Synthesis in infinite structures

Sophie Pinchinat

IRISA, Univ Rennes

WAS ESSLLI, 30 July 2021

Motivations

The usefulness of (automated) synthesis for time saving, correctness by construction, etc.

Automated synthesis is a very broad area.

Motivations

The usefulness of (automated) synthesis for time saving, correctness by construction, etc.

Automated synthesis is a very broad area.

If we focus on agency:

- What to start with? *i.e.* problem inputs.

Motivations

The usefulness of (automated) synthesis for time saving, correctness by construction, etc.

Automated synthesis is a very broad area.

If we focus on agency:

- What to start with? *i.e.* problem inputs.
  - Domain specification: which sort of models? transition systems, qualitative/quantitative, extensional/intentional representation, etc.
  - Requirements specification: logical formula, set of examples/counter-examples, etc.

Motivations

The usefulness of (automated) synthesis for time saving, correctness by construction, etc.

Automated synthesis is a very broad area.

If we focus on agency:

- What to start with? *i.e.* problem inputs.
  - Domain specification: which sort of models? transition systems, qualitative/quantitative, extensional/intentional representation, etc.
  - Requirements specification: logical formula, set of examples/counter-examples, etc.
- What to return? *i.e.* problem output(s).

Motivations

The usefulness of (automated) synthesis for time saving, correctness by construction, etc.

Automated synthesis is a very broad area.

If we focus on agency:

- What to start with? *i.e.* problem inputs.
  - Domain specification: which sort of models? transition systems, qualitative/quantitative, extensional/intentional representation, etc.
  - Requirements specification: logical formula, set of examples/counter-examples, etc.
- What to return? *i.e.* problem output(s).
  - A "solution";
  - Constrained solutions;
  - All solutions (what if they are finitely many?)

Synthesis in this talk

- Inputs:
  - Relational structures, possibly infinite to capture dynamics, knowledge, etc.

Synthesis in this talk

- Inputs:
  - Relational structures, possibly infinite to capture dynamics, knowledge, etc.
  - A logical formula in (a fragment of) classical logic

$$\mathrm{FO} \text{ and } \mathrm{MSO}$$

Synthesis in this talk

- Inputs:
  - Relational structures, possibly infinite to capture dynamics, knowledge, etc.
  - A logical formula in (a fragment of) classical logic

    $$\mathrm{FO} \text{ and } \mathrm{MSO}$$

    to capture a lot, *e.g.* temporal logics CTL* or mu-calculus, knowledge and time CTL*K or epistemic mu-calculus, etc.

Synthesis in this talk

- Inputs:
  - Relational structures, possibly infinite to capture dynamics, knowledge, etc.
  - A logical formula in (a fragment of) classical logic

    $$\mathrm{FO} \text{ and } \mathrm{MSO}$$

    to capture a lot, *e.g.* temporal logics CTL* or mu-calculus, knowledge and time CTL*K or epistemic mu-calculus, etc.
- Output(s): set of assignments of free variables

Synthesis in this talk

- Inputs:
  - Relational structures, possibly infinite to capture dynamics, knowledge, etc.
  - A logical formula in (a fragment of) classical logic

    $$\mathrm{FO} \text{ and } \mathrm{MSO}$$

    to capture a lot, *e.g.* temporal logics CTL* or mu-calculus, knowledge and time CTL*K or epistemic mu-calculus, etc.
- Output(s): set of assignments of free variables
  - If closed formula, output is the model checking verdict;

Outline of the talk

Outline

(Relational) structures

Example

- Natural numbers $\mathcal{S} = \langle \mathbb{N}, \leq \rangle$
- Graphs $\mathcal{G} = \langle V, E \rangle$
- Transition systems $TS = \langle S, S_0, \rightarrow, \{p\}_{p \in Prop} \rangle$
- Trees $\mathcal{T} = \langle D, r, Succ_1, \ldots, Succ_n, R_1, \ldots, R_p \rangle$

Definition (Relational structure)

$\mathcal{S} = \langle D, R_1 \ldots R_p \rangle$ where

- $D \neq \emptyset$ is the *domain*
- $R_i \subseteq D^{r_i}$

Write $R_i(d_1, \ldots, d_{r_i})$ for $(d_1, \ldots, d_{r_i}) \in R_i$

Zoom on bounded-degree Trees

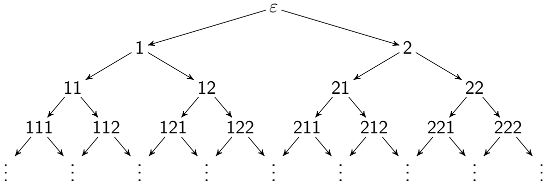
Definition (Tree structures)

$\mathcal{T} = \langle D, r, Succ_1, \ldots, Succ_n, R_1, \ldots, R_p \rangle$

- $D \subseteq \{1, \ldots, n\}^*$ prefix-closed (node addresses)
- $r \stackrel{def}{=} \{\varepsilon\}$ (being the root)
- $Succ_j \stackrel{def}{=} \{(u, u.j) \mid u.j \in D\}$ (j-th child)
- $+$ other relations $R_1, \ldots, R_p$ over D

Full bounded-degree infinite trees

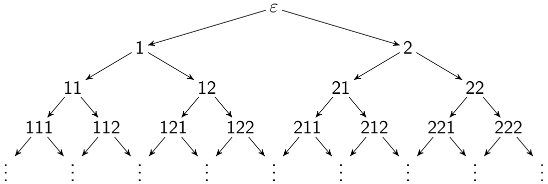
- $\mathcal{T}_2 = \langle \{1,2\}^*, Succ_1, Succ_2 \rangle$



$Succ_i(u, u.i)$ for every $u \in \{1,2\}^*$

Full bounded-degree infinite trees

- $\mathcal{T}_2 = \langle \{1,2\}^*, Succ_1, Succ_2 \rangle$

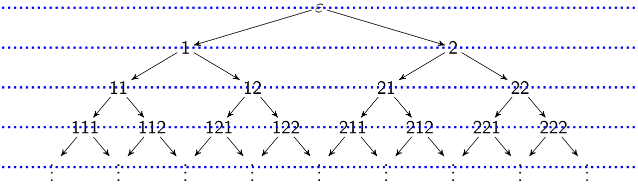


$Succ_i(u, u.i)$ for every $u \in \{1,2\}^*$

- $\mathcal{T}_2^{\mathrm{el}} = \langle \{1,2\}^*, Succ_1, Succ_2, \mathrm{el} \rangle$

Full bounded-degree infinite trees

- $\mathcal{T}_2 = \langle \{1,2\}^*, Succ_1, Succ_2 \rangle$



$Succ_i(u, u.i)$ for every $u \in \{1,2\}^*$

- $\mathcal{T}_2^{\mathsf{el}} = \langle \{1,2\}^*, Succ_1, Succ_2, \mathsf{el} \rangle$ with "equal level" (binary) relation.

Outline

Logics FO and MSO

- $\mathcal{V}_1 = \{x, x_1, x_2, \ldots\}$ set of first-order variables.

$$\mathrm{FO} \ni \varphi, \psi ::= R_i(x_1 \ldots x_{r_i}) \mid \neg\varphi \mid \varphi \wedge \psi \mid \exists x \varphi$$

- $\mathcal{V}_2 = \{X, X_1, \ldots, Y, \ldots\}$ set of second-order variables:

$$\mathrm{MSO} \ni \Phi ::= R_i(x_1 \ldots x_{r_i}) \mid \neg\Phi \mid \Phi \wedge \Psi \mid \exists x \Phi \mid x \in X \mid \exists X \Phi$$

Logics FO and MSO

- $\mathcal{V}_1 = \{x, x_1, x_2, \ldots\}$ set of first-order variables.

$$\mathrm{FO} \ni \varphi, \psi ::= R_i(x_1 \ldots x_{r_i}) \,|\, \neg\varphi \,|\, \varphi \wedge \psi \,|\, \exists x \varphi$$

- $\mathcal{V}_2 = \{X, X_1, \ldots, Y, \ldots\}$ set of second-order variables:

$$\cancel{\mathrm{MSO} \ni \Phi ::= R_i(x_1 \ldots x_{r_i}) \,|\, \neg\Phi \,|\, \Phi \wedge \Psi \,|\, \exists x \Phi \,|\, x \in X \,|\, \exists X \Phi}$$

$$\mathrm{MSO} \ni \Phi ::= \mathsf{Sing}(X) \,|\, X \subseteq Y \,|\, R_i(X_1 \ldots X_{r_i}) \,|\, \neg\Phi \,|\, \Phi \wedge \Psi \,|\, \exists X \Phi$$

Logics FO and MSO

- $\mathcal{V}_1 = \{x, x_1, x_2, \ldots\}$ set of first-order variables.

$$\mathrm{FO} \ni \varphi, \psi ::= R_i(x_1 \ldots x_{r_i}) \mid \neg\varphi \mid \varphi \wedge \psi \mid \exists x \varphi$$

- $\mathcal{V}_2 = \{X, X_1, \ldots, Y, \ldots\}$ set of second-order variables:

$$\cancel{\mathrm{MSO} \ni \Phi ::= R_i(x_1 \ldots x_{r_i}) \mid \neg\Phi \mid \Phi \wedge \Psi \mid \exists x \Phi \mid x \in X \mid \exists X \Phi}$$

$$\mathrm{MSO} \ni \Phi ::= \mathsf{Sing}(X) \mid X \subseteq Y \mid R_i(X_1 \ldots X_{r_i}) \mid \neg\Phi \mid \Phi \wedge \Psi \mid \exists X \Phi$$

We also will consider

$$\textsc{chain}\mathrm{MSO}\text{: same syntax as } \mathrm{MSO}$$

but over tree structures and where interpretation of second order variables X is restricted to chains (see later).

Here synthesis problems are seen as functions

Fix a class $\mathbb{C}$ of relational structures.

Definition ($\textsc{Synth}(\mathbb{C},\mathrm{FO})$)

$$\begin{cases} \textbf{In: } \text{A finite description of } \mathcal{S} \in \mathbb{C},\ \varphi(x_1,\ldots,x_k) \in \mathrm{FO} \\ \textbf{Out: } \varphi^{\mathcal{S}} \stackrel{def}{=} \{(e_1,\ldots,e_k) \in D^k \mid \mathcal{S},[\vec{x} := \vec{e}] \models \varphi(x_1,\ldots,x_k)\} \end{cases}$$

Remark (Model checking subsumption)

If $\varphi(x_1,\ldots,x_k)$ has no free variables (i.e. $k=0$), ouput set $\subseteq D^0$: output is either the full set or the empty set.

$\Rightarrow$ *Synthesis becomes Model Cheking, i.e.*

$$\textbf{Out: } \mathcal{S} \models \varphi?$$

Here synthesis problems are seen as functions

Fix a class $\mathbb{C}$ of relational structures.

Definition ($\textsc{Synth}(\mathbb{C},\mathrm{FO})$)

$$\begin{cases} \textbf{In: } \text{A finite description of } \mathcal{S} \in \mathbb{C},\ \varphi(x_1,\ldots,x_k) \in \mathrm{FO} \\ \textbf{Out: } \varphi^{\mathcal{S}} \stackrel{def}{=} \{(e_1,\ldots,e_k) \in D^k \mid \mathcal{S},[\vec{x} := \vec{e}] \models \varphi(x_1,\ldots,x_k)\} \end{cases}$$

Definition ($\textsc{Synth}(\mathbb{C},\mathrm{MSO})$)

$$\begin{cases} \textbf{In: } \text{A finite description of } \mathcal{S} \in \mathbb{C},\ \Phi(X_1,\ldots,X_m) \in \mathrm{MSO} \\ \textbf{Out: } \Phi^{\mathcal{S}} \stackrel{def}{=} \{(E_1,\ldots,E_m) \in (2^D)^m \mid \mathcal{S},[\vec{X} := \vec{E}] \models \Phi(X_1,\ldots,X_m)\} \end{cases}$$

Here synthesis problems are seen as functions

Fix a class $\mathbb{C}$ of relational structures.

Definition ($\textsc{Synth}(\mathbb{C},\mathrm{FO})$)

$$\begin{cases} \textbf{In: } \text{A finite description of } \mathcal{S} \in \mathbb{C},\ \varphi(x_1,\dots,x_k) \in \mathrm{FO} \\ \textbf{Out: } \varphi^{\mathcal{S}} \stackrel{def}{=} \{(e_1,\dots,e_k) \in D^k \mid \mathcal{S},[\vec{x} := \vec{e}] \models \varphi(x_1,\dots,x_k)\} \end{cases}$$

Definition ($\textsc{Synth}(\mathbb{C},\mathrm{MSO})$)

$$\begin{cases} \textbf{In: } \text{A finite description of } \mathcal{S} \in \mathbb{C},\ \Phi(X_1,\dots,X_m) \in \mathrm{MSO} \\ \textbf{Out: } \Phi^{\mathcal{S}} \stackrel{def}{=} \{(E_1,\dots,E_m) \in (2^D)^m \mid \mathcal{S},[\vec{X} := \vec{E}] \models \Phi(X_1,\dots,X_m)\} \end{cases}$$

We similarly define $\textsc{Synth}(\mathbb{C},\textsc{chain}\mathrm{MSO})$.

What do we mean by **In:** and **Out:** ?

Structures in $\mathbb{C}$ (**In:**) and output sets (**Out:**) should be representable in a finite way, if not themselves already finite:

- by a binary string, or
- by an algorithm, or
- by a collection of automata, or
- by an axiomatisation in some logic, or
- by an interpretation, or
- etc.

We will focus on automata collections.

Synthesis in $\mathbb{F}$ (finite structures)

Theorem (Stockmeyer 1974, Vardi 1982)

Model-checking over $\mathbb{F}$ *against* FO *and* MSO *is* PSPACE*-complete**.

(*) If class $\mathbb{C}$ contains a structure with at least two elements.

Corollary

SYNTH($\mathbb{F}$,FO) *and* SYNTH($\mathbb{F}$,MSO) *are computable.*

What can we do with infinite structures?

Synthesis in class $\mathbb{C}$ with infinite structures

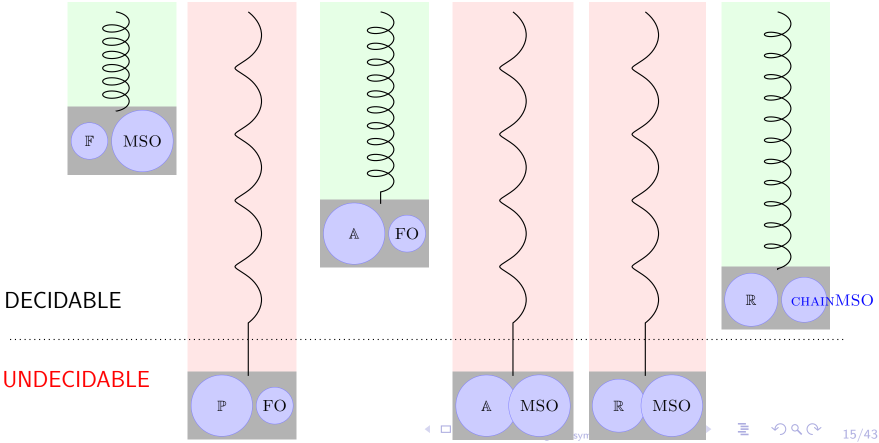
We consider special cases for $\mathbb{C}$:

- Post Correspondance Structures ($\mathbb{P}$)
- Automatic Structures ($\mathbb{A}$)
- Regular automatic trees ($\mathbb{R}$)

Synthesis in infinite structures

$\mathbb{P}$ (PCP structures)

$\mathbb{A}$ (Automatic structures), $\mathbb{R}$ (Regular automatic trees)



Outline

1. Motivations
2. Background
3. Synthesis Problem(s)
4. Synthesis in infinite Structures
   - Class of Post Correspondance Problem structures
   - Class of Automatic structures
   - Class of Regular automatic trees
5. Concluding remarks

The class $\mathbb{P}$

Definition (Post Correspondance Problem (PCP))

In: A finite set of dominoes $\Delta = \{\binom{u_i}{v_i}\}_{i=1,\ldots,n}$ where $u_i, v_i \in \{a, b\}^*$

Out: Does there exists a solution, *i.e.* a sequence $i_1, \ldots, i_k$ of dominoes s.t. $u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$?

Two predicate symbols: `nonEmpty` (monadic) and `dominoes` (binary).

Definition (Structures of $\mathbb{P}$)

Structures of the form $\mathcal{S}_\Delta = \langle \{a, b\}^*, \mathtt{nonEmpty}^{\mathcal{S}_\Delta}, \mathtt{dominoes}^{\mathcal{S}_\Delta} \rangle$, where

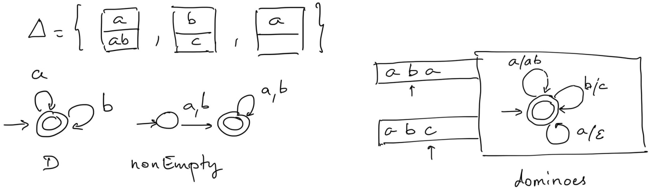
- Δ is a finite set of dominoes;
- $\mathtt{nonEmpty}^{\mathcal{S}_\Delta} \ni u$ whenever u is a non-empty word;
- $\mathtt{dominoes}^{\mathcal{S}_\Delta} = \Delta^* \ni \binom{u}{v}$ whenever u and v are upper and the lower part of some domino concatenation.

Automata-based finite presentation of structures in $\mathbb{P}$

Each structure $\mathcal{S}_\Delta = \langle \{a, b\}^*, \texttt{nonEmpty}^{\mathcal{S}_\Delta}, \texttt{dominoes}^{\mathcal{S}_\Delta} \rangle$ can be finitely presented with finite-state (multi-tape) automata:

- One-tape automaton for the domain $\{a, b\}^*$;
- One-tape automaton for $\texttt{nonEmpty}^{\mathcal{S}_\Delta} = \{a, b\}^* \setminus \{\varepsilon\}$;
- A two-tape automaton for Δ^*

Ex:



$\textsc{Synth}(\mathbb{P},\mathrm{FO})$

Theorem

$\textsc{Synth}(\mathbb{P},\mathrm{FO})$ *is not computable.*

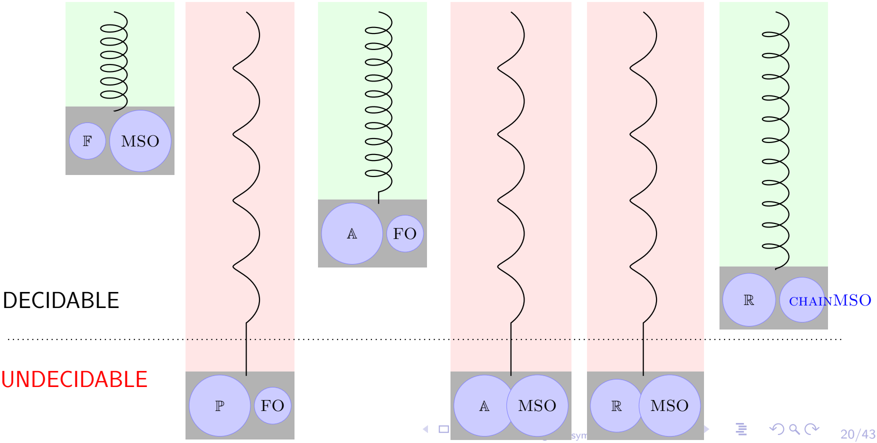
Reduction from PCP (undecidable (Post, 1946)):

$$\Delta \text{ has a solution} \quad \text{iff} \quad \mathcal{S}_\Delta \models \exists x(\mathtt{nonEmpty}(x) \wedge \mathtt{dominoes}(x,x))$$

Synthesis in infinite structures

$\mathbb{P}$ (PCP structures)

$\mathbb{A}$ (Automatic structures), $\mathbb{R}$ (Regular automatic trees)



Outline

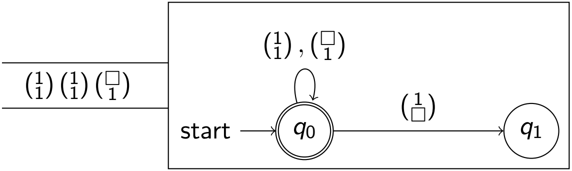
Describe $\langle \mathbb{N}, \leq \rangle$ with automata

$$0 \longrightarrow 1 \longrightarrow 2 \longrightarrow 3 \longrightarrow \cdots$$

- Encode each $n \in \mathbb{N}$ by $enc(n) = \overbrace{11\ldots1}^{n}$
- Encode pairs as a word over alphabet $(\Sigma_\Box)^2$:

  $1^2 \otimes 1^3 := \binom{1}{1}\binom{1}{1}\binom{\Box}{1}$ (the convolution of 1^2 and 1^3)

  Automaton $\mathcal{A}_{\leq}$:



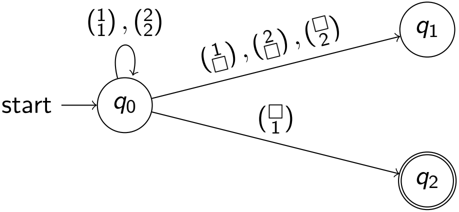
  - $1^2 \otimes 1^3 \in \mathcal{L}(\mathcal{A}_{\leq})$
  - $1^3 \otimes 1^2 = \binom{1}{1}\binom{1}{1}\binom{1}{\Box} \notin \mathcal{L}(\mathcal{A}_{\leq})$
  - $1^n \otimes 1^m$ is accepted by $\mathcal{A}_{\leq}$ iff $n \leq m$.

Binary infinite tree $\mathcal{T}_2$ with "equal level" using automata

Recall $\mathcal{T}_2^{\mathrm{el}} = \langle \{1,2\}^*, Succ_1, Succ_2, \mathrm{el} \rangle$

- Node encoding is the address $u \in \{1,2\}^*$
- $Succ_1(u,v)$ iff $v = u.1$

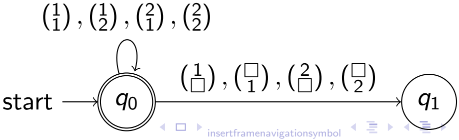
$\mathcal{A}_{Succ_1}$:



Similary for $\mathcal{A}_{Succ_2}$...

- $\mathrm{el}(u,v)$ iff $|u| = |v|$

$\mathcal{A}_{\mathrm{el}}$:



Structures with an automatic presentation: class $\mathbb{A}$

Definition ((Khoussainov et al., 2007; Blumensath and Grädel, 2000))

A structure $\mathcal{S} = \langle D, R_1 \dots R_p \rangle$ is automatic if it has an automatic presentation $(\mathcal{A}_D, \mathcal{A}_1, \dots, \mathcal{A}_p)$ where

- $(\mathcal{A}_D, \mathcal{A}_1, \dots, \mathcal{A}_p)$ is a tuple of (finite-state) automata;
- there is a (bijective) encoding function $enc : D \to \mathcal{L}(\mathcal{A}_D)$;
- relation R_i is encoded by $\mathcal{L}(\mathcal{A}_i)$:

$$u_1 \otimes \cdots \otimes u_{r_i} \in \mathcal{L}(\mathcal{A}_i)$$
$$\text{iff}$$
$$(u_1, \dots, u_{r_i}) \in enc(R_i)$$

where $enc(R_i) = \{(enc(e_1), \dots, enc(e_{r_i})) \mid (e_1, \dots, e_{r_i}) \in R_i\}$

$\textsc{Synth}(\mathbb{A},\mathrm{FO})$

Definition ($\textsc{Synth}(\mathbb{A},\mathrm{FO})$)

In: An automatic presentation $(\mathcal{A}_D, \mathcal{A}_1, \ldots, \mathcal{A}_p)$ of $\mathcal{S} \in \mathbb{A}$, and $\varphi(x_1, \ldots, x_k) \in \mathrm{FO}$

Out: $\varphi^{\mathcal{S}} \stackrel{def}{=} \{(e_1, \ldots, e_k) \in D^k \mid \mathcal{S}, [\vec{x} := \vec{e}] \models \varphi(x_1, \ldots, x_k)\}$

Theorem

$\textsc{Synth}(\mathbb{A},\mathrm{FO})$ *is computable*

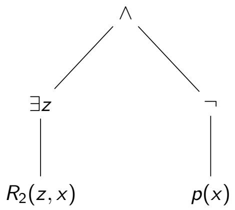
Build automaton $\mathcal{A}_\varphi$ with $\mathcal{L}(\mathcal{A}_\varphi) = \varphi^{\mathcal{S}}$ (actually $enc(\varphi^{\mathcal{S}})$)

Inductively over φ:

Formula	Automaton
$R_i(x_1 \ldots x_{r_i})$	the given $\mathcal{A}_i$ of $\mathcal{S}$
$\neg\varphi$	the complement of $\mathcal{A}_\varphi$
$\varphi \wedge \psi$	the product of $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$
$\exists x \varphi$	component abstract from $\mathcal{A}_\varphi$

Bottom-up construction of $\mathcal{A}_\varphi$: intuitive example

$$\varphi(x) := \exists z R_2(z,x) \wedge \neg p(x)$$



1. $\mathcal{A}_{\exists z R_2(x,z)}$: obtained by abstracting the second component of $\mathcal{A}_{R_2(x,z)}$ (given by the automatic presentation);
2. $\mathcal{A}_{\neg p(x)}$: obtained as $\mathcal{A}_D \cap \mathcal{A}^c_{p(x)}$;
3. $\mathcal{A}_{\exists z R_2(z,x) \wedge \neg p(x)} \stackrel{def}{=} \mathcal{A}_{\exists z R_2(x,z)} \cap \mathcal{A}_{\neg p(x)}$.

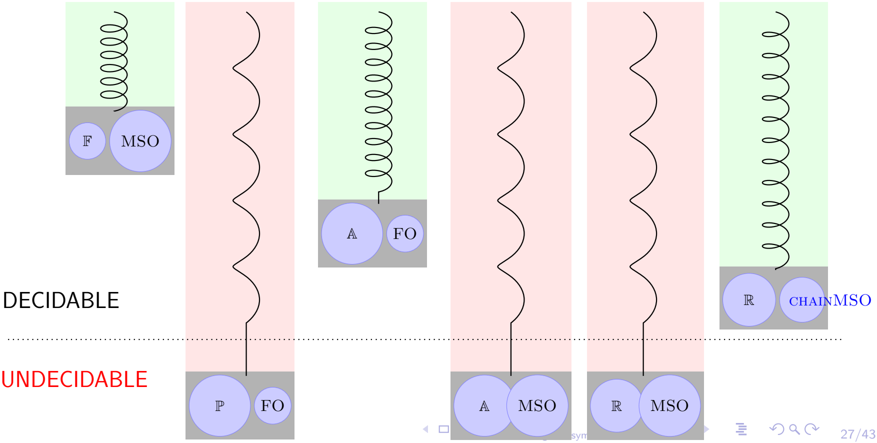
$$\varphi^{\mathcal{S}} := \{e \in D \mid \mathcal{S}, [x \mapsto e] \models \varphi(x)\}$$

$$\mathcal{L}(\mathcal{A}_{\varphi(x)}) = \{enc(e) \mid e \in \varphi^{\mathcal{S}}\}.$$

Synthesis in infinite structures

$\mathbb{P}$ (PCP structures)

$\mathbb{A}$ (Automatic structures), $\mathbb{R}$ (Regular automatic trees)



$\textsc{Synth}(\mathbb{A},\mathrm{MSO})$

$$\mathrm{MSO} \ni \Phi, \Psi ::= \mathsf{Sing}(X) \mid X \subseteq Y \mid R_i(X_1 \dots X_{r_i}) \mid \neg\Phi \mid (\Phi \wedge \Psi) \mid \exists X \Phi$$

Definition ($\textsc{Synth}(\mathbb{A},\mathrm{MSO})$)

$$\begin{cases} \textbf{In:} & \text{An automatic presentation } (\mathcal{A}_D, \mathcal{A}_1, \dots, \mathcal{A}_p) \text{ of } \mathcal{S} \in \mathbb{A}, \text{ and } \Phi(X_1, \dots, X_m) \in \mathrm{MSO} \\ \textbf{Out:} & \Phi^{\mathcal{S}} \stackrel{def}{=} \{(E_1, \dots, E_m) \in (2^D)^m \mid \mathcal{S}, [\vec{X} := \vec{E}] \models \Phi(X_1, \dots, X_m)\} \end{cases}$$

Theorem

$\textsc{Synth}(\mathbb{A},\mathrm{MSO})$ *is not computable.*

A corollary of:

- $\mathcal{T}_2^{\mathrm{el}} = \langle \{1,2\}^*, Succ_1, Succ_2, \mathrm{el} \rangle \in \mathbb{A}$, and
- the MSO-theory of $\mathcal{T}_2^{\mathrm{el}}$ is undecidable (Thomas, 1990).

Outline

The class $\mathbb{R}$

Automatic trees with encoding of nodes by their very addresses.

Definition (Regular automatic trees)

Tree $\mathcal{T} = \langle D, r, Succ_1, \ldots, Succ_n, R_1, \ldots, R_p \rangle$ is regular automatic if

- Set of addresses $D \subseteq \{1, \ldots, n\}^*$ is a regular language;
- The identity encoding function provides an automatic presentation $\langle \mathcal{A}_D, \mathcal{A}_r, (\mathcal{A}_{Succ_i})_{1 \leq i \leq n}, (\mathcal{A}_{R_i})_{1 \leq i \leq p} \rangle$ of $\mathcal{T}$.

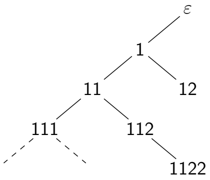
Intuition: substructure $\langle D, r, Succ_1, \ldots, Succ_n \rangle$ is the unfolding a finite structure, and relations $R_1, \ldots, R_p$ are regular.

Example

Binary infinite tree $\mathcal{T}_2$ + equal level is in $\mathbb{R}$.

$\mathbb{R}$ is a strict subset of automatic trees

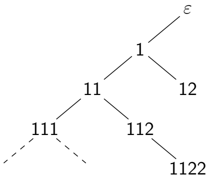
Consider tree



whose domain $\{1^i 2^j \mid 0 \leq j \leq i\}$ is not regular, so that $\notin \mathbb{R}$.

$\mathbb{R}$ is a strict subset of automatic trees

Consider tree

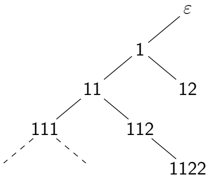


whose domain $\{1^i2^j \mid 0 \leq j \leq i\}$ is not regular, so that $\notin \mathbb{R}$.

And yet it has an automatic presentation, so $\in \mathbb{A}$:

$\mathbb{R}$ is a strict subset of automatic trees

Consider tree



whose domain $\{1^i 2^j \mid 0 \leq j \leq i\}$ is not regular, so that $\notin \mathbb{R}$.

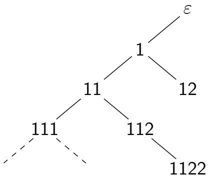
And yet it has an automatic presentation, so $\in \mathbb{A}$:

$$enc(1^i 2^j) := \mathtt{bin}(i) \otimes \mathtt{bin}(j)$$

where $\mathtt{bin}(n)$ is the binary string for n with least significant digit first:

$\mathbb{R}$ is a strict subset of automatic trees

Consider tree



whose domain $\{1^i 2^j \mid 0 \leq j \leq i\}$ is not regular, so that $\notin \mathbb{R}$.

And yet it has an automatic presentation, so $\in \mathbb{A}$:

$$enc(1^i 2^j) := \mathtt{bin}(i) \otimes \mathtt{bin}(j)$$

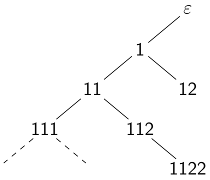
where $\mathtt{bin}(n)$ is the binary string for n with least significant digit first:

$enc(1) = \mathtt{bin}(1) \otimes \mathtt{bin}(0) = \binom{1}{0}$

$enc(112) = \mathtt{bin}(2) \otimes \mathtt{bin}(1) = \binom{0}{1}\binom{1}{\square}$

$\mathbb{R}$ is a strict subset of automatic trees

Consider tree



whose domain $\{1^i 2^j \mid 0 \leq j \leq i\}$ is not regular, so that $\notin \mathbb{R}$.

And yet it has an automatic presentation, so $\in \mathbb{A}$:

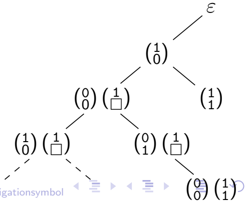
$$enc(1^i 2^j) := \mathtt{bin}(i) \otimes \mathtt{bin}(j)$$

where $\mathtt{bin}(n)$ is the binary string for n with least significant digit first:

$enc(1) = \mathtt{bin}(1) \otimes \mathtt{bin}(0) = \binom{1}{0}$

$enc(112) = \mathtt{bin}(2) \otimes \mathtt{bin}(1) = \binom{0}{1}\binom{1}{\square}$

One can verify that $enc(D)$, $enc(Succ_1)$, $enc(Succ_2)$ and $enc(\mathrm{el})$ are regular.



SYNTH($\mathbb{R}$,MSO)

For $\mathcal{T} = \langle D, r, Succ_1, \ldots, Succ_n, R_1, \ldots, R_p \rangle \in \mathbb{R}$, define:

- Generalized successor relation $Succ \stackrel{def}{=} \bigcup_{i=1}^{n} Succ_i$, and its reflexive and transitive closure $Succ^*$.
- Binary relations $\preccurlyeq$ for "deeper in the tree", el for "at equal level", and equality $=$.

Lemma

$\mathcal{T} \in \mathbb{R}$ *implies* $(\mathcal{T} + \{Succ^*, el, =\}) \in \mathbb{R}$.

:-) Allows for more a expressive FO logic, *e.g.* with some transitive closure.

$\textsc{Synth}(\mathbb{R},\text{MSO})$

For $\mathcal{T} = \langle D, r, Succ_1, \ldots, Succ_n, R_1, \ldots, R_p \rangle \in \mathbb{R}$, define:

- Generalized successor relation $Succ \stackrel{def}{=} \bigcup_{i=1}^{n} Succ_i$, and its reflexive and transitive closure $Succ^*$.
- Binary relations $\preccurlyeq$ for "deeper in the tree", el for "at equal level", and equality $=$.

Lemma

$\mathcal{T} \in \mathbb{R}$ implies $(\mathcal{T} + \{Succ^*, el, =\}) \in \mathbb{R}$.

:-) Allows for more a expressive FO logic, *e.g.* with some transitive closure. So, since $\mathcal{T}_2 \in \mathbb{R}$, we have $\mathcal{T}_2^{\mathrm{el}} \in \mathbb{R}$.

Corollary

$\textsc{Synth}(\mathbb{R},\text{MSO})$ *is not computable.*

$\textsc{Synth}(\mathbb{R},\mathrm{MSO})$

For $\mathcal{T} = \langle D, r, Succ_1, \ldots, Succ_n, R_1, \ldots, R_p \rangle \in \mathbb{R}$, define:

- Generalized successor relation $Succ \stackrel{def}{=} \bigcup_{i=1}^{n} Succ_i$, and its reflexive and transitive closure $Succ^*$.
- Binary relations $\preccurlyeq$ for "deeper in the tree", el for "at equal level", and equality $=$.

Lemma

$\mathcal{T} \in \mathbb{R}$ *implies* $(\mathcal{T} + \{Succ^*, el, =\}) \in \mathbb{R}$.

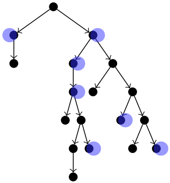
:-) Allows for more a expressive FO logic, *e.g.* with some transitive closure. So, since $\mathcal{T}_2 \in \mathbb{R}$, we have $\mathcal{T}_2^{\mathrm{el}} \in \mathbb{R}$.

Corollary

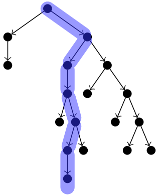
$\textsc{Synth}(\mathbb{R},\mathrm{MSO})$ *is not computable.*

However, we can get something by restricting MSO to $\textsc{chain}\mathrm{MSO}$.

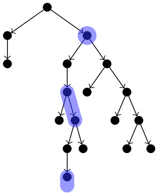
A variant of MSO over trees: CHAINMSO



(a) MSO
quantification over arbitrary subsets



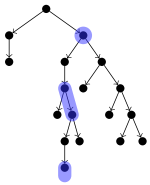
(b) PATHMSO
quantification over paths only



(c) CHAINMSO quantification over chains only

Chains in trees

$$\mathcal{T} = \langle D, r, Succ_1, \ldots, Succ_n, R_1, \ldots, R_p \rangle.$$



.

Definition (Chain)

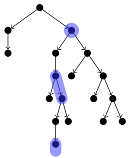
$\Gamma \subseteq D$ is a chain if it is totally ordered with respect to $Succ^*$:

$$\text{for every } u, v \in \Gamma, \text{ either } Succ^*(u,v) \text{ or } Succ^*(u,v).$$

As opposed to paths, there might be holes in a chain.

Logic $\textsc{chain}\mathrm{MSO}$

$\textsc{chain}\mathrm{MSO} \ni \Phi, \Psi ::= \mathsf{Sing}(X) \mid X \subseteq Y \mid R_i(X_1 \ldots X_{r_i}) \mid \neg\Phi \mid (\Phi \wedge \Psi) \mid \exists X \Phi$



$\mathcal{T}, \sigma \models \exists X \Phi$ iff there exists a chain Γ in $\mathcal{T}$ s.t. $\mathcal{T}, \sigma[X \mapsto \Gamma] \models \Phi$.

Example (Force chain X to be a maximal path starting at node x_0)

$x_0 \in X \wedge$
$\forall x \{x \in X \to [(\exists y Succ(x,y) \to \exists y(Succ(x,y) \wedge y \in X)) \wedge \neg Succ(x, x_0)]\}$

Corollary

*$\textsc{chain}\mathrm{MSO}$ subsumes $\textsc{path}\mathrm{MSO}$, CTL*K (Branching-time LTL), BL_μ^{lin} (Branching-time linear-time epistemic mu-calculus), etc.*

$\textsc{Synth}(\mathbb{R},\textsc{chain}\mathrm{MSO})$

Definition ($\textsc{Synth}(\mathbb{R},\textsc{chain}\mathrm{MSO})$)

$$\begin{cases} \textbf{In:} & \mathcal{T} = \langle D, r, Succ_1, \dots, Succ_n, R_1, \dots, R_p \rangle \in \mathbb{R}, \text{ and} \\ & \Phi(X_1, \dots, X_m) \in \textsc{chain}\mathrm{MSO} \\ \textbf{Out:} & \Phi^{\mathcal{S}} \stackrel{def}{=} \{(E_1, \dots, E_m) \in (2^D)^m \mid \mathcal{S}, [\vec{X} := \vec{E}] \models \Phi(X_1, \dots, X_m)\} \end{cases}$$

Theorem

$\textsc{Synth}(\mathbb{R},\textsc{chain}\mathrm{MSO})$ *is computable.*

The proof uses automata constructions, inspired from (Thomas, 1997).

$\textsc{Synth}(\mathbb{R},\textsc{chain}\mathrm{MSO})$ is computable: proof ingredients

1. Define *enc*(Γ) the encoding of chain Γ that "extends" the encoding of nodes.

$\textsc{Synth}(\mathbb{R}, \textsc{chain}MSO)$ is computable: proof ingredients

1. Define *enc*(Γ) the encoding of chain Γ that "extends" the encoding of nodes.

2. Design an automaton* $\mathcal{C}_m$ that recognizes (the encoding of) *m*-tuples of chains, *i.e.*

$$\bigcup_{\Gamma_1,\ldots,\Gamma_m \in Chains(\mathcal{T})} enc(\Gamma_1) \otimes \ldots \otimes enc(\Gamma_m).$$

$\textsc{Synth}(\mathbb{R}, \textsc{chain}MSO)$ is computable: proof ingredients

1. Define $enc(\Gamma)$ the encoding of chain Γ that "extends" the encoding of nodes.

2. Design an automaton* $\mathcal{C}_m$ that recognizes (the encoding of) m-tuples of chains, *i.e.*

$$\bigcup_{\Gamma_1,\ldots,\Gamma_m \in Chains(\mathcal{T})} enc(\Gamma_1) \otimes \ldots \otimes enc(\Gamma_m).$$

3. Design an automaton* $\mathcal{B}_\Phi$ that recognizes (the encoding of) $\Phi^{\mathcal{T}}$.

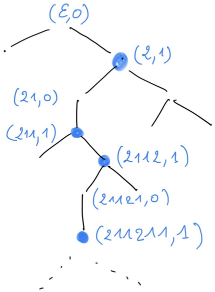
(*) Büchi automaton.

Encoding of chains - $enc(\Gamma)$

A set of addresses is a chain if, and only if, it is contained in the set of all prefixes of some infinite word.

- Given a chain $\Gamma \subseteq D$, define Branches(Γ) $:= \bigcap\{u\Sigma^\omega \mid u \in \Gamma\}$ the set of infinite words whose set of prefixes contain Γ.

  Branches(Γ) is a singleton $\{\alpha\}$ iff Γ is infinite.



Encoded as

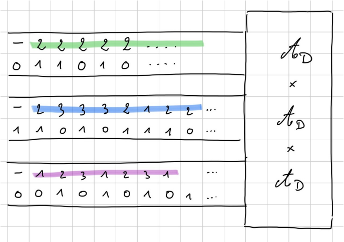
$$\binom{-}{0}\overbrace{\binom{2}{1}\binom{1}{0}\binom{1}{1}\binom{2}{1}\binom{1}{0}\binom{1}{1}\ldots}^{\alpha}$$

Büchi automaton for chain tuples - $\mathcal{C}_m$

Lemma

One can effectively construct a Büchi automaton $\mathcal{C}_m$ that recognizes the encoding of m-tuples of chains, i.e.

$$\bigcup_{\Gamma_1,\ldots,\Gamma_m \in Chains(\mathcal{T})} enc(\Gamma_1) \otimes \ldots \otimes enc(\Gamma_m)$$

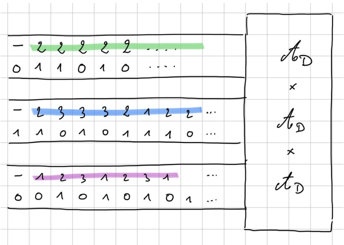


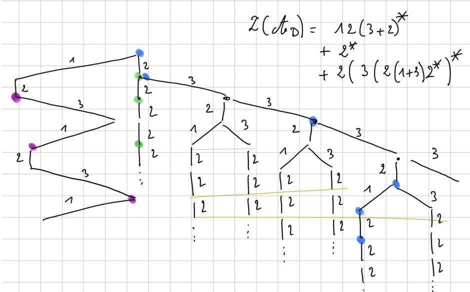
Büchi automaton for chain tuples - $\mathcal{C}_m$

Lemma

One can effectively construct a Büchi automaton $\mathcal{C}_m$ that recognizes the encoding of m-tuples of chains, i.e.

$$\bigcup_{\Gamma_1,\ldots,\Gamma_m \in Chains(\mathcal{T})} enc(\Gamma_1) \otimes \ldots \otimes enc(\Gamma_m)$$





Büchi automaton $\mathcal{B}_\Phi$ for $enc(\Phi^{\mathcal{T}})$

$\mathcal{T} = \langle D, r, Succ_1, \ldots, Succ_n, R_1, \ldots, R_p \rangle \in \mathbb{R}$ and $\Phi \in \textsc{chain}\mathrm{MSO}$

Define $\mathcal{B}_\Phi$ s.t. $\mathcal{L}(\mathcal{B}_\Phi) = enc(\Phi^{\mathcal{T}})$ where

$$enc(\Phi^{\mathcal{T}}) := \bigcup_{\substack{\Gamma_1, \ldots, \Gamma_m \in Chains(\mathcal{T}) \\ \mathcal{T}, [X_i \to \Gamma_i]_{1 \leq i \leq m} \models \Phi(X_1, \ldots, X_m)}} enc(\Gamma_1) \otimes \ldots \otimes enc(\Gamma_m)$$

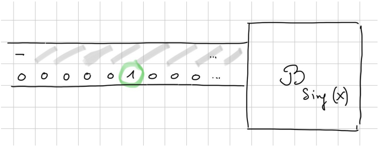
We define $\mathcal{B}_\Phi$ by induction over Φ...

Büchi automaton $\mathcal{B}_\Phi$ for $enc(\Phi^\mathcal{T})$

by induction over Φ

$$\text{CHAINMSO} \ni \Phi, \Psi ::= \mathsf{Sing}(X) \mid X \subseteq Y \mid R(X_1 \ldots X_r) \mid \neg\Phi \mid (\Phi \wedge \Psi) \mid \exists X \Phi$$

$\boxed{\text{The case of } \mathsf{Sing}(X)}$



Automaton $\mathcal{B}_{\mathsf{Sing}(X)}$ is the product of

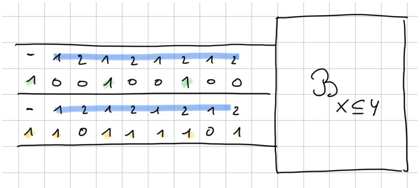
- automaton $\mathcal{C}$ that verifies that it is a chain encoding, and
- an automaton that verifies that the second component of $enc(\Gamma)$ has a single occurrence of symbol 1.

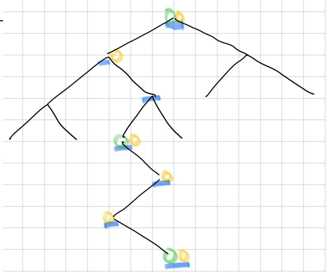
Büchi automaton $\mathcal{B}_\Phi$ for $enc(\Phi^\mathcal{T})$

by induction over Φ

$$\text{CHAINMSO} \ni \Phi, \Psi ::= \mathrm{Sing}(X) \mid X \subseteq Y \mid R(X_1 \ldots X_r) \mid \neg\Phi \mid (\Phi \wedge \Psi) \mid \exists X \Phi$$

The case of $X \subseteq Y$





Automaton $\mathcal{B}_{X \subseteq Y}$ is the product of

- automaton $\mathcal{C}$ to check that input (Γ_1, Γ_2) is a pair of chains
- an automaton that verifies that each time symbol 1 occurs in $enc(\Gamma_1)$ so does it in $enc(\Gamma_2)$.

Büchi automaton $\mathcal{B}_\Phi$ for $enc(\Phi^{\mathcal{T}})$

by induction over Φ

$$\textsc{chain}\mathrm{MSO} \ni \Phi, \Psi ::= \mathsf{Sing}(X) \,|\, X \subseteq Y \,|\, R(X_1 \dots X_r) \,|\, \neg\Phi \,|\, (\Phi \wedge \Psi) \,|\, \exists X \Phi$$

$$\boxed{\text{The case of } R(X_1 \dots X_r)}$$

Automaton $\mathcal{B}_{R(X_1 \dots X_r)}$ is the product of

- r copies of automaton $\mathcal{B}_{\mathsf{Sing}(X)}$
- a simulation of automaton $\mathcal{A}_R$
  - over $(\Sigma \times \{0,1\})^r$, instead of Σ^r
  - replaces by symbol $\square$ each letter after the unique symbol 1

Büchi automaton $\mathcal{B}_\Phi$ for $enc(\Phi^{\mathcal{T}})$

by induction over Φ

$$\text{CHAINMSO} \ni \Phi, \Psi ::= \mathsf{Sing}(X) \mid X \subseteq Y \mid R(X_1 \dots X_r) \mid \neg\Phi \mid (\Phi \wedge \Psi) \mid \exists X \Phi$$

$\boxed{\text{Propositional connectors}}$

- $\mathcal{B}_{\neg\Phi}$ is the complement of $\mathcal{B}_\Phi$ (see for instance (Vardi, 2007))
  - $+$ product with some $\mathcal{C}_m$ (in case Φ has m free variables)
- $\mathcal{B}_{\Phi\wedge\Psi}$ is the product of $\mathcal{B}_\Phi$ and $\mathcal{B}_\Psi$.

Büchi automaton $\mathcal{B}_\Phi$ for $enc(\Phi^\mathcal{T})$

by induction over Φ

$$\text{CHAINMSO} \ni \Phi, \Psi ::= \mathsf{Sing}(X) \,|\, X \subseteq Y \,|\, R(X_1 \ldots X_r) \,|\, \neg\Phi \,|\, (\Phi \wedge \Psi) \,|\, \exists X \Phi$$

$$\boxed{\text{The case of } \exists X \Phi}$$

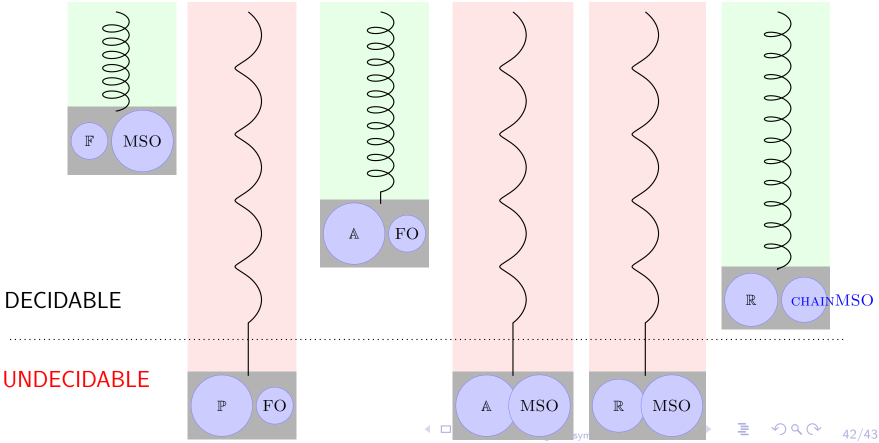
$\mathcal{B}_{\exists X_1 \Phi(X_1, X_2, \ldots, X_m)}$ is the projection of automaton $\mathcal{B}_{\Phi(X_1, X_2, \ldots, X_m)}$.

(case $m = 1$) $\mathcal{B}_{\exists X \Phi(X)}$ is input-free.

Synthesis in infinite structures

$\mathbb{P}$ (PCP structures)

$\mathbb{A}$ (Automatic structures), $\mathbb{R}$ (Regular automatic trees)



Concluding remarks

- Synthesis complexity is high, *e.g.* $\mathcal{T} \models \mathrm{FO}$ non-elementary in altermation depth φ.

Concluding remarks

- Synthesis complexity is high, *e.g.* $\mathcal{T} \models \mathrm{FO}$ non-elementary in altermation depth φ.
- Other kinds of finite presentations for input structures?

Concluding remarks

- Synthesis complexity is high, *e.g.* $\mathcal{T} \models \mathrm{FO}$ non-elementary in altermation depth φ.
- Other kinds of finite presentations for input structures?
  - Dynamic Epistemic Logic (DEL) (Van Ditmarsch et al., 2007) presentations: a finite description $(\mathcal{M}, \mathcal{E})$ denotes the whole relational structure with iterated update product.

Concluding remarks

- Synthesis complexity is high, *e.g.* $\mathcal{T} \models \mathrm{FO}$ non-elementary in altermation depth φ.
- Other kinds of finite presentations for input structures?
  - Dynamic Epistemic Logic (DEL) (Van Ditmarsch et al., 2007) presentations: a finite description $(\mathcal{M}, \mathcal{E})$ denotes the whole relational structure with iterated update product.
    Synthesis (of plans) cannot be computed in general, see (Bolander et al., 2020) for a survey.
    - Computable in the class of propositional DEL structures see (Douéneau-Tabot et al., 2018) for $\textsc{chain}\mathrm{MSO}$ goals
    - Computable in some fragment of first-order DEL see Côme Neyrand's talk at this workshop

Concluding remarks

- Synthesis complexity is high, *e.g.* $\mathcal{T} \models \mathrm{FO}$ non-elementary in altermation depth φ.
- Other kinds of finite presentations for input structures?
    - Dynamic Epistemic Logic (DEL) (Van Ditmarsch et al., 2007) presentations: a finite description $(\mathcal{M}, \mathcal{E})$ denotes the whole relational structure with iterated update product.
      Synthesis (of plans) cannot be computed in general, see (Bolander et al., 2020) for a survey.
        - Computable in the class of propositional DEL structures see (Douéneau-Tabot et al., 2018) for $\textsc{chain}\mathrm{MSO}$ goals
        - Computable in some fragment of first-order DEL see Côme Neyrand's talk at this workshop
    - Caucal hierarchy (Caucal, 2002) presentations: apply a finite sequence of unfolding operation then some inverse rational mappings to tree $\mathcal{T}_2$.
      All have a decidable MSO theory, synthesis should work.

Blumensath, A. and Grädel, E. (2000).
Automatic structures.
In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science*, LICS '00, page 51, USA. IEEE Computer Society.

Bolander, T., Charrier, T., Pinchinat, S., and Schwarzentruber, F. (2020).
Del-based epistemic planning: Decidability and complexity.
Artificial Intelligence, 287:103304.

Caucal, D. (2002).
On infinite terms having a decidable monadic theory.
In *International Symposium on Mathematical Foundations of Computer Science*, pages 165–176. Springer.

Douéneau-Tabot, G., Pinchinat, S., and Schwarzentruber, F. (2018).
Chain-monadic second order logic over regular automatic trees and epistemic planning synthesis.
In *AiML'18*.

Khoussainov, B., Nies, A., Rubin, S., and Stephan, F. (2007).
Automatic structures: Richness and limitations.
Log. Methods Comput. Sci., 3(2).

Post, E. L. (1946).
A variant of a recursively unsolvable problem.
Bulletin of the American Mathematical Society, 52(4):264–268.

Thomas, W. (1990).
Automata on infinite objects.
Hand. of theoretical computer science, Volume B, pages 133–191.

Thomas, W. (1997).
Languages, automata, and logic.
In *Handbook of formal languages*, pages 389–455. Springer.

Van Ditmarsch, H., van Der Hoek, W., and Kooi, B. (2007).
Dynamic epistemic logic, volume 337.
Springer Science & Business Media.

Vardi, M. Y. (2007).
The büchi complementation saga.
In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 12–22. Springer.