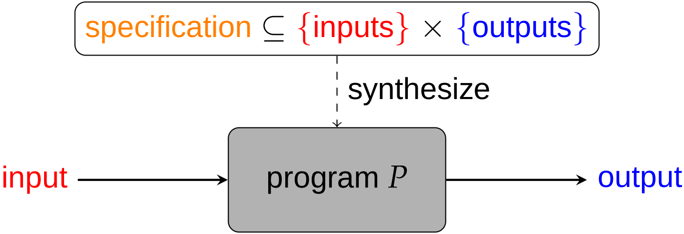


input $\boxed{a_0 \; a_1 \; a_2}$

output $b_0 \; b_1 \; b_2$

P

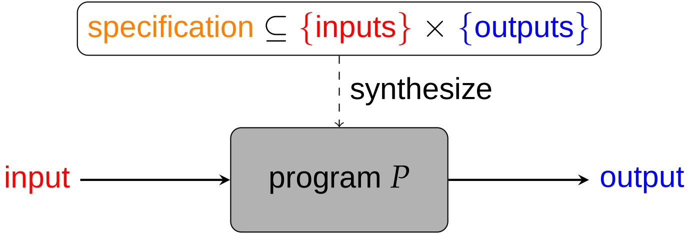
Motivation: Realizing Specifications



input $a_0\ a_1\ a_2\ a_3$

output $b_0\ b_1\ b_2$

Motivation: Realizing Specifications

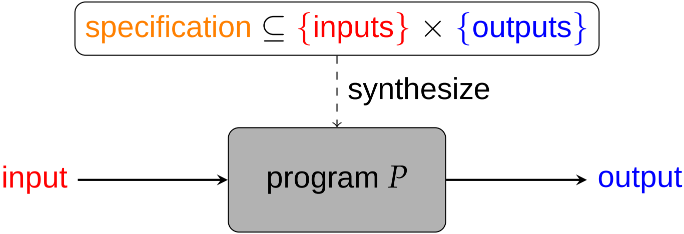


input $a_0\ a_1\ a_2\ a_3$

output $b_0\ b_1\ b_2\ b_3$

P

Motivation: Realizing Specifications



input $a_0\ a_1\ a_2\ a_3\ a_4$

output $b_0\ b_1\ b_2\ b_3$

Motivation: Realizing Specifications

specification $\subseteq$ {inputs} $\times$ {outputs}

synthesize

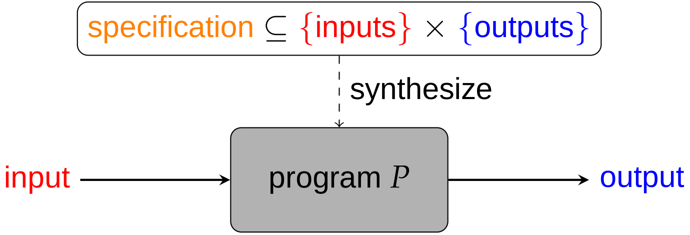
input $\longrightarrow$ program P $\longrightarrow$ output

input $\boxed{a_0\ a_1\ a_2\ a_3\ a_4}$

P

output $b_0\ b_1\ b_2\ b_3\ b_4$

Motivation: Realizing Specifications



input $\quad a_0\ a_1\ a_2\ a_3\ a_4 \cdots$

output $\quad b_0\ b_1\ b_2\ b_3\ b_4 \cdots$

$\in$ specification

Viewed as a Game

Two players I (input) and II (output) play letters from finite alphabets (I and J) in alternation:

I

II

Viewed as a Game

Two players I (input) and II (output) play letters from finite alphabets (I and J) in alternation:

I	a_0
II	

Viewed as a Game

Two players I (input) and II (output) play letters from finite alphabets (I and J) in alternation:

I	a_0
II	b_0

Viewed as a Game

Two players I (input) and II (output) play letters from finite alphabets (I and J) in alternation:

I	$a_0\ a_1$
II	b_0

Viewed as a Game

Two players I (input) and II (output) play letters from finite alphabets (I and J) in alternation:

I	$a_0 \; a_1$
II	$b_0 \; b_1$

Viewed as a Game

Two players I (input) and II (output) play letters from finite alphabets (I and J) in alternation:

I	$a_0\ a_1\ a_2$
II	$b_0\ b_1$

Viewed as a Game

Two players I (input) and II (output) play letters from finite alphabets (I and J) in alternation:

I	$a_0 \; a_1 \; a_2$
II	$b_0 \; b_1 \; b_2$

Viewed as a Game

Two players I (input) and II (output) play letters from finite alphabets (I and J) in alternation:

I $\quad a_0 \ a_1 \ a_2 \cdots$

II $\quad b_0 \ b_1 \ b_2$

Viewed as a Game

Two players I (input) and II (output) play letters from finite alphabets (I and J) in alternation:

I $\quad a_0\ a_1\ a_2 \cdots$

II $\quad b_0\ b_1\ b_2 \cdots$

Viewed as a Game

Two players I (input) and II (output) play letters from finite alphabets (I and J) in alternation:

I	$a_0 \; a_1 \; a_2 \cdots$
II	$b_0 \; b_1 \; b_2 \cdots$

Winning condition: II wins if the pair $(a_0 a_1 a_2 \cdots, b_0 b_1 b_2 \cdots)$ is in the relation given by the specification.

Viewed as a Game

Two players I (input) and II (output) play letters from finite alphabets (I and J) in alternation:

I	$a_0\ a_1\ a_2 \cdots$
II	$b_0\ b_1\ b_2 \cdots$

Winning condition: II wins if the pair $(a_0a_1a_2\cdots, b_0b_1b_2\cdots)$ is in the relation given by the specification.

The desired program P now corresponds to a winning strategy for player output.

Finite automaton solution: P is a finite state machine (S, I, s_0, δ, f) with output function $f : S \times I \to J$.

Example

Alphabet $\{0, 1\}$ for both players.

Winning condition for output player:

- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1

Example

Alphabet $\{0, 1\}$ for both players.

Winning condition for output player:

- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1

Finite automaton solution:

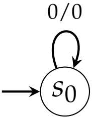
Example

Alphabet $\{0, 1\}$ for both players.

Winning condition for output player:

- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1

Finite automaton solution:



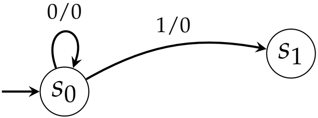
Example

Alphabet $\{0, 1\}$ for both players.

Winning condition for output player:

- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1

Finite automaton solution:



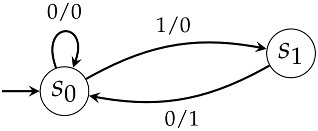
Example

Alphabet $\{0, 1\}$ for both players.

Winning condition for output player:

- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1

Finite automaton solution:



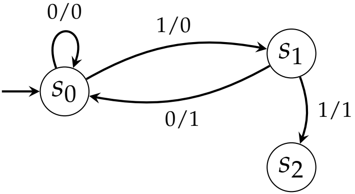
Example

Alphabet $\{0, 1\}$ for both players.

Winning condition for output player:

- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1

Finite automaton solution:



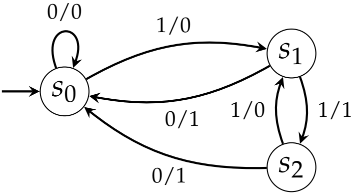
Example

Alphabet $\{0, 1\}$ for both players.

Winning condition for output player:

- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1

Finite automaton solution:



Automatic Relations as Specifications

Winning condition for output player:

- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1

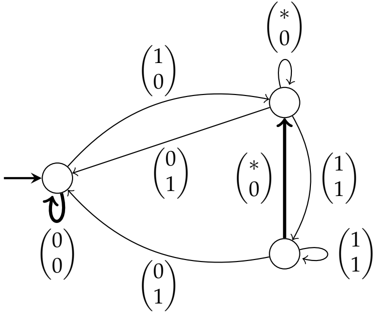
$$\begin{array}{l} 0\,1\,0\,1\,0\,0\,1\,0\,0\,0 \quad \cdots \\ 0\,0\,1\,0\,0\,0\,1\,0\,1\,0 \quad \cdots \end{array}$$

Automatic Relations as Specifications

Winning condition for output player:

- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1

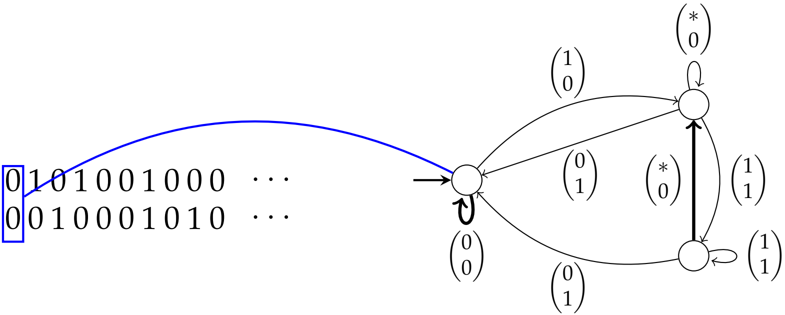
$$\begin{array}{l} 0\,1\,0\,1\,0\,0\,1\,0\,0\,0 \quad \cdots \\ 0\,0\,1\,0\,0\,0\,1\,0\,1\,0 \quad \cdots \end{array}$$



Automatic Relations as Specifications

Winning condition for output player:

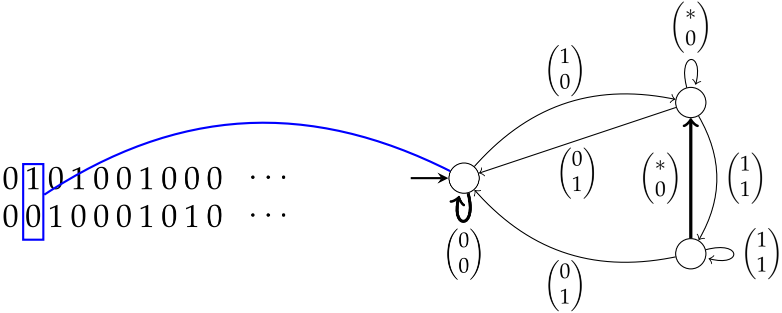
- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1



Automatic Relations as Specifications

Winning condition for output player:

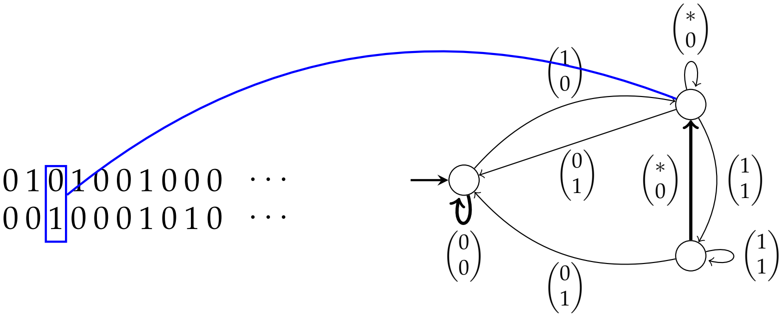
- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1



Automatic Relations as Specifications

Winning condition for output player:

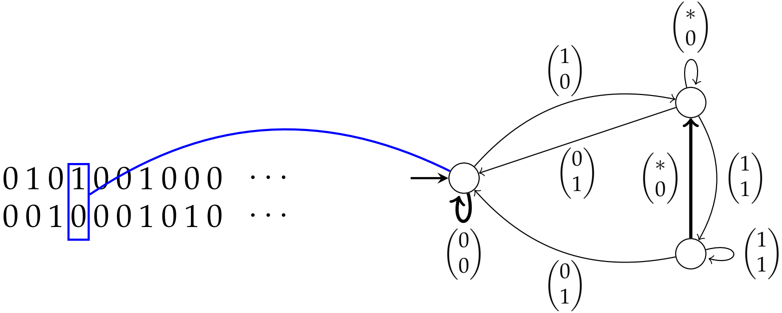
- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1



Automatic Relations as Specifications

Winning condition for output player:

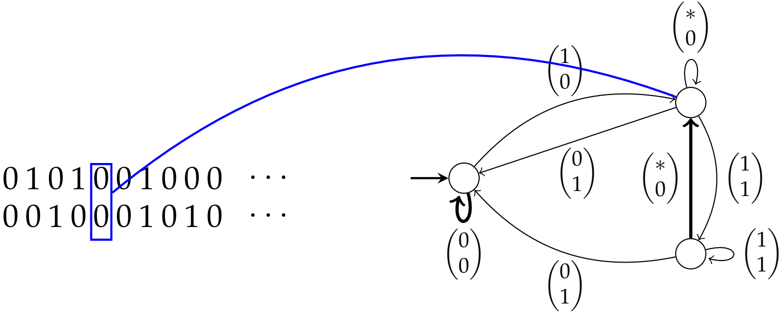
- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1



Automatic Relations as Specifications

Winning condition for output player:

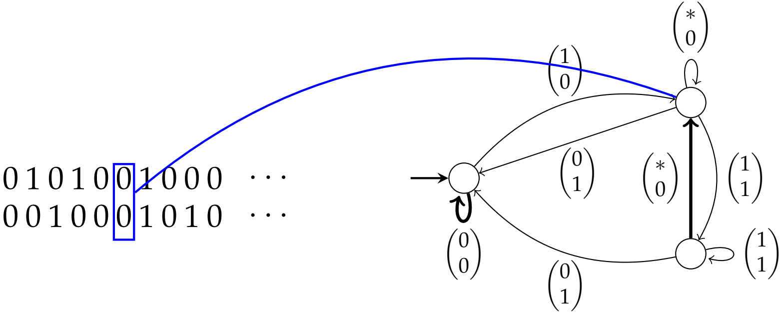
- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1



Automatic Relations as Specifications

Winning condition for output player:

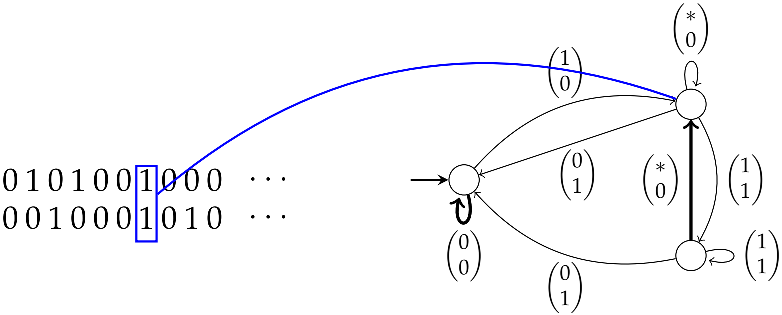
- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1



Automatic Relations as Specifications

Winning condition for output player:

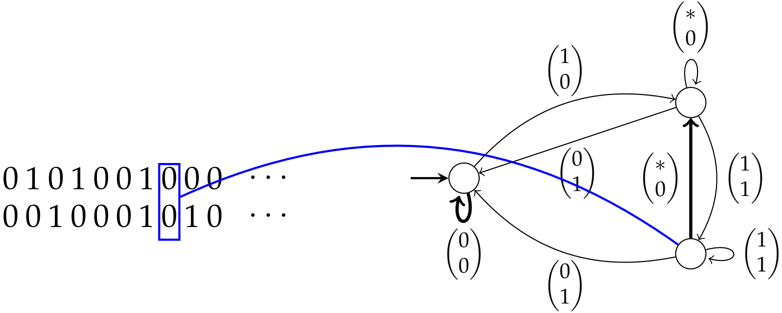
- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1



Automatic Relations as Specifications

Winning condition for output player:

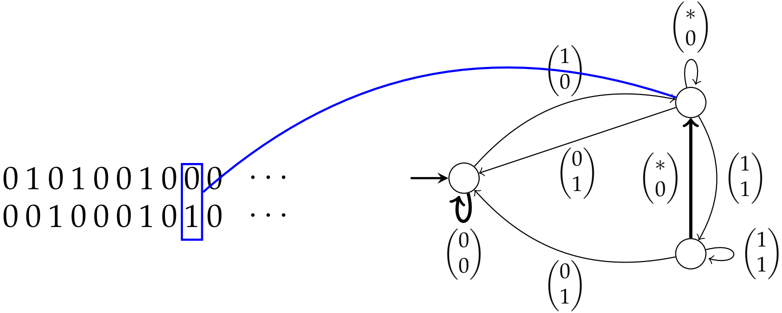
- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1



Automatic Relations as Specifications

Winning condition for output player:

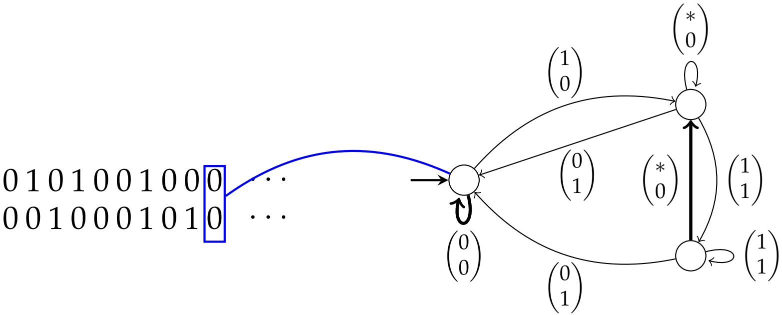
- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1



Automatic Relations as Specifications

Winning condition for output player:

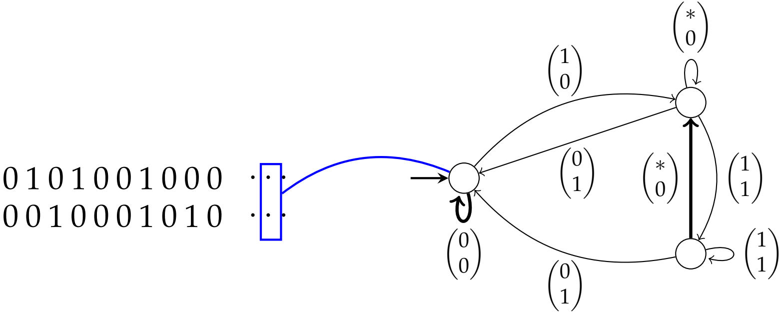
- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1



Automatic Relations as Specifications

Winning condition for output player:

- each input 1 later followed by output 1
- infinitely often output 0
- between two outputs 1 there is an input 1



Büchi-Landweber Theorem

Theorem (Büchi/Landweber 1969). The synchronous synthesis problem for ω-automatic specifications is solvable. If the specification is realizable, then a finite automaton solution can be constructed.

Proof idea:

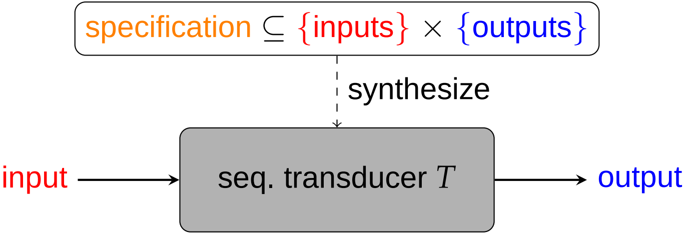
- Use game view of problem.
- Reduce the game with
  - simple rules (players play bits in alternation) but a complex winning condition
  - to a game with more complex rules (played on a finite graph) but much simpler winning condition.
- Compute a strategy in the new game and transfer it back to the initial game.

Outline

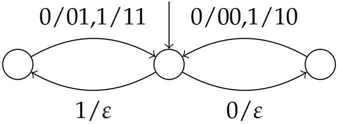
Realizing Specifications with Sequential Transducers

$$\boxed{\text{specification} \subseteq \{\text{inputs}\} \times \{\text{outputs}\}}$$

Realizing Specifications with Sequential Transducers



Sequential transducer: can output a finite word for each input letter



Games with Delay

Player output can skip moves or play several symbols at once.

I

J

Games with Delay

Player output can skip moves or play several symbols at once.

I $\quad a_0$

J

Games with Delay

Player output can skip moves or play several symbols at once.

$I \quad a_0$

$J \quad b_0$

Games with Delay

Player output can skip moves or play several symbols at once.

$I \qquad a_0 \; a_1$

$J \qquad b_0$

Games with Delay

Player output can skip moves or play several symbols at once.

I	a_0	a_1
J	b_0	skip

Games with Delay

Player output can skip moves or play several symbols at once.

I $a_0\ a_1\ a_2$

J b_0

Games with Delay

Player output can skip moves or play several symbols at once.

$I \qquad a_0\ a_1\ a_2$

$J \qquad b_0\ b_1$

Games with Delay

Player output can skip moves or play several symbols at once.

$I \qquad a_0\ a_1\ a_2\ a_3$

$J \qquad b_0\ b_1$

Games with Delay

Player output can skip moves or play several symbols at once.

$I \qquad a_0\ a_1\ a_2\ a_3$

$J \qquad b_0\ b_1$ skip

Games with Delay

Player output can skip moves or play several symbols at once.

$I \qquad a_0\ a_1\ a_2\ a_3\ a_4$

$J \qquad b_0\ b_1$

Games with Delay

Player output can skip moves or play several symbols at once.

$I \qquad a_0\ a_1\ a_2\ a_3\ a_4$

$J \qquad b_0\ b_1\ b_2\ b_3$

Games with Delay

Player output can skip moves or play several symbols at once.

$$\begin{array}{ll} I & a_0\ a_1\ a_2\ a_3\ a_4 \cdots \\ J & b_0\ b_1\ b_2\ b_3 \cdots \end{array} \in \text{specification}$$

Games with Delay

Player output can skip moves or play several symbols at once.

$$\begin{array}{ll} I & a_0\ a_1\ a_2\ a_3\ a_4 \cdots \\ J & b_0\ b_1\ b_2\ b_3 \cdots \end{array} \in \text{specification}$$

A finite automaton winning strategy for II in a delay game corresponds to a sequential transducer realizing the specification.

Examples

$I = J = \{0, 1\}$

- specification = "output at i equals input at $i + 1$"

Examples

$I = J = \{0, 1\}$

- specification = “output at i equals input at $i + 1$”
  Output has to skip once at the beginning.

  I

  J

Examples

$I = J = \{0, 1\}$

- specification = “output at i equals input at $i+1$”
  Output has to skip once at the beginning.

$I \quad 0$

J

Examples

$I = J = \{0, 1\}$

- specification = "output at i equals input at $i+1$"
  Output has to skip once at the beginning.

I	0
J	skip

Examples

$I = J = \{0, 1\}$

- specification = "output at i equals input at $i + 1$"
  Output has to skip once at the beginning.

$$\begin{array}{lcc} I & 0 & 0 \\ & & \\ J & & \end{array}$$

Examples

$I = J = \{0, 1\}$

- specification = “output at i equals input at $i + 1$”
  Output has to skip once at the beginning.

I	0	0
J		0

Examples

$I = J = \{0, 1\}$

- specification = "output at i equals input at $i+1$"
  Output has to skip once at the beginning.

$$\begin{array}{llll} I & 0 & 0 & 1 \cdots \\ J & 0 & & \end{array}$$

Examples

$I = J = \{0, 1\}$

- specification = "output at i equals input at $i + 1$"
  Output has to skip once at the beginning.

  $$\begin{array}{llll} I & 0 & 0 & 1 \cdots \\ & & & \\ J & 0 & 1 \cdots & \end{array}$$

Examples

$I = J = \{0, 1\}$

- specification = "output at i equals input at $i + 1$"
  Output has to skip once at the beginning.

  $$\begin{array}{llll} I & 0 & 0 & 1 \cdots \\ \\ J & & 0 & 1 \cdots \end{array}$$

- "start output with 1 iff there is 1 somewhere in the input"

Examples

$I = J = \{0, 1\}$

- specification = "output at i equals input at $i+1$"
  Output has to skip once at the beginning.

  $$\begin{array}{llll} I & 0 & 0 & 1\cdots \\ J & 0 & 1\cdots & \end{array}$$

- "start output with 1 iff there is 1 somewhere in the input"

  There is no strategy with delay for this specification.

  I

  J

Examples

$I = J = \{0,1\}$

- specification = “output at i equals input at $i+1$”
  Output has to skip once at the beginning.

  $$\begin{array}{llll} I & 0 & 0 & 1 \cdots \\ J & 0 & 1 \cdots & \end{array}$$

- “start output with 1 iff there is 1 somewhere in the input”

  There is no strategy with delay for this specification.

  $$\begin{array}{ll} I & 0 \\ J & \end{array}$$

Examples

$I = J = \{0, 1\}$

- specification = "output at i equals input at $i+1$"
  Output has to skip once at the beginning.

  $I \quad 0 \;\; 0 \;\; 1 \cdots$

  $J \quad 0 \;\; 1 \cdots$

- "start output with 1 iff there is 1 somewhere in the input"

  There is no strategy with delay for this specification.

  $I \quad 0$

  J skip

Examples

$I = J = \{0,1\}$

- specification = "output at i equals input at $i+1$"
  Output has to skip once at the beginning.

  $$\begin{array}{llll} I & 0 & 0 & 1\cdots \\ J & 0 & 1\cdots & \end{array}$$

- "start output with 1 iff there is 1 somewhere in the input"

  There is no strategy with delay for this specification.

  $$\begin{array}{lll} I & 0 & 0 \\ J & & \end{array}$$

Examples

$I = J = \{0,1\}$

- specification = "output at i equals input at $i+1$"
  Output has to skip once at the beginning.

  $I \quad 0 \;\; 0 \;\; 1 \cdots$

  $J \quad 0 \;\; 1 \cdots$

- "start output with 1 iff there is 1 somewhere in the input"

  There is no strategy with delay for this specification.

  $I \quad 0 \;\; 0$

  J skip

Examples

$I = J = \{0, 1\}$

- specification = "output at i equals input at $i + 1$"
  Output has to skip once at the beginning.

  $$\begin{array}{llll} I & 0 & 0 & 1 \cdots \\ J & 0 & 1 \cdots & \end{array}$$

- "start output with 1 iff there is 1 somewhere in the input"

  There is no strategy with delay for this specification.

  $$\begin{array}{llll} I & 0 & 0 & 0 \\ J & & & \end{array}$$

Examples

$I = J = \{0, 1\}$

- specification = "output at i equals input at $i + 1$"
  Output has to skip once at the beginning.

  $I \quad 0 \;\; 0 \;\; 1 \cdots$

  $J \quad 0 \;\; 1 \cdots$

- "start output with 1 iff there is 1 somewhere in the input"

  There is no strategy with delay for this specification.

  $I \quad 0 \;\; 0 \;\; 0$

  J skip

Examples

$I = J = \{0, 1\}$

- specification = “output at i equals input at $i+1$”
  Output has to skip once at the beginning.

  $$\begin{array}{llll} I & 0 & 0 & 1 \cdots \\ J & 0 & 1 \cdots & \end{array}$$

- “start output with 1 iff there is 1 somewhere in the input”

  There is no strategy with delay for this specification.

  $$\begin{array}{lllll} I & 0 & 0 & 0 & 0 \\ J & & & & \end{array}$$

Examples

$I = J = \{0, 1\}$

- specification = "output at i equals input at $i + 1$"
  Output has to skip once at the beginning.

  $$\begin{array}{lllll} I & 0 & 0 & 1 \cdots \\ J & 0 & 1 \cdots \end{array}$$

- "start output with 1 iff there is 1 somewhere in the input"

  There is no strategy with delay for this specification.

  $$\begin{array}{lllll} I & 0 & 0 & 0 & 0 \\ J & 0 \end{array}$$

Examples

$I = J = \{0, 1\}$

- specification = "output at i equals input at $i+1$"
  Output has to skip once at the beginning.

  $$\begin{array}{lllll} I & 0 & 0 & 1 \cdots \\ J & 0 & 1 \cdots \end{array}$$

- "start output with 1 iff there is 1 somewhere in the input"

  There is no strategy with delay for this specification.

  $$\begin{array}{llllll} I & 0 & 0 & 0 & 0 & 1 \cdots \\ J & 0 \end{array}$$

Bounded Delay in ω-Regular Games

Theorem (Hosch/Landweber'72,Holtmann/Kaiser/Thomas'10). For ω-automatic specifications it is decidable if there is a strategy with delay realizing the specification. Furthermore, strategies with bounded delay are sufficient.

Bounded Delay in ω-Regular Games

Theorem (Hosch/Landweber'72,Holtmann/Kaiser/Thomas'10). For ω-automatic specifications it is decidable if there is a strategy with delay realizing the specification. Furthermore, strategies with bounded delay are sufficient.

Corollary. It is decidable whether an ω-automatic specification can be realized by a sequential transducer.

Bounded Delay in ω-Regular Games

Theorem (Hosch/Landweber'72,Holtmann/Kaiser/Thomas'10). For ω-automatic specifications it is decidable if there is a strategy with delay realizing the specification. Furthermore, strategies with bounded delay are sufficient.

Corollary. It is decidable whether an ω-automatic specification can be realized by a sequential transducer.

Bounded delay is sufficient basically because Player output has to produce an infinite sequence for each input.

$\rightsquigarrow$ What about finite words?

Finite Words

Given: Specification as automatic relation over finite words.

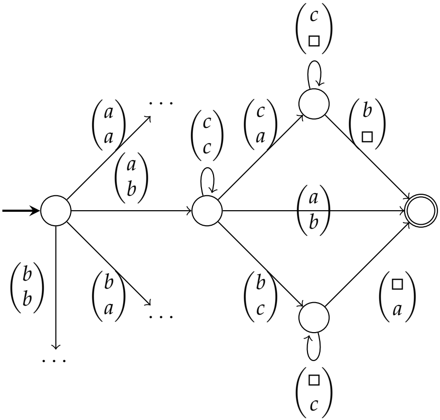
Question: Does there exist a sequential transducer implementing the specifictaion?

Automatic Relations over Finite Words

Example: Alphabet $\{a, b, c\}$

$$R = (ac^*b, bc^*a) \cup (bc^*a, ac^*b) \cup (ac^*a, ac^*a) \cup (bc^*b, bc^*b)$$

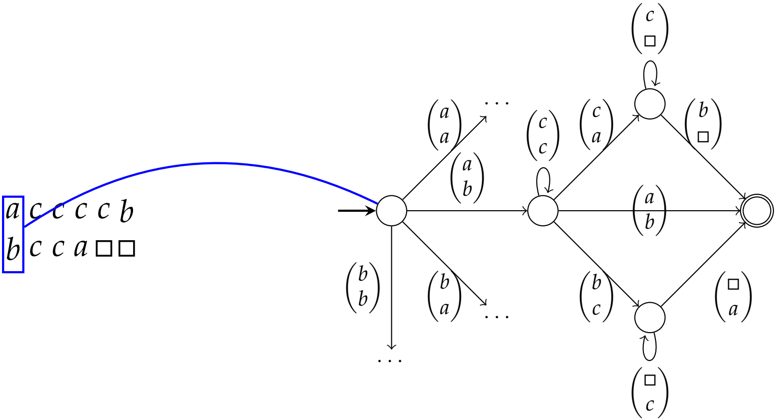
$a\ c\ c\ c\ c\ b$
$b\ c\ c\ a\ \square\ \square$



Automatic Relations over Finite Words

Example: Alphabet $\{a, b, c\}$

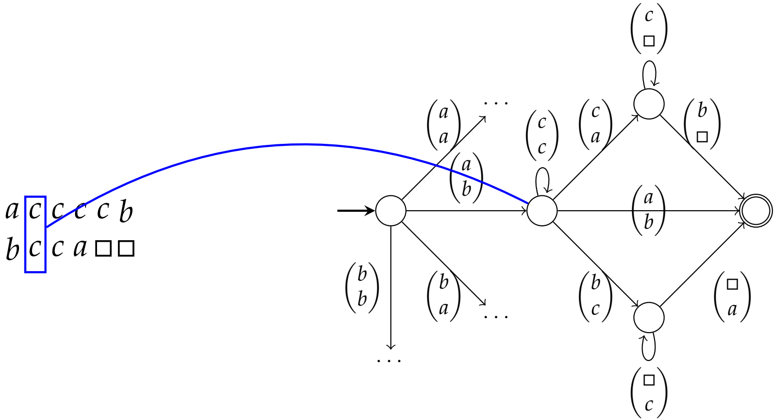
$$R = (ac^*b, bc^*a) \cup (bc^*a, ac^*b) \cup (ac^*a, ac^*a) \cup (bc^*b, bc^*b)$$



Automatic Relations over Finite Words

Example: Alphabet $\{a, b, c\}$

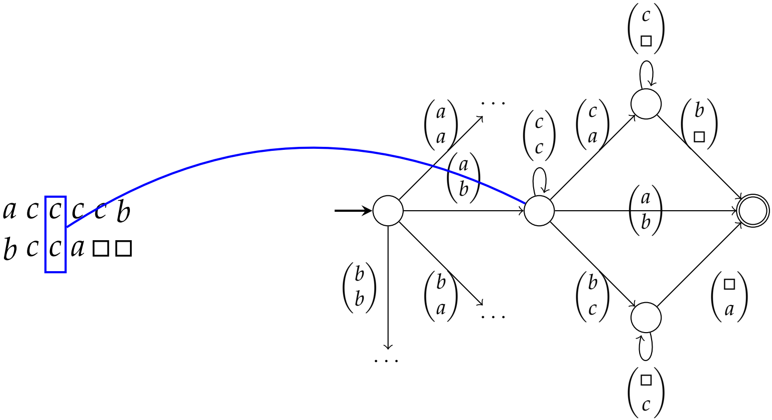
$$R = (ac^*b, bc^*a) \cup (bc^*a, ac^*b) \cup (ac^*a, ac^*a) \cup (bc^*b, bc^*b)$$



Automatic Relations over Finite Words

Example: Alphabet $\{a, b, c\}$

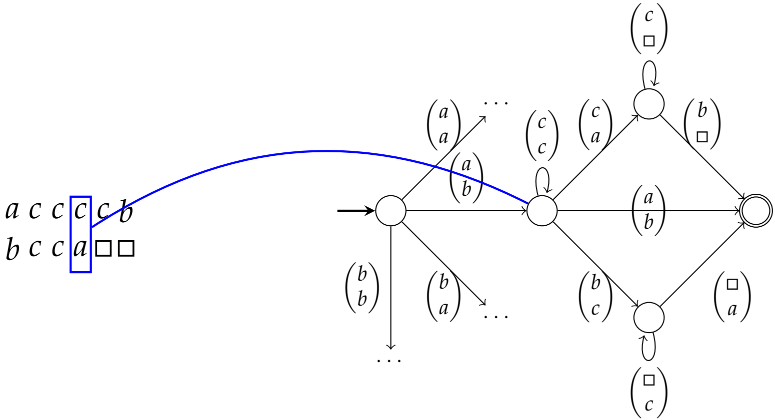
$$R = (ac^*b, bc^*a) \cup (bc^*a, ac^*b) \cup (ac^*a, ac^*a) \cup (bc^*b, bc^*b)$$



Automatic Relations over Finite Words

Example: Alphabet $\{a, b, c\}$

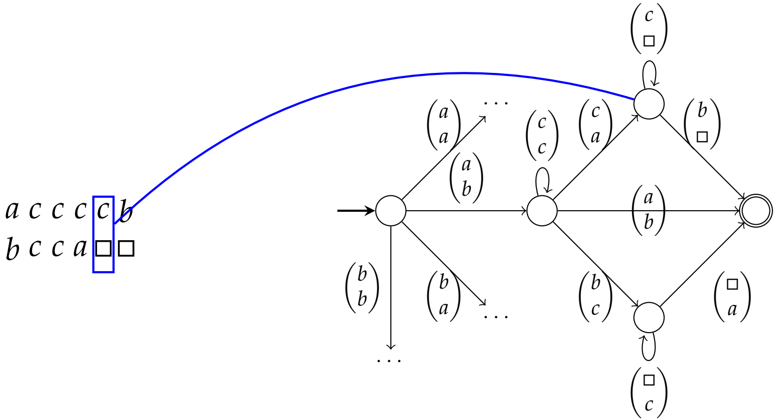
$$R = (ac^*b, bc^*a) \cup (bc^*a, ac^*b) \cup (ac^*a, ac^*a) \cup (bc^*b, bc^*b)$$



Automatic Relations over Finite Words

Example: Alphabet $\{a, b, c\}$

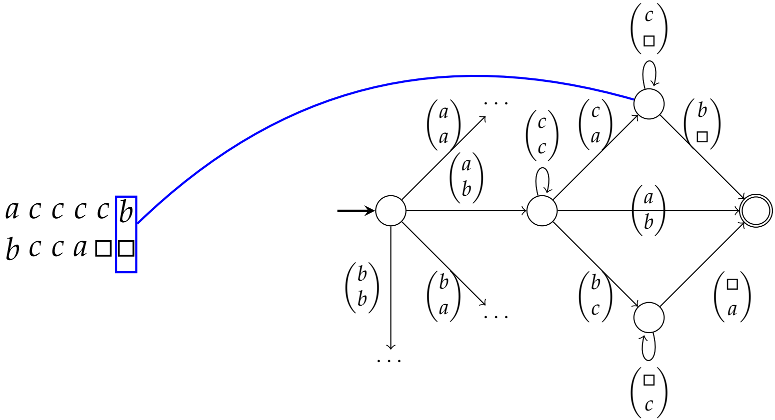
$$R = (ac^*b, bc^*a) \cup (bc^*a, ac^*b) \cup (ac^*a, ac^*a) \cup (bc^*b, bc^*b)$$



Automatic Relations over Finite Words

Example: Alphabet $\{a, b, c\}$

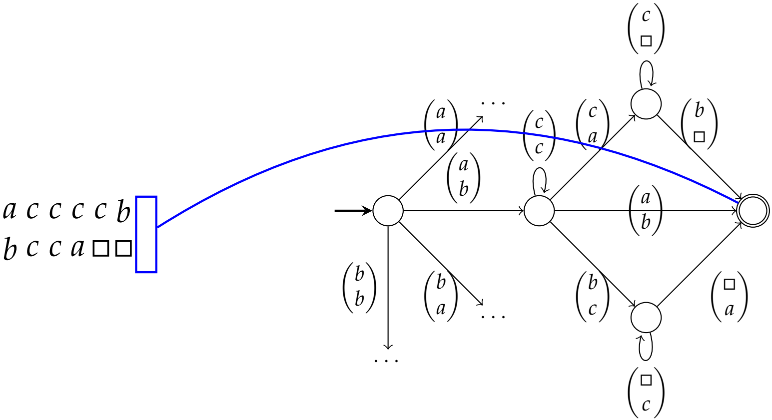
$$R = (ac^*b, bc^*a) \cup (bc^*a, ac^*b) \cup (ac^*a, ac^*a) \cup (bc^*b, bc^*b)$$



Automatic Relations over Finite Words

Example: Alphabet $\{a, b, c\}$

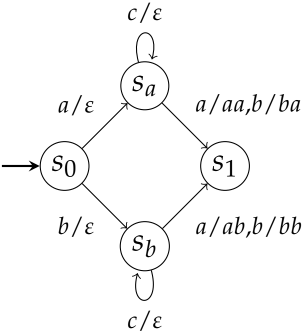
$$R = (ac^*b, bc^*a) \cup (bc^*a, ac^*b) \cup (ac^*a, ac^*a) \cup (bc^*b, bc^*b)$$



Example: Unbounded Delay

$$R = (ac^*b, bc^*a) \cup (bc^*a, ac^*b) \cup (ac^*a, ac^*a) \cup (bc^*b, bc^*b)$$

Realization by sequential transducer:



Result

Theorem (Carayol/L.'12). For an automatic specification (over finite words) it is decidable whether it can be realized by a sequential transducer.

Proof idea:

- Either the delay remains within a computable bound K ($\leadsto$ use the standard game theory techniques),
- or the sequential transducer can delay its output until the whole input is known.

Outline

1. Automaton Synthesis from Specifications
   - Classical Setting
   - Delaying the Output
   - Beyond Finite Automata

2. Synthesis of Lookahead Delegators

Beyond Finite Automata

Use pushdown automata (finite automata + stack) instead of finite automata.

Without Delay:

Theorem (Walukiewicz'96). The synchronous synthesis problem for deterministic pushdown specifications is decidable. If the specification is realizable, then it can be implemented by a pushdown automaton.

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0, 1\}$):

$\binom{0}{0}^\omega$ or

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0, 1\}$):

$$\binom{0}{0}^{\omega} \text{ or } \binom{0}{0}^{n} \binom{0}{1}^{n} \binom{1}{J} \binom{I}{J}^{\omega} \text{ or}$$

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0, 1\}$):

$$\binom{0}{0}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n}\binom{1}{J}\binom{I}{J}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n+1}\binom{1}{J}\binom{I}{J}^{\omega}$$

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0, 1\}$):

$$\binom{0}{0}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n}\binom{1}{J}\binom{I}{J}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n+1}\binom{1}{J}\binom{I}{J}^{\omega}$$

There is a strategy with linear delay:

I

J

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0, 1\}$):

$$\binom{0}{0}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n}\binom{1}{J}\binom{I}{J}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n+1}\binom{1}{J}\binom{I}{J}^{\omega}$$

There is a strategy with linear delay:

I	0
J	

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0,1\}$):

$$\binom{0}{0}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n}\binom{1}{J}\binom{I}{J}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n+1}\binom{1}{J}\binom{I}{J}^{\omega}$$

There is a strategy with linear delay:

I $\quad 0$

J $\quad$ skip

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0, 1\}$):

$$\binom{0}{0}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n}\binom{1}{J}\binom{I}{J}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n+1}\binom{1}{J}\binom{I}{J}^{\omega}$$

There is a strategy with linear delay:

I $\quad 0 \quad 0$

J

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0, 1\}$):

$$\binom{0}{0}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n}\binom{1}{J}\binom{I}{J}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n+1}\binom{1}{J}\binom{I}{J}^{\omega}$$

There is a strategy with linear delay:

I	0	0
J	0	

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0,1\}$):

$$\binom{0}{0}^{\omega} \text{ or } \binom{0}{0}^{n} \binom{0}{1}^{n} \binom{1}{J} \binom{I}{J}^{\omega} \text{ or } \binom{0}{0}^{n} \binom{0}{1}^{n+1} \binom{1}{J} \binom{I}{J}^{\omega}$$

There is a strategy with linear delay:

I	0	0	0
J	0		

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0,1\}$):

$$\binom{0}{0}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n}\binom{1}{J}\binom{I}{J}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n+1}\binom{1}{J}\binom{I}{J}^{\omega}$$

There is a strategy with linear delay:

$$\begin{array}{ll} I & 0\ \ 0\ \ 0 \\ J & 0\ \text{skip} \end{array}$$

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0,1\}$):

$$\binom{0}{0}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n}\binom{1}{J}\binom{I}{J}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n+1}\binom{1}{J}\binom{I}{J}^{\omega}$$

There is a strategy with linear delay:

I	0	0	0	0
J	0			

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0, 1\}$):

$$\binom{0}{0}^{\omega} \text{ or } \binom{0}{0}^{n} \binom{0}{1}^{n} \binom{1}{J} \binom{I}{J}^{\omega} \text{ or } \binom{0}{0}^{n} \binom{0}{1}^{n+1} \binom{1}{J} \binom{I}{J}^{\omega}$$

There is a strategy with linear delay:

$$\begin{array}{llllll} I & \quad & 0 & 0 & 0 & 0 \\ J & \quad & 0 & 0 & & \end{array}$$

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0, 1\}$):

$$\binom{0}{0}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n}\binom{1}{J}\binom{I}{J}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n+1}\binom{1}{J}\binom{I}{J}^{\omega}$$

There is a strategy with linear delay:

I	0	0	0	0	0
J	0	0			

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0, 1\}$):

$$\binom{0}{0}^{\omega} \text{ or } \binom{0}{0}^{n} \binom{0}{1}^{n} \binom{1}{J} \binom{I}{J}^{\omega} \text{ or } \binom{0}{0}^{n} \binom{0}{1}^{n+1} \binom{1}{J} \binom{I}{J}^{\omega}$$

There is a strategy with linear delay:

I	0	0	0	0	0
J	0	0 skip			

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0,1\}$):

$$\binom{0}{0}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n}\binom{1}{J}\binom{I}{J}^{\omega} \text{ or } \binom{0}{0}^{n}\binom{0}{1}^{n+1}\binom{1}{J}\binom{I}{J}^{\omega}$$

There is a strategy with linear delay:

I	0	0	0	0	0	1
J	0	0				

Beyond Finite Automata

With Delay:

Example: Specification allows the following pairs of input/output sequences (with $I = J = \{0, 1\}$):

$$\binom{0}{0}^{\omega} \text{ or } \binom{0}{0}^{n} \binom{0}{1}^{n} \binom{1}{J} \binom{I}{J}^{\omega} \text{ or } \binom{0}{0}^{n} \binom{0}{1}^{n+1} \binom{1}{J} \binom{I}{J}^{\omega}$$

There is a strategy with linear delay:

$$\begin{array}{llllllll} I & 0 & 0 & 0 & 0 & 0 & 1 \cdots \\ J & 0 & 0 & 1 \cdots & & & \end{array}$$

Beyond Finite Automata

Theorem (Fridman/L./Zimmerman'11).

- There are deterministic pushdown specifications for which there is a delay strategy but each such strategy needs non-elementary delay.
- For deterministic pushdown specifications it is undecidable if there is a strategy with delay realizing the specification.

Outline

1. Automaton Synthesis from Specifications
   - Classical Setting
   - Delaying the Output
   - Beyond Finite Automata

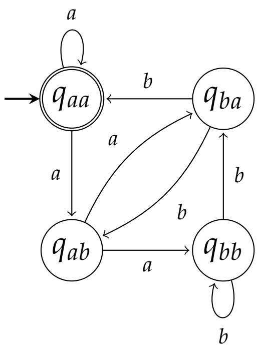
2. Synthesis of Lookahead Delegators

Problem Setting

Given an NFA, decide whether it can deterministically choose its transitions using a bounded lookahead.

A function choosing the transitions with lookahead k is called a k-lookahead delegator for $\mathcal{A}$.

Decision Problem: Given NFA $\mathcal{A}$ and a number k, does there exist a k-lookahead delegator for $\mathcal{A}$?

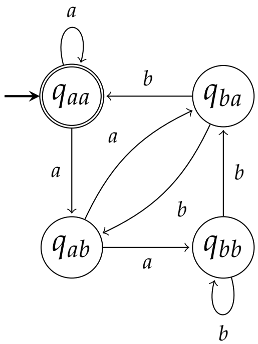


Problem Setting

Given an NFA, decide whether it can deterministically choose its transitions using a bounded lookahead.

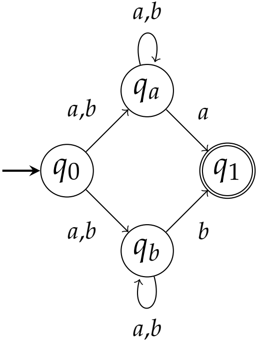
A function choosing the transitions with lookahead k is called a k-lookahead delegator for $\mathcal{A}$.

Decision Problem: Given NFA $\mathcal{A}$ and a number k, does there exist a k-lookahead delegator for $\mathcal{A}$?



In this example the two symbols after the current input letter are needed to choose a transition.
$\rightsquigarrow$ 2-lookahead delegator

Example without Lookahead Delegator

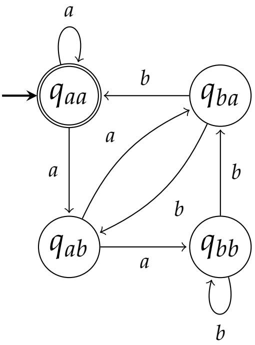


- The NFA guesses the last letter.
- To choose the first transition, the last input letter needs to be known.

Game Approach

Game for $\mathcal{A}$ and k:

- Player I (input) plays an input word.
- Player II (run) plays a run on the input word, and can delay its choices up to k letters.
- In order to win, II has to construct an accepting run if the input is accepted by $\mathcal{A}$.



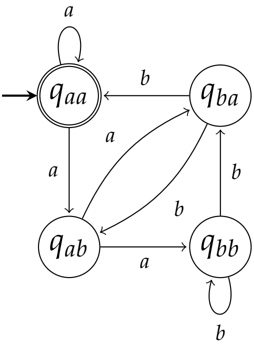
input

run

Game Approach

Game for $\mathcal{A}$ and k:

- Player I (input) plays an input word.
- Player II (run) plays a run on the input word, and can delay its choices up to k letters.
- In order to win, II has to construct an accepting run if the input is accepted by $\mathcal{A}$.

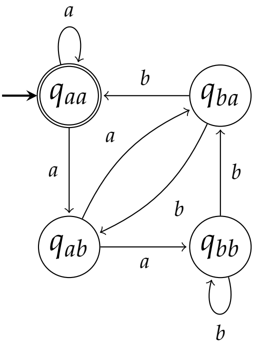


input		a
run	q_{aa}	

Game Approach

Game for $\mathcal{A}$ and k:

- Player I (input) plays an input word.
- Player II (run) plays a run on the input word, and can delay its choices up to k letters.
- In order to win, II has to construct an accepting run if the input is accepted by $\mathcal{A}$.

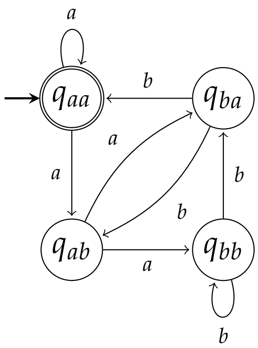


input		a
run	q_{aa}	skip

Game Approach

Game for $\mathcal{A}$ and k:

- Player I (input) plays an input word.
- Player II (run) plays a run on the input word, and can delay its choices up to k letters.
- In order to win, II has to construct an accepting run if the input is accepted by $\mathcal{A}$.

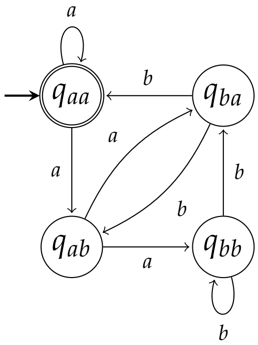


input	a	a
run q_{aa}		

Game Approach

Game for $\mathcal{A}$ and k:

- Player I (input) plays an input word.
- Player II (run) plays a run on the input word, and can delay its choices up to k letters.
- In order to win, II has to construct an accepting run if the input is accepted by $\mathcal{A}$.

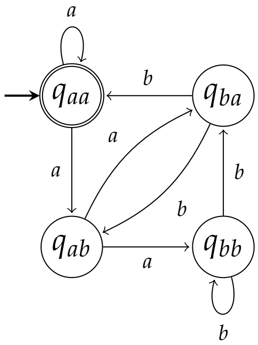


input		a	a
run	q_{aa}	skip	

Game Approach

Game for $\mathcal{A}$ and k:

- Player I (input) plays an input word.
- Player II (run) plays a run on the input word, and can delay its choices up to k letters.
- In order to win, II has to construct an accepting run if the input is accepted by $\mathcal{A}$.

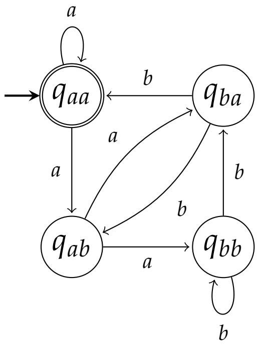


input	a	a	a
run q_{aa}			

Game Approach

Game for $\mathcal{A}$ and k:

- Player I (input) plays an input word.
- Player II (run) plays a run on the input word, and can delay its choices up to k letters.
- In order to win, II has to construct an accepting run if the input is accepted by $\mathcal{A}$.

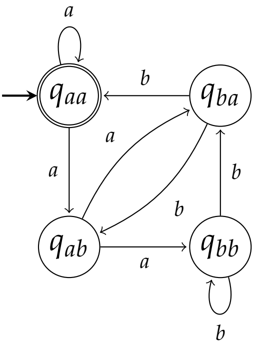


input		a	a	a
run	q_{aa}	q_{aa}		

Game Approach

Game for $\mathcal{A}$ and k:

- Player I (input) plays an input word.
- Player II (run) plays a run on the input word, and can delay its choices up to k letters.
- In order to win, II has to construct an accepting run if the input is accepted by $\mathcal{A}$.

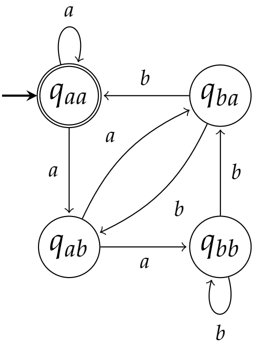


input		a	a	a	b
run	q_{aa}	q_{aa}			

Game Approach

Game for $\mathcal{A}$ and k:

- Player I (input) plays an input word.
- Player II (run) plays a run on the input word, and can delay its choices up to k letters.
- In order to win, II has to construct an accepting run if the input is accepted by $\mathcal{A}$.

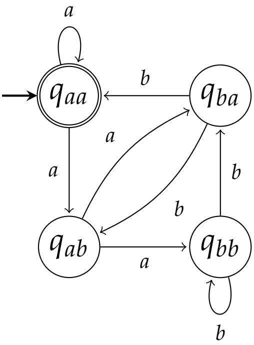


input	a	a	a	b
run	q_{aa}	q_{aa}	q_{ab}	

Game Approach

Game for $\mathcal{A}$ and k:

- Player I (input) plays an input word.
- Player II (run) plays a run on the input word, and can delay its choices up to k letters.
- In order to win, II has to construct an accepting run if the input is accepted by $\mathcal{A}$.

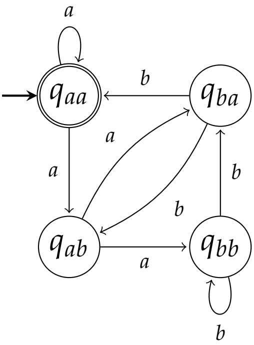


input	a	a	a	b	$\triangleleft$
run	q_{aa}	q_{aa}	q_{ab}		

Game Approach

Game for $\mathcal{A}$ and k:

- Player I (input) plays an input word.
- Player II (run) plays a run on the input word, and can delay its choices up to k letters.
- In order to win, II has to construct an accepting run if the input is accepted by $\mathcal{A}$.

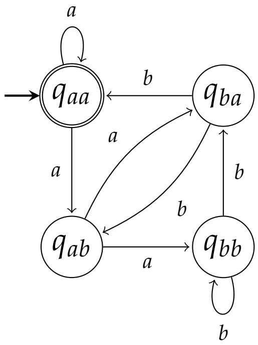


input		a		a		a		b	$\triangleleft$
run	q_{aa}		q_{aa}		q_{ab}		q_{ba}		

Game Approach

Game for $\mathcal{A}$ and k:

- Player I (input) plays an input word.
- Player II (run) plays a run on the input word, and can delay its choices up to k letters.
- In order to win, II has to construct an accepting run if the input is accepted by $\mathcal{A}$.



input		a	a	a	b	$\triangleleft$
run	q_{aa}	q_{aa}	q_{ab}	q_{ba}	q_{aa}	

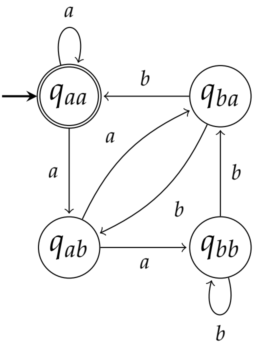
Problem with this Approach

- The winning condition requires to check whether the input is accepted by $\mathcal{A}$.
- This makes the game difficult to solve.

Solution

- Player input also chooses a transition, but after Player run has chosen one.
- New winning condition: if player input ends in a final state, then player run also has to end in a final state.

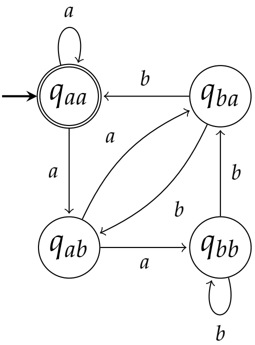
Extended Game



input

run

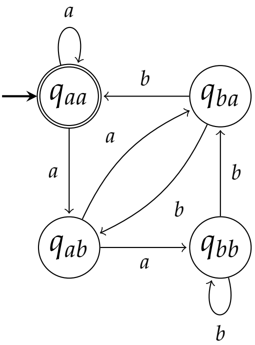
Extended Game



input $\quad q_{aa}\ a$

run $\quad q_{aa}$

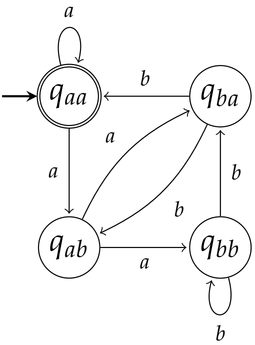
Extended Game



input q_{aa} a

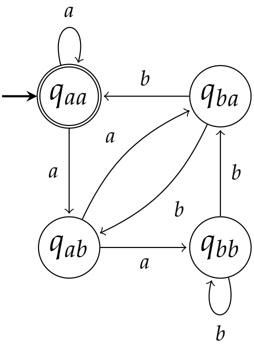
run q_{aa} skip

Extended Game



input	q_{aa}	a		a
run	q_{aa}			

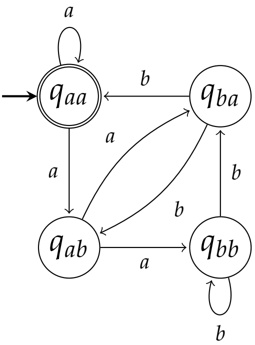
Extended Game



input q_{aa} a a

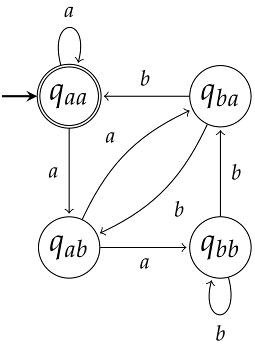
run q_{aa} q_{aa}

Extended Game



input	q_{aa}	a	q_{ab}	a		b
run	q_{aa}		q_{aa}			

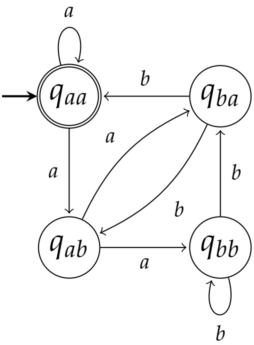
Extended Game



input $q_{aa}\ a\ q_{ab}\ a\ \quad b$

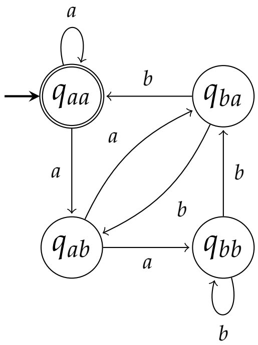
run $q_{aa}\quad q_{aa}\quad q_{ab}$

Extended Game



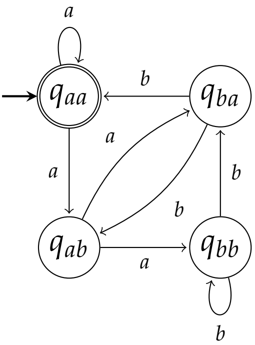
input	q_{aa}	a	q_{ab}	a	q_{ba}	b		$\triangleleft$
run	q_{aa}		q_{aa}		q_{ab}			

Extended Game



input	$q_{aa}\ a\ q_{ab}\ a\ q_{ba}\ b\ q_{aa}\ \triangleleft$
run	$q_{aa}\quad q_{aa}\quad q_{ab}$

Extended Game



input	q_{aa}	a	q_{ab}	a	q_{ba}	b	q_{aa}	$\triangleleft$
run	q_{aa}		q_{aa}		q_{ab}			

- This extended game can be viewed as a safety game of size roughly $|Q|^2 \times |\Sigma|^{k+1}$.
- A winning strategy for II corresponds to a k-lookahead delegator for $\mathcal{A}$.

Result

Theorem (L./Repke'13). Let k be a fixed number. It is decidable in polynomial time whether a given NFA $\mathcal{A}$ has a k-lookahead delegator.

Remark: For $k = 0$ this correponds to deciding whether $\mathcal{A}$ has an equivalent deterministic subautomaton.

Conclusion

Games with delay as a useful tool for synthesis problems in automata theory:

- Sequential transducers from automatic specifications
- Lookahead delegators for NFAs

Some open problems:

- Synthesis of tree transducers
- Lookahead delegators for ω-automata