Uniformization in Automata Theory

Christof Löding

RWTH Aachen

14th Congress of Logic, Methodology and Philosophy of Science
A3 – Logic and Computation

Nancy, July 19-26, 2011

Outline

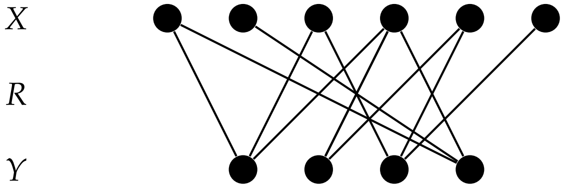
Outline

Uniformization

Given a relation $R \subseteq X \times Y$, a uniformization of R is a function

$$f_R : X \to Y$$

whose graph is a subset of R.

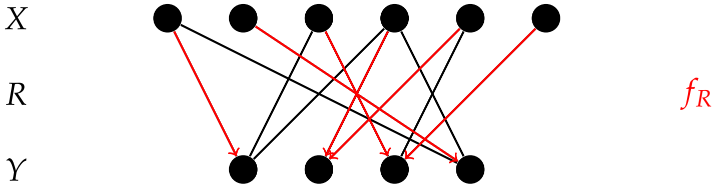


Uniformization

Given a relation $R \subseteq X \times Y$, a uniformization of R is a function

$$f_R : X \to Y$$

whose graph is a subset of R.



Uniformization in Descriptive Set Theory

Axiom of Choice: For each relation R there is some uniformization f_R

Descriptive Set Theory: Study of definable objects. Relate the complexity of R and the complexity of a uniformization f_R:

- For a class $\mathcal{C}$ of relations and a class $\mathcal{F}$ of functions, does each R in $\mathcal{C}$ have a uniformization in $\mathcal{F}$?

Uniformization in Automata Theory

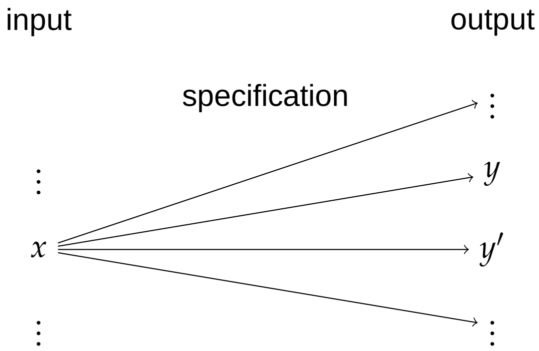
Idea: effective uniformization

Classes $\mathcal{C}, \mathcal{F}$ defined by automata (or equivalent logical formalism)

Questions:

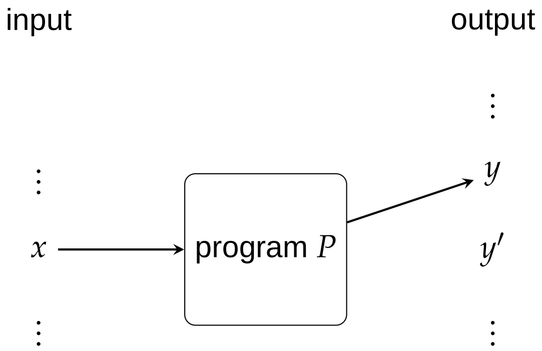
- Does each $R \in \mathcal{C}$ have a uniformization in $\mathcal{F}$?
- Given $R \in \mathcal{C}$ decide if it has a uniformization in $\mathcal{F}$ and effectively construct one if possible.

From Specification to Implementation



Effective uniformization can be seen as a form of automatic program synthesis from specifications.

From Specification to Implementation



Effective uniformization can be seen as a form of automatic program synthesis from specifications.

Outline

Automatic Relations over Different Structures

$$\begin{tikzcd}
\boxed{\text{finite words}} \arrow[r] \arrow[d] & \boxed{\text{finite trees}} \arrow[d] \\
\boxed{\text{infinite words}} \arrow[r] & \boxed{\text{infinite trees}}
\end{tikzcd}$$

Automatic Relations over Finite Words

Example: Alphabet $\{0,1\}$

$$R = \{(1^n, 1^m01^k0^*) \mid n = m + k + 1\}$$

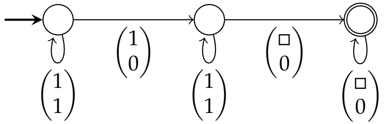
$$\begin{array}{l} 1\,1\,1\,1\,1\,1\,1\,1\,\square\,\square\,\square \\ 1\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0 \end{array}$$

Automatic Relations over Finite Words

Example: Alphabet $\{0, 1\}$

$$R = \{(1^n, 1^m 0 1^k 0^*) \mid n = m + k + 1\}$$

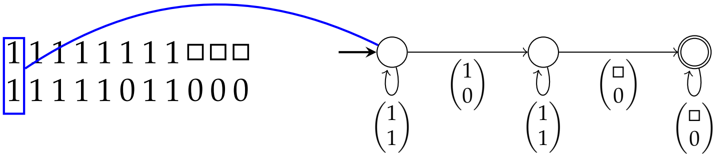
$$\begin{array}{ccccccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \square & \square & \square \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}$$



Automatic Relations over Finite Words

Example: Alphabet $\{0,1\}$

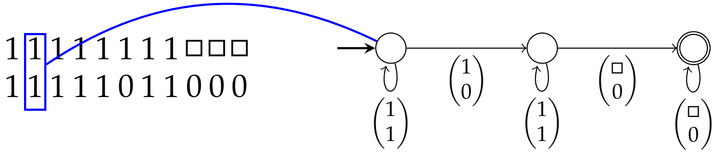
$$R = \{(1^n, 1^m 0 1^k 0^*) \mid n = m+k+1\}$$



Automatic Relations over Finite Words

Example: Alphabet $\{0, 1\}$

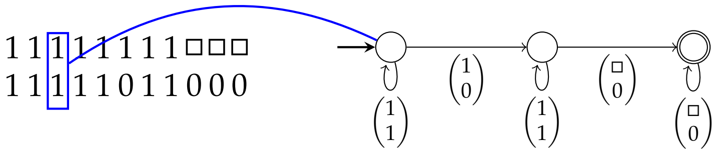
$$R = \{(1^n, 1^m01^k0^*) \mid n = m + k + 1\}$$



Automatic Relations over Finite Words

Example: Alphabet $\{0,1\}$

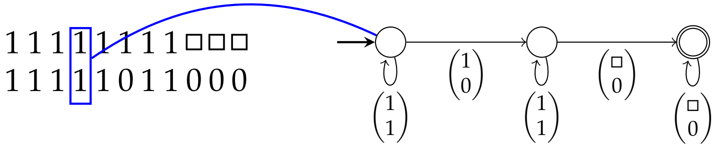
$$R = \{(1^n, 1^m01^k0^*) \mid n = m+k+1\}$$



Automatic Relations over Finite Words

Example: Alphabet $\{0, 1\}$

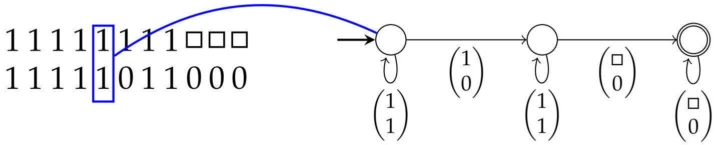
$$R = \{(1^n, 1^m 0 1^k 0^*) \mid n = m + k + 1\}$$



Automatic Relations over Finite Words

Example: Alphabet $\{0, 1\}$

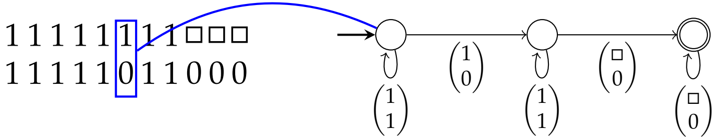
$$R = \{(1^n, 1^m01^k0^*) \mid n = m + k + 1\}$$



Automatic Relations over Finite Words

Example: Alphabet $\{0,1\}$

$$R = \{(1^n, 1^m 0 1^k 0^*) \mid n = m + k + 1\}$$

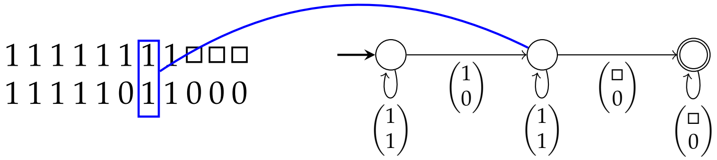


$$\begin{tikzcd} \arrow[r] & q_0 \arrow[loop below, "\binom{1}{1}"] \arrow[r, "\binom{1}{0}"] & q_1 \arrow[loop below, "\binom{1}{1}"] \arrow[r, "\binom{\square}{0}"] & q_2 \arrow[loop below, "\binom{\square}{0}"] \end{tikzcd}$$

Automatic Relations over Finite Words

Example: Alphabet $\{0, 1\}$

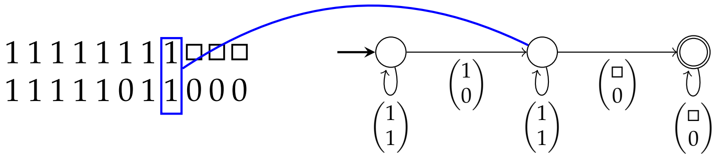
$$R = \{(1^n, 1^m 01^k 0^*) \mid n = m + k + 1\}$$



Automatic Relations over Finite Words

Example: Alphabet $\{0, 1\}$

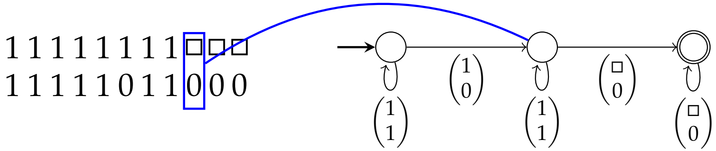
$$R = \{(1^n, 1^m 0 1^k 0^*) \mid n = m + k + 1\}$$



Automatic Relations over Finite Words

Example: Alphabet $\{0, 1\}$

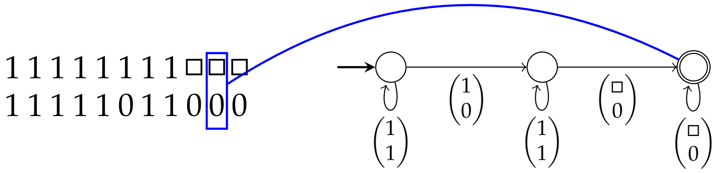
$$R = \{(1^n, 1^m 0 1^k 0^*) \mid n = m + k + 1\}$$



Automatic Relations over Finite Words

Example: Alphabet $\{0, 1\}$

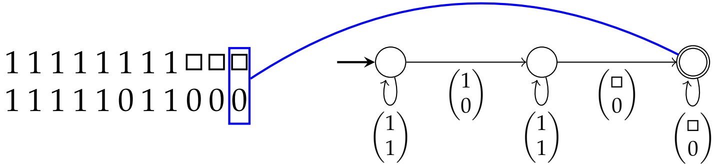
$$R = \{(1^n, 1^m 0 1^k 0^*) \mid n = m + k + 1\}$$



Automatic Relations over Finite Words

Example: Alphabet $\{0, 1\}$

$$R = \{(1^n, 1^m01^k0^*) \mid n = m + k + 1\}$$

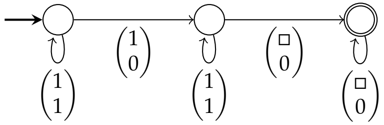


Automatic Relations over Finite Words

Example: Alphabet $\{0, 1\}$

$$R = \{(1^n, 1^m 0 1^k 0^*) \mid n = m + k + 1\}$$

$$\begin{array}{ccccccccccc} 1&1&1&1&1&1&1&1&\square&\square&\square \\ 1&1&1&1&1&0&1&1&0&0&0 \end{array}$$



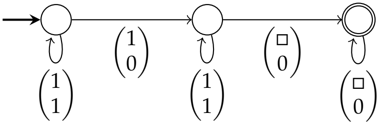
Length-lexicographic uniformization: $f_R = \{(1^n, 01^{n-1})\}$

Automatic Relations over Finite Words

Example: Alphabet $\{0,1\}$

$$R = \{(1^n, 1^m 0 1^k 0^*) \mid n = m + k + 1\}$$

$1\,1\,1\,1\,1\,1\,1\,1\,\square\,\square\,\square$
$1\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0$



Length-lexicographic uniformization: $f_R = \{(1^n, 01^{n-1})\}$

Theorem. For an automatic relation R over finite words the length-lexicographic uniformization f_R is also automatic.

Automatic Relations over Different Structures

$$\begin{tikzcd}
\boxed{\text{finite words}} \arrow[r] \arrow[d] & \boxed{\text{finite trees}} \arrow[d] \\
\boxed{\text{infinite words}} \arrow[r] & \boxed{\text{infinite trees}}
\end{tikzcd}$$

Automatic Relations of Infinite Words

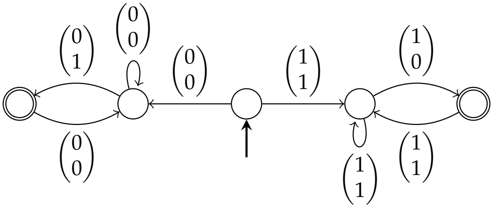
Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$

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

Automatic Relations of Infinite Words

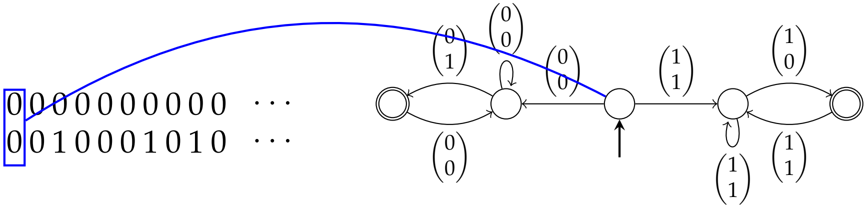
Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$

$$\begin{matrix} 0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & \cdots \\ 0\,0\,1\,0\,0\,0\,1\,0\,1\,0 & \cdots \end{matrix}$$



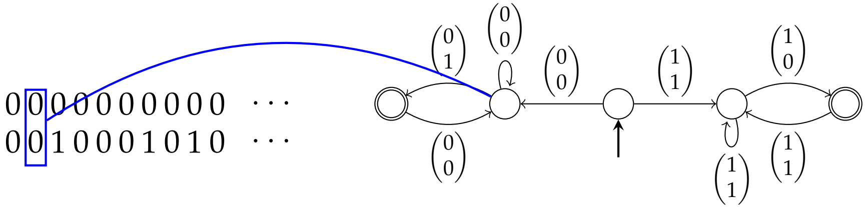
Automatic Relations of Infinite Words

Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$



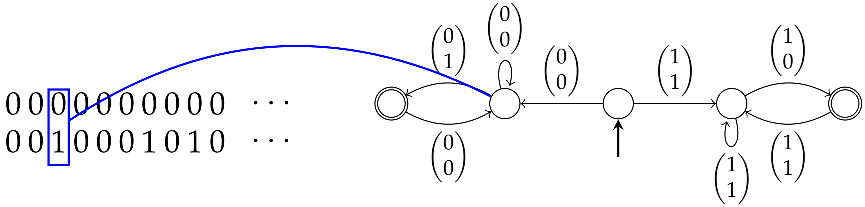
Automatic Relations of Infinite Words

Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$



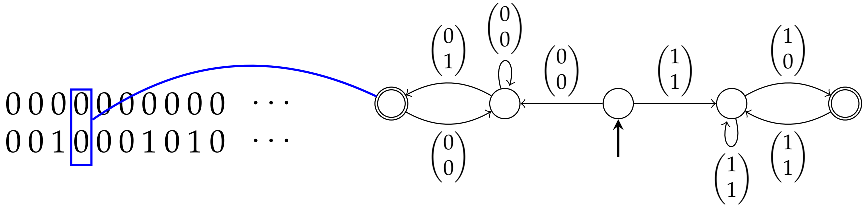
Automatic Relations of Infinite Words

Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$



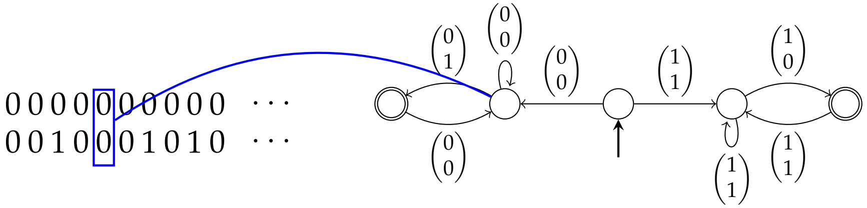
Automatic Relations of Infinite Words

Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$



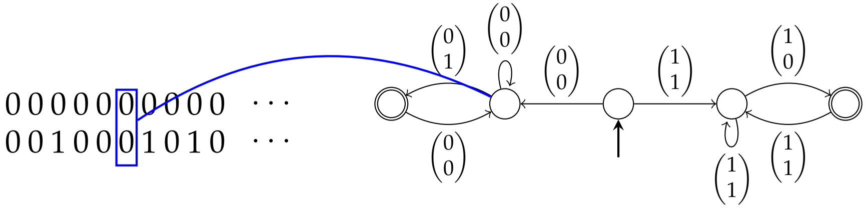
Automatic Relations of Infinite Words

Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$



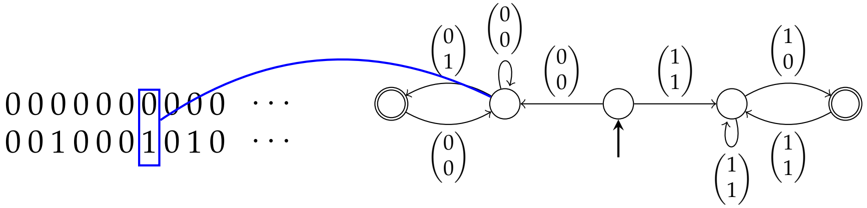
Automatic Relations of Infinite Words

Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$



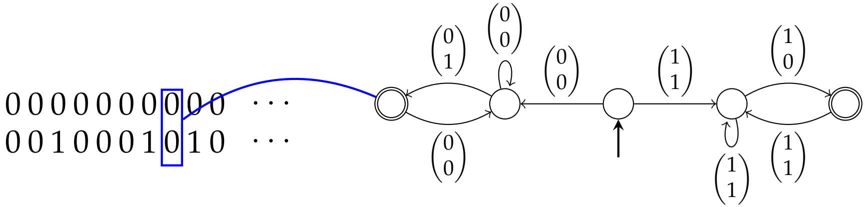
Automatic Relations of Infinite Words

Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$



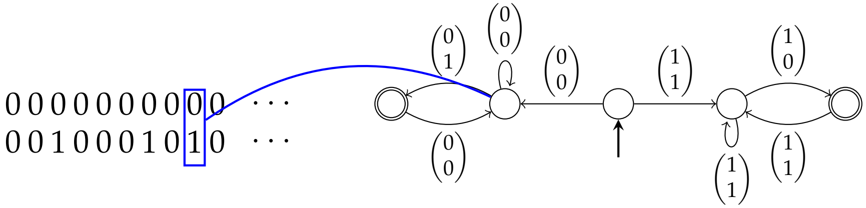
Automatic Relations of Infinite Words

Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$



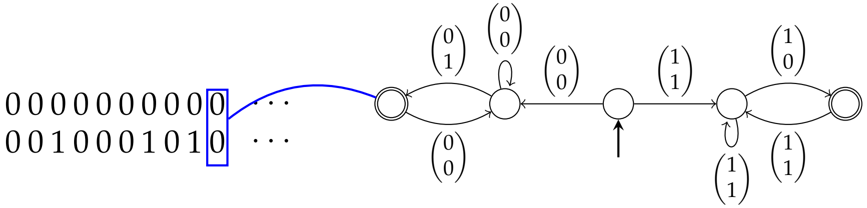
Automatic Relations of Infinite Words

Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$



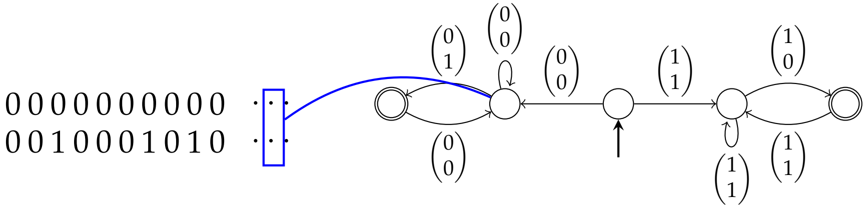
Automatic Relations of Infinite Words

Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$



Automatic Relations of Infinite Words

Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$

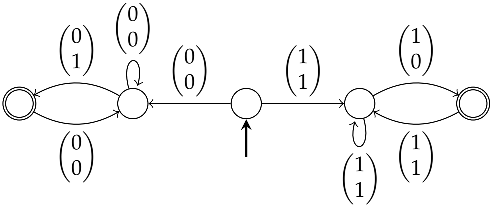


$$\binom{0}{1}\ \binom{0}{0}\ \binom{0}{0}\ \binom{0}{0}\ \binom{1}{1}\ \binom{1}{0}\ \binom{1}{1}\ \binom{1}{1}$$

Automatic Relations of Infinite Words

Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$

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



Observation: lexicographic uniformization does not work

Solution: greedy ordering of accepting runs (visit accepting state as soon as possible)

Theorem (Siefkes'75, Choffrut/Grigorieff'99). Automatic relations over infinite words have automatic uniformizations.

Connection to Logic

- Infinite 0-1-sequences $\leftrightarrow$ sets of natural numbers
- Monadic second-order logic (MSO) over the structure $(\mathbb{N}, +1)$

  MSO $\hat{=}$ first-order logic $+$ set quantification
- Relations of 0-1-sequences can be expressed by formulas $\varphi(X, Y)$

Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$

$$\begin{aligned} \varphi(X,Y) = \quad & (\forall x \neg X(x)) \rightarrow \\ & \forall y \exists z > y (Y(z) \wedge (Y(y) \rightarrow \neg Y(y+1))) \\ & \dots \end{aligned}$$

Connection to Logic

- Infinite 0-1-sequences $\leftrightarrow$ sets of natural numbers
- Monadic second-order logic (MSO) over the structure $(\mathbb{N}, +1)$

  MSO $\hat{=}$ first-order logic $+$ set quantification
- Relations of 0-1-sequences can be expressed by formulas $\varphi(X, Y)$

Example: $(0^\omega, (0^+1)^\omega) \cup (1^\omega, (1^+0)^\omega)$

$$\begin{aligned}\varphi(X,Y) = \;& (\forall x \neg X(x)) \rightarrow \\ & \forall y \exists z > y (Y(z) \wedge (Y(y) \rightarrow \neg Y(y+1))) \\ & \ldots\end{aligned}$$

Theorem (Büchi'60). Nondeterministic finite automata and MSO over infinite words have the same expressive power.

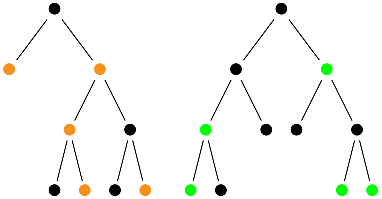
Corollary. For every MSO formula $\varphi(X, Y)$ there is a formula $\psi(X, Y)$ defining a uniformization of the relation defined by φ.

Automatic Relations over Different Structures

$$\begin{tikzcd}
\boxed{\text{finite words}} \arrow[r] \arrow[d] & \boxed{\text{finite trees}} \arrow[d] \\
\boxed{\text{infinite words}} \arrow[r] & \boxed{\text{infinite trees}}
\end{tikzcd}$$

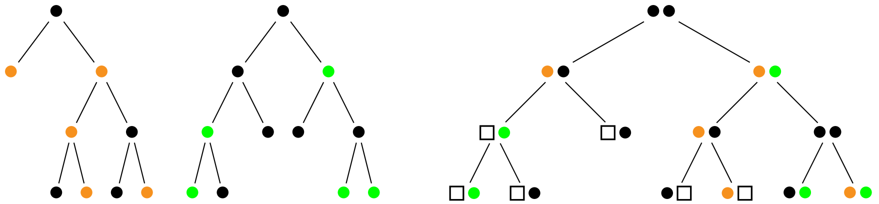
Finite Trees

How to combine trees?



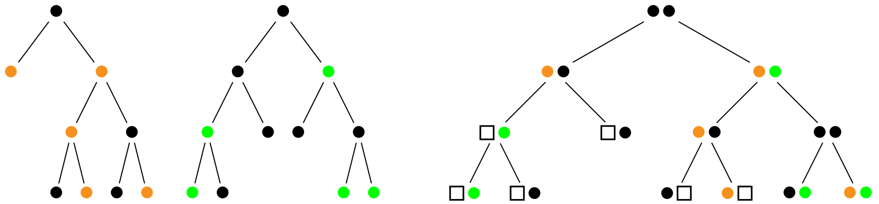
Finite Trees

How to combine trees?



Finite Trees

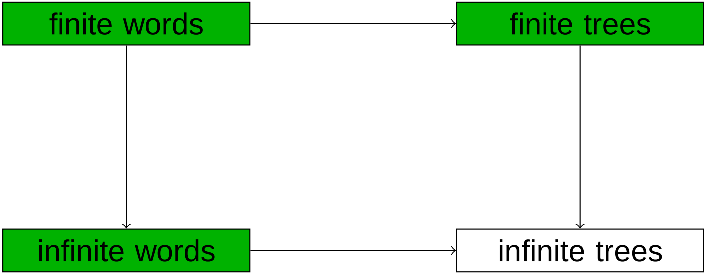
How to combine trees?



From results in Colcombet/L.'06 and Kuske/Weidner'11:

Theorem. Every tree automatic relation has a tree automatic uniformization.

Automatic Relations over Different Structures

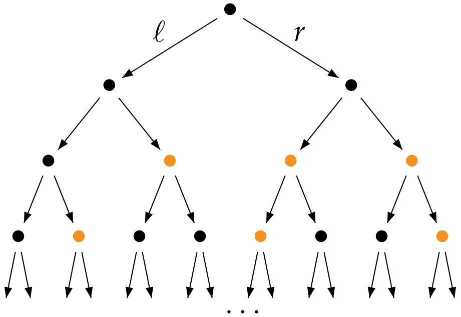


Infinite Trees

Infinite binary tree structure $T_2 = (\{\ell, r\}^*, S_\ell, S_r)$

A 0-1-labeling of T_2 corresponds to subset of $\{\ell, r\}^*$.

As for infinite words: Finite automata $\hat{=}$ MSO over T_2 (Rabin'69)



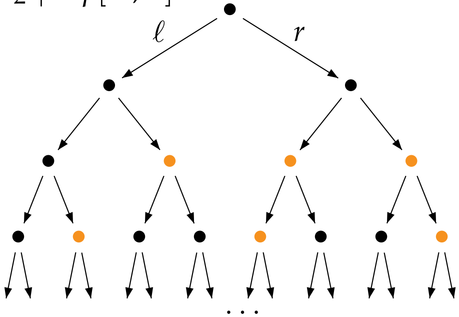
Infinite Trees

Infinite binary tree structure $T_2 = (\{\ell, r\}^*, S_\ell, S_r)$

A 0-1-labeling of T_2 corresponds to subset of $\{\ell, r\}^*$.

As for infinite words: Finite automata $\hat{=}$ MSO over T_2 (Rabin'69)

Theorem(Gurevich/Shelah'83). There is no MSO-definable choice function over the infinite binary tree, i.e., there is no formula $\varphi(X, y)$ such that for each (nonempty) set U there is exactly one node $u \in U$ such that $T_2 \models \varphi[U, u]$.



Infinite Trees

Infinite binary tree structure $T_2 = (\{\ell, r\}^*, S_\ell, S_r)$

A 0-1-labeling of T_2 corresponds to subset of $\{\ell, r\}^*$.

As for infinite words: Finite automata $\hat{=}$ MSO over T_2 (Rabin'69)

Theorem(Gurevich/Shelah'83). There is no MSO-definable choice function over the infinite binary tree, i.e., there is no formula $\varphi(X, y)$ such that for each (nonempty) set U there is exactly one node $u \in U$ such that $T_2 \models \varphi[U, u]$.

Corollary. There is no MSO definable (automatic) uniformization of the "element relation".

Infinite Trees

Infinite binary tree structure $T_2 = (\{\ell, r\}^*, S_\ell, S_r)$

A 0-1-labeling of T_2 corresponds to subset of $\{\ell, r\}^*$.

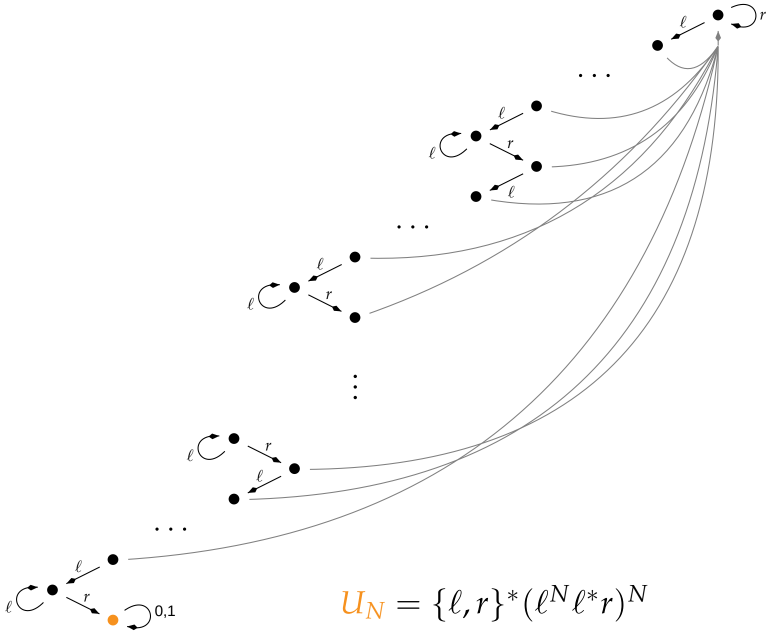
As for infinite words: Finite automata $\hat{=}$ MSO over T_2 (Rabin'69)

Theorem(Gurevich/Shelah'83). There is no MSO-definable choice function over the infinite binary tree, i.e., there is no formula $\varphi(X, y)$ such that for each (nonempty) set U there is exactly one node $u \in U$ such that $T_2 \models \varphi[U, u]$.

Corollary. There is no MSO definable (automatic) uniformization of the "element relation".

Remark: The proof of Gurevich/Shelah'83 used methods from set theory. A new automata theoretic proof yields a simple family of counter examples.

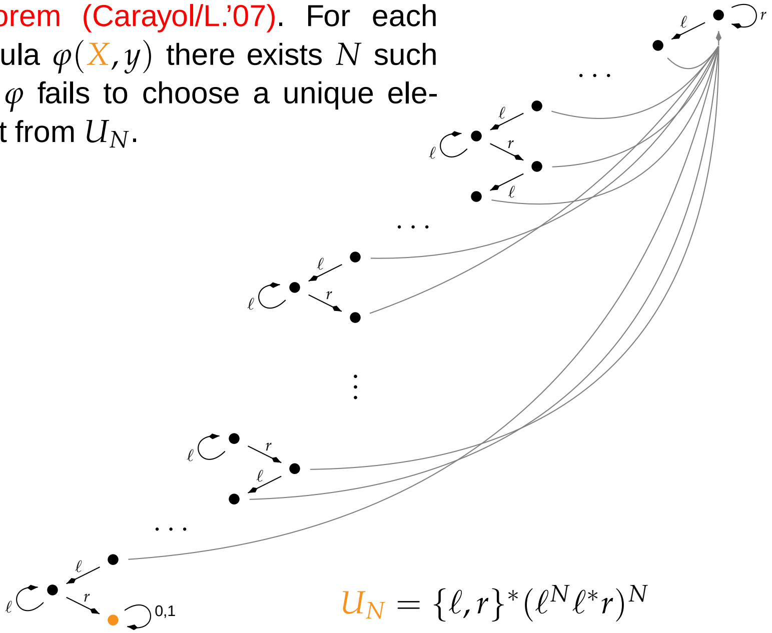
Counterexamples for MSO Choice



$$U_N = \{\ell, r\}^*(\ell^N \ell^* r)^N$$

Counterexamples for MSO Choice

Theorem (Carayol/L.'07). For each formula $\varphi(X, y)$ there exists N such that φ fails to choose a unique element from U_N.



$$U_N = \{\ell, r\}^*(\ell^N \ell^* r)^N$$

Automatic Relations over Different Structures

$$\begin{tikzcd}
\boxed{\text{finite words}} \arrow[r] \arrow[d] & \boxed{\text{finite trees}} \arrow[d] \\
\boxed{\text{infinite words}} \arrow[r] & \boxed{\text{infinite trees}}
\end{tikzcd}$$

Outline

Back to Infinite Words

- Infinite words are used to model the behavior of non-terminating processes (e.g. protocols, controllers)
- Such processes are usually reactive: they receive inputs and produce outputs

$$\begin{tikzcd}
\text{input stream} \arrow[r] & \boxed{\text{program } P} \arrow[r] & \text{output stream}
\end{tikzcd}$$

Back to Infinite Words

- Infinite words are used to model the behavior of non-terminating processes (e.g. protocols, controllers)
- Such processes are usually reactive: they receive inputs and produce outputs

$$\begin{tikzcd} \text{input stream} \arrow[r] & \boxed{\text{program } P} \arrow[r] & \text{output stream} \end{tikzcd}$$

- A specification relates the possible input sequences to allowed output sequences $\leadsto$ relation of infinite words, e.g., “each request is eventually followed by a grant”
- Uniformization $=$ Implementation?

Back to Infinite Words

- Infinite words are used to model the behavior of non-terminating processes (e.g. protocols, controllers)
- Such processes are usually reactive: they receive inputs and produce outputs

$$\begin{tikzcd}\text{input stream} \arrow[r] & \boxed{\text{program } P} \arrow[r] & \text{output stream}\end{tikzcd}$$

- A specification relates the possible input sequences to allowed output sequences $\rightsquigarrow$ relation of infinite words, e.g., “each request is eventually followed by a grant”
- Uniformization $=$ Implementation?
  No! An implementation has to produce the outputs “online”

Problem Setting

$$\text{input} \xrightarrow{\qquad\qquad} \boxed{\text{program } P} \xrightarrow{\qquad\qquad} \text{output}$$

- Alphabets I and J for input and output (assume here $I = J = \{0, 1\}$)
- MSO-formula $\varphi(X, Y)$ defines specification

Problem Setting

$$\begin{tikzcd} \text{input} \arrow[r] & \boxed{\text{program } P} \arrow[r] & \text{output} \end{tikzcd}$$

- Alphabets I and J for input and output (assume here $I = J = \{0,1\}$)
- MSO-formula $\varphi(X, Y)$ defines specification

Synthesis problem: Decide if the there is a program $P : I^* \longrightarrow J$ realizing φ, and construct one if possible.

Problem Setting

$$\text{input} \longrightarrow \boxed{\text{program } P} \longrightarrow \text{output}$$

- Alphabets I and J for input and output (assume here $I = J = \{0, 1\}$)
- MSO-formula $\varphi(X, Y)$ defines specification

Synthesis problem: Decide if the there is a program $P : I^* \to J$ realizing φ, and construct one if possible.

$X \quad a_0$

Y

Problem Setting

$$\text{input} \longrightarrow \boxed{\text{program } P} \longrightarrow \text{output}$$

- Alphabets I and J for input and output (assume here $I = J = \{0, 1\}$)
- MSO-formula $\varphi(X, Y)$ defines specification

Synthesis problem: Decide if the there is a program $P : I^* \longrightarrow J$ realizing φ, and construct one if possible.

$$\begin{tikzcd} X & \boxed{a_0} \arrow[d, bend left, "P"] \\ Y & b_0 \end{tikzcd}$$

Problem Setting

$$\begin{tikzcd} \text{input} \arrow[r] & \boxed{\text{program } P} \arrow[r] & \text{output} \end{tikzcd}$$

- Alphabets I and J for input and output (assume here $I = J = \{0, 1\}$)
- MSO-formula $\varphi(X, Y)$ defines specification

Synthesis problem: Decide if the there is a program $P : I^* \longrightarrow J$ realizing φ, and construct one if possible.

X	$a_0\ a_1$
Y	b_0

Problem Setting

$$\begin{tikzcd}
\text{input} \arrow[r] & \boxed{\text{program } P} \arrow[r] & \text{output}
\end{tikzcd}$$

- Alphabets I and J for input and output (assume here $I = J = \{0, 1\}$)
- MSO-formula $\varphi(X, Y)$ defines specification

Synthesis problem: Decide if the there is a program $P : I^* \longrightarrow J$ realizing φ, and construct one if possible.

$$\begin{tikzcd}
X & \boxed{a_0\ a_1} \arrow[d, bend left, "P"] \\
Y & b_0\ b_1
\end{tikzcd}$$

Problem Setting

$$\begin{tikzcd}\text{input} \arrow[r] & \boxed{\text{program } P} \arrow[r] & \text{output}\end{tikzcd}$$

- Alphabets I and J for input and output (assume here $I = J = \{0,1\}$)
- MSO-formula $\varphi(X,Y)$ defines specification

Synthesis problem: Decide if the there is a program $P : I^* \to J$ realizing φ, and construct one if possible.

$$\begin{array}{ll} X & a_0\ a_1\ a_2 \\ Y & b_0\ b_1 \end{array}$$

Problem Setting

$$\begin{tikzcd}\text{input} \arrow[r] & \boxed{\text{program } P} \arrow[r] & \text{output}\end{tikzcd}$$

- Alphabets I and J for input and output (assume here $I = J = \{0,1\}$)
- MSO-formula $\varphi(X,Y)$ defines specification

Synthesis problem: Decide if the there is a program $P : I^* \longrightarrow J$ realizing φ, and construct one if possible.

$$\begin{tikzcd}X & \boxed{a_0\ a_1\ a_2} \arrow[d, bend left, "P"] \\ Y & b_0\ b_1\ b_2\end{tikzcd}$$

Problem Setting

$$\text{input} \xrightarrow{\qquad\qquad} \boxed{\text{program } P} \xrightarrow{\qquad\qquad} \text{output}$$

- Alphabets I and J for input and output (assume here $I = J = \{0,1\}$)
- MSO-formula $\varphi(X,Y)$ defines specification

Synthesis problem: Decide if the there is a program $P : I^* \longrightarrow J$ realizing φ, and construct one if possible.

X	$a_0\ a_1\ a_2\ a_3$
Y	$b_0\ b_1\ b_2$

Problem Setting

$$\begin{tikzcd}\text{input} \arrow[r] & \boxed{\text{program } P} \arrow[r] & \text{output}\end{tikzcd}$$

- Alphabets I and J for input and output (assume here $I = J = \{0,1\}$)
- MSO-formula $\varphi(X,Y)$ defines specification

Synthesis problem: Decide if the there is a program $P : I^* \longrightarrow J$ realizing φ, and construct one if possible.

$$\begin{tikzcd} X & \boxed{a_0\ a_1\ a_2\ a_3} \arrow[d, bend left, "P"] \\ Y & b_0\ b_1\ b_2\ b_3 \end{tikzcd}$$

Problem Setting

$$\text{input} \longrightarrow \boxed{\text{program } P} \longrightarrow \text{output}$$

- Alphabets I and J for input and output (assume here $I = J = \{0, 1\}$)
- MSO-formula $\varphi(X, Y)$ defines specification

Synthesis problem: Decide if the there is a program $P : I^* \longrightarrow J$ realizing φ, and construct one if possible.

$X \quad a_0\ a_1\ a_2\ a_3\ a_4$

$Y \quad b_0\ b_1\ b_2\ b_3$

Problem Setting

$$\begin{tikzcd} \text{input} \arrow[r] & \boxed{\text{program } P} \arrow[r] & \text{output} \end{tikzcd}$$

- Alphabets I and J for input and output (assume here $I = J = \{0,1\}$)
- MSO-formula $\varphi(X,Y)$ defines specification

Synthesis problem: Decide if the there is a program $P : I^* \longrightarrow J$ realizing φ, and construct one if possible.

$$\begin{tikzcd} X & \boxed{a_0\ a_1\ a_2\ a_3\ a_4} \arrow[d, bend left, "P"] \\ Y & b_0\ b_1\ b_2\ b_3\ b_4 \end{tikzcd}$$

Problem Setting

$$\text{input} \longrightarrow \boxed{\text{program } P} \longrightarrow \text{output}$$

- Alphabets I and J for input and output (assume here $I = J = \{0, 1\}$)
- MSO-formula $\varphi(X, Y)$ defines specification

Synthesis problem: Decide if the there is a program $P : I^* \to J$ realizing φ, and construct one if possible.

$$\begin{array}{ll} X & a_0\ a_1\ a_2\ a_3\ a_4 \cdots \\ Y & b_0\ b_1\ b_2\ b_3\ b_4 \cdots \end{array} \models \varphi$$

Problem Setting

$$\begin{tikzcd}\text{input} \arrow[r] & \boxed{\text{program } P} \arrow[r] & \text{output}\end{tikzcd}$$

- Alphabets I and J for input and output (assume here $I = J = \{0,1\}$)
- MSO-formula $\varphi(X,Y)$ defines specification

Synthesis problem: Decide if the there is a program $P : I^* \to J$ realizing φ, and construct one if possible.

$$\left.\begin{array}{ll} X & a_0\ a_1\ a_2\ a_3\ a_4 \cdots \\ Y & b_0\ b_1\ b_2\ b_3\ b_4 \cdots \end{array}\right. \models \varphi$$

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

Example

$$\varphi(X,Y) =$$
$$\underbrace{\forall x\Big(X(x) \to \exists y > x\,(Y(y))\Big)}_{\text{each input } 1 \text{ later followed by output } 1} \wedge \underbrace{\forall x \exists y > x\,(\neg Y(y))}_{\text{infinitely often output } 0}$$

Example

$$\varphi(X,Y) = \underbrace{\forall x \Big(X(x) \to \exists y > x\,(Y(y)) \Big)}_{\text{each input } 1 \text{ later followed by output } 1} \wedge \underbrace{\forall x \exists y > x\,(\neg Y(y))}_{\text{infinitely often output } 0}$$

$$\wedge \underbrace{\forall y_1 < y_2 (Y(y_1) \wedge Y(y_2) \to \exists x\,(y_1 \leq x < y_2 \wedge X(x)))}_{\text{between two outputs 1 there is an input 1}}$$

Example

$$\varphi(X,Y) =$$

$$\underbrace{\forall x \Big(X(x) \rightarrow \exists y > x\,(Y(y)) \Big)}_{\text{each input } 1 \text{ later followed by output } 1} \wedge \underbrace{\forall x \exists y > x\,(\neg Y(y))}_{\text{infinitely often output } 0}$$

$$\wedge\, \underbrace{\forall y_1 < y_2 (Y(y_1) \wedge Y(y_2) \rightarrow \exists x\,(y_1 \leq x < y_2 \wedge X(x)))}_{\text{between two outputs 1 there is an input 1}}$$

Finite automata solution:

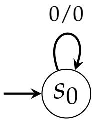
Example

$$\varphi(X,Y) =$$

$$\underbrace{\forall x\Big(X(x) \to \exists y > x\,(Y(y))\Big)}_{\text{each input 1 later followed by output 1}} \wedge \underbrace{\forall x \exists y > x\,(\neg Y(y))}_{\text{infinitely often output 0}}$$

$$\wedge \underbrace{\forall y_1 < y_2 (Y(y_1) \wedge Y(y_2) \to \exists x\,(y_1 \leq x < y_2 \wedge X(x)))}_{\text{between two outputs 1 there is an input 1}}$$

Finite automata solution:

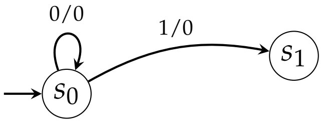


$$\begin{tikzcd} {} \arrow[r] & s_0 \arrow[loop above, "0/0"] \end{tikzcd}$$

Example

$$\varphi(X,Y) =$$
$$\underbrace{\forall x\Big(X(x) \rightarrow \exists y > x\,(Y(y))\Big)}_{\text{each input 1 later followed by output 1}} \wedge \underbrace{\forall x \exists y > x\,(\neg Y(y))}_{\text{infinitely often output 0}}$$
$$\wedge\ \underbrace{\forall y_1 < y_2 (Y(y_1) \wedge Y(y_2) \rightarrow \exists x\,(y_1 \leq x < y_2 \wedge X(x)))}_{\text{between two outputs 1 there is an input 1}}$$

Finite automata solution:


$$\begin{tikzcd}
\arrow[r] & s_0 \arrow[loop above, "0/0"] \arrow[r, bend left, "1/0"] & s_1
\end{tikzcd}$$

Example

$$\varphi(X,Y) =$$

$$\underbrace{\forall x \Big(X(x) \rightarrow \exists y > x\,(Y(y)) \Big)}_{\text{each input } 1 \text{ later followed by output } 1} \wedge \underbrace{\forall x \exists y > x\,(\neg Y(y))}_{\text{infinitely often output } 0}$$

$$\wedge \underbrace{\forall y_1 < y_2 (Y(y_1) \wedge Y(y_2) \rightarrow \exists x\,(y_1 \leq x < y_2 \wedge X(x)))}_{\text{between two outputs 1 there is an input 1}}$$

Finite automata solution:

$$\begin{tikzcd}
{} \arrow[r] & s_0 \arrow[loop above, "0/0"] \arrow[r, bend left, "1/0"] & s_1 \arrow[l, bend left, "0/1"]
\end{tikzcd}$$

Example

$$\varphi(X,Y) =$$

$$\underbrace{\forall x\Big(X(x) \to \exists y > x\,(Y(y))\Big)}_{\text{each input 1 later followed by output 1}} \wedge \underbrace{\forall x \exists y > x\,(\neg Y(y))}_{\text{infinitely often output 0}}$$

$$\wedge\ \underbrace{\forall y_1 < y_2 (Y(y_1) \wedge Y(y_2) \to \exists x\,(y_1 \leq x < y_2 \wedge X(x)))}_{\text{between two outputs 1 there is an input 1}}$$

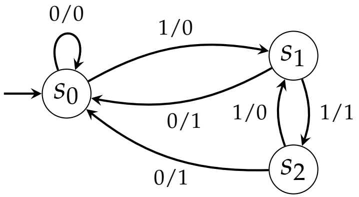
Finite automata solution:

$$\begin{tikzcd}
{} \arrow[r] & s_0 \arrow[loop above, "0/0"] \arrow[r, bend left, "1/0"] & s_1 \arrow[l, bend left, "0/1"] \arrow[d, "1/1"] \\
& & s_2
\end{tikzcd}$$

Example

$$\varphi(X,Y) =$$
$$\underbrace{\forall x\Big(X(x) \to \exists y > x\,(Y(y))\Big)}_{\text{each input 1 later followed by output 1}} \wedge \underbrace{\forall x \exists y > x\,(\neg Y(y))}_{\text{infinitely often output 0}}$$
$$\wedge\ \underbrace{\forall y_1 < y_2 (Y(y_1) \wedge Y(y_2) \to \exists x\,(y_1 \leq x < y_2 \wedge X(x)))}_{\text{between two outputs 1 there is an input 1}}$$

Finite automata solution:



Büchi-Landweber Theorem

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

Proof idea:

- View problem as game between players Input and Output: game with simple rules (players play bits in alternation) but a complex winning condition (the formula)
- Reduction to a game with more complex rules but much simpler winning condition

Continuous Functions

$\{0,1\}^\omega$ as metric space: $d(\alpha,\beta) = \frac{1}{2^n}$

where n is the length of the longest common prefix of α and β

$$\begin{array}{ll} \alpha & 0\,0\,1\,0\,0\,0\,1\,\textcolor{red}{1}\,0\,0 \quad \cdots \\ \beta & 0\,0\,1\,0\,0\,0\,1\,\textcolor{red}{0}\,1\,0 \quad \cdots \end{array} \qquad d(\alpha,\beta) = \frac{1}{2^7}$$

Continuous Functions

$\{0,1\}^\omega$ as metric space: $d(\alpha,\beta) = \frac{1}{2^n}$

where n is the length of the longest common prefix of α and β

$$\begin{array}{ll} \alpha & 0\,0\,1\,0\,0\,0\,1\,1\,0\,0\ \cdots \\ \beta & 0\,0\,1\,0\,0\,0\,1\,0\,1\,0\ \cdots \end{array} \qquad d(\alpha,\beta) = \frac{1}{2^7}$$

Previous synthesis question: exists program (function) P with

$$d(P(\alpha),P(\beta)) \leq d(\alpha,\beta)?$$

Continuous Functions

$\{0,1\}^\omega$ as metric space: $d(\alpha,\beta) = \frac{1}{2^n}$

where n is the length of the longest common prefix of α and β

$$\begin{array}{ll} \alpha & 0\,0\,1\,0\,0\,0\,1\,1\,0\,0\ \cdots \\ \beta & 0\,0\,1\,0\,0\,0\,1\,0\,1\,0\ \cdots \end{array} \qquad d(\alpha,\beta) = \frac{1}{2^7}$$

Previous synthesis question: exists program (function) P with

$$d(P(\alpha),P(\beta)) \leq d(\alpha,\beta)?$$

Generalized question: exists program (function) P realizing a continuous function: Each finite prefix of the output only depends on a finite prefix of the input.

Delaying the Output

I

J

Delaying the Output

I a_0

J

Delaying the Output

I	a_0
J	b_0

Delaying the Output

I	$a_0\ a_1$
J	b_0

Delaying the Output

I	$a_0\ a_1$
J	b_0 skip

Delaying the Output

I $a_0\ a_1\ a_2$

J b_0

Delaying the Output

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

Delaying the Output

I	$a_0\ a_1\ a_2\ a_3$
J	$b_0\ b_1$

Delaying the Output

I $a_0\ a_1\ a_2\ a_3$

J $b_0\ b_1$ skip

Delaying the Output

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

J $b_0\ b_1$

Delaying the Output

I	$a_0\ a_1\ a_2\ a_3\ a_4$
J	$b_0\ b_1\ b_2$

Delaying the Output

$$\begin{array}{ll} I & a_0\ a_1\ a_2\ a_3\ a_4 \cdots \\ J & b_0\ b_1\ b_2 \cdots \end{array} \models \varphi$$

Delaying the Output

$$\begin{array}{ll} I & a_0\ a_1\ a_2\ a_3\ a_4 \cdots \\ J & b_0\ b_1\ b_2 \cdots \end{array} \models \varphi$$

Question:

Given a specification, does there exist a strategy (program) $P : I^* \longrightarrow (J \cup \{\text{skip}\})$ that

- produces an infinite output sequence for each input and
- realizes the specification?

Delaying the Output

$$\begin{array}{lll} I & a_0\ a_1\ a_2\ a_3\ a_4 \cdots & \\ & & \models \varphi \\ J & b_0\ b_1\ b_2 \cdots & \end{array}$$

Question:

Given a specification, does there exist a strategy (program) $P : I^* \to (J \cup \{\text{skip}\})$ that

- produces an infinite output sequence for each input and
- realizes the specification?

Can the specification be uniformized by a continuous function?

Examples

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

- $\varphi(X, Y) = \forall x (Y(x) \leftrightarrow X(x + 1))$

Examples

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

- $\varphi(X, Y) = \forall x (Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

  I

  J

Examples

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

- $\varphi(X, Y) = \forall x(Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

I	0
J	

Examples

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

- $\varphi(X, Y) = \forall x (Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

I	0
J	skip

Examples

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

- $\varphi(X,Y) = \forall x(Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

I	0	0
J		

Examples

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

- $\varphi(X,Y) = \forall x(Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

I	0	0
J	0	

Examples

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

- $\varphi(X,Y) = \forall x(Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

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

Examples

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

- $\varphi(X,Y) = \forall x(Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

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

Examples

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

- $\varphi(X, Y) = \forall x(Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

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

- $\exists x(X(x) \rightarrow Y(\underline{0})) \wedge \forall x(\neg X(x) \rightarrow \neg Y(\underline{0}))$
  “start output with 1 iff there is 1 somewhere in the input”

Examples

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

- $\varphi(X, Y) = \forall x (Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

I	0	0	$1 \cdots$
J	0	$1 \cdots$	

- $\exists x (X(x) \rightarrow Y(\underline{0})) \wedge \forall x (\neg X(x) \rightarrow \neg Y(\underline{0}))$
  "start output with 1 iff there is 1 somewhere in the input"

  There is no strategy with delay for this specification.

  I

  J

Examples

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

- $\varphi(X, Y) = \forall x(Y(x) \leftrightarrow X(x + 1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

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

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

- $\exists x(X(x) \rightarrow Y(\underline{0})) \wedge \forall x(\neg X(x) \rightarrow \neg Y(\underline{0}))$
  "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

Examples

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

- $\varphi(X, Y) = \forall x(Y(x) \leftrightarrow X(x + 1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

  $I \qquad 0 \;\; 0 \;\; 1 \cdots$

  $J \qquad 0 \;\; 1 \cdots$

- $\exists x(X(x) \rightarrow Y(\underline{0})) \wedge \forall x(\neg X(x) \rightarrow \neg Y(\underline{0}))$
  “start output with 1 iff there is 1 somewhere in the input”

  There is no strategy with delay for this specification.

  $I \qquad 0$

  J skip

Examples

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

- $\varphi(X,Y) = \forall x(Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

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

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

- $\exists x(X(x) \rightarrow Y(\underline{0})) \wedge \forall x(\neg X(x) \rightarrow \neg Y(\underline{0}))$
  “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

Examples

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

- $\varphi(X,Y) = \forall x(Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

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

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

- $\exists x(X(x) \rightarrow Y(\underline{0})) \wedge \forall x(\neg X(x) \rightarrow \neg Y(\underline{0}))$
  “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

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

- $\varphi(X, Y) = \forall x(Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

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

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

- $\exists x(X(x) \rightarrow Y(\underline{0})) \wedge \forall x(\neg X(x) \rightarrow \neg Y(\underline{0}))$
  “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

Examples

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

- $\varphi(X, Y) = \forall x(Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

I	0	0	$1 \cdots$
J	0	$1 \cdots$	

- $\exists x(X(x) \rightarrow Y(\underline{0})) \wedge \forall x(\neg X(x) \rightarrow \neg Y(\underline{0}))$
  "start output with 1 iff there is 1 somewhere in the input"

  There is no strategy with delay for this specification.

I	0	0	0
J	skip		

Examples

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

- $\varphi(X,Y) = \forall x(Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

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

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

- $\exists x(X(x) \rightarrow Y(\underline{0})) \wedge \forall x(\neg X(x) \rightarrow \neg Y(\underline{0}))$
  "start output with 1 iff there is 1 somewhere in the input"

  There is no strategy with delay for this specification.

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

  J

Examples

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

- $\varphi(X,Y) = \forall x(Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

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

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

- $\exists x(X(x) \rightarrow Y(\underline{0})) \wedge \forall x(\neg X(x) \rightarrow \neg Y(\underline{0}))$
  “start output with 1 iff there is 1 somewhere in the input”

  There is no strategy with delay for this specification.

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

  $J \quad 0$

Examples

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

- $\varphi(X,Y) = \forall x(Y(x) \leftrightarrow X(x+1))$
  Output has to skip once at the beginning and then plays 1 iff the input is 1.

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

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

- $\exists x(X(x) \rightarrow Y(\underline{0})) \wedge \forall x(\neg X(x) \rightarrow \neg Y(\underline{0}))$
  "start output with 1 iff there is 1 somewhere in the input"

  There is no strategy with delay for this specification.

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

  $J \quad 0$

Bounded Delay in Regular Games

Theorem (Hosch/Landweber'72,Holtmann/Kaiser/Thomas'10). For MSO 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 MSO definable relation over infinite words can be uniformized by a continuous function.

Beyond Finite Automata

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

Without Delay:

Theorem (Walukiewicz'96). The 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	0
J	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:

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

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
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 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
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	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			

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:

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

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:

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

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:

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

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.

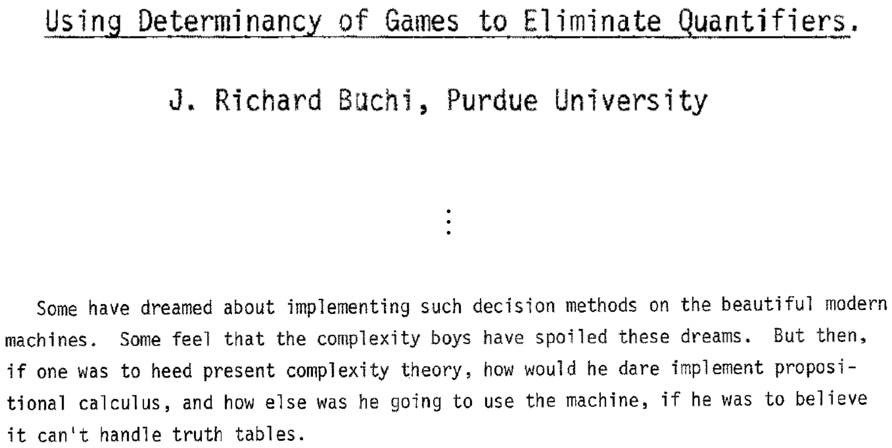
Conclusion

Summary

- Uniformization in automata theory: effective theory of uniformization for relations definable by finite automata (or equivalently in MSO logic)
- Connection between synthesis of nonterminating programs and uniformization by continuous functions

Outlook

- Many more models to study: asynchronous automata, non-regular specifications
- Uniformization on infinite trees: decidability questions
- Implementation: make synthesis algorithms usable in practice



Using Determinancy of Games to Eliminate Quantifiers.

J. Richard Büchi, Purdue University

⋮

Some have dreamed about implementing such decision methods on the beautiful modern machines. Some feel that the complexity boys have spoiled these dreams. But then, if one was to heed present complexity theory, how would he dare implement propositional calculus, and how else was he going to use the machine, if he was to believe it can't handle truth tables.