GandALF: synthesis

Jan Otop

May 16, 2018

Plan for today

- Reactive synthesis
- Distributed synthesis
- Synthesis from components
- Next week: Other approaches to synthesis (CEGIS, SyGus, ...)

From model checking to synthesis

- A system modeled by an automaton M.
- Correct behaviors specified by an LTL formula φ.
- The system is correct $\iff$ $\mathrm{M} \models \varphi$.

From model checking to synthesis

- A system modeled by an automaton M.
- Correct behaviors specified by an LTL formula φ.
- The system is correct $\iff$ $M \models \varphi$.

Problem: incorrect systems need to be redesigned.

Solution: automatically synthesize M to satisfy φ.

Synthesis — "extreme form of declarative programming".

Reactive systems

Reactive system — system with input and output.

Transducers (Mealy machines)

A transducer M is a tuple $\langle \Sigma_I, \Sigma_O, Q, q_0, \delta, F \rangle$ such that $\delta : \Sigma_I \times Q \to \Sigma_O \times Q$.

- A transducer M represents a function $\mathcal{L}(M) : \Sigma_I^\omega \to \Sigma_O^\omega$.
- Graph of a transducer: language over $\Sigma_I \times \Sigma_O$.
- Büchi automata recognize graphs of transducers:

  Transducers and automata: $\langle a, q \rangle \to \langle b, q' \rangle$ and $\langle (a, b), q \rangle \to q'$.

The language of a transducer M represented by a Σ_O-labeled Σ_I-tree.

LTL specifications

LTL specifications

An LTL formula is a propositions formula with operators:
$\mathrm{G}\varphi, \mathrm{F}\varphi, \mathrm{X}\varphi, \varphi_1 \mathrm{U} \varphi_2$.

Semantics of LTL.

Theorem

Given an LTL formula φ over variables $\mathrm{P} = \{\mathrm{p}_1, \ldots, \mathrm{p}_\mathrm{k}\}$, we can construct an exponential-size Büchi automaton $\mathcal{M}$ over $\Sigma = 2^\mathrm{P}$ such that $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\varphi) = \{\mathrm{w} \mid \mathrm{w} \models \varphi\}$.

Example: Arbiter

- Two components C_1, C_2 sending requests r_1, r_2 to access a resource.
- Arbiter sends grants g_1, g_2.
- Mutual exclusion: $G(\neg g_1 \vee \neg g_2)$.
- For $i = 1, 2$ fairness for C_i: $G(r_i \rightarrow Fg_i)$.

Synthesis of reactive systems

Realizability

Given an LTL spec. φ decide whether there exists a transducer M such that $\mathcal{L}(\mathrm{M}) \subseteq \mathcal{L}(\varphi)$.

The synthesis problem: find such M.

Problems: Domain (partial function) and clairvoyance.

We have all the ingredients to solve it!

Solving reactive synthesis

1. Construct a NBW $\mathcal{M}_{\mathrm{B}}$ for φ.
2. Transform $\mathcal{M}_{\mathrm{B}}$ to an equivalent DPW $\mathcal{M}_{\mathrm{P}}$.
3. Construct a DPT $\mathcal{M}_{\mathrm{T}}$ based on $\mathcal{M}_{\mathrm{P}}$.
4. Check emptiness of $\mathcal{M}_{\mathrm{T}}$.
5. If the language of $\mathcal{M}_{\mathrm{T}}$ is non-empty, take a regular tree and transform it into a transducer M.

Solving reactive synthesis

1. Construct a NBW $\mathcal{M}_{\mathrm{B}}$ for φ.
2. Transform $\mathcal{M}_{\mathrm{B}}$ to an equivalent DPW $\mathcal{M}_{\mathrm{P}}$.
3. Construct a DPT $\mathcal{M}_{\mathrm{T}}$ based on $\mathcal{M}_{\mathrm{P}}$.
4. Check emptiness of $\mathcal{M}_{\mathrm{T}}$.
5. If the language of $\mathcal{M}_{\mathrm{T}}$ is non-empty, take a regular tree and transform it into a transducer M.

Theorem

The realizability is 2-EXPTIME-complete.

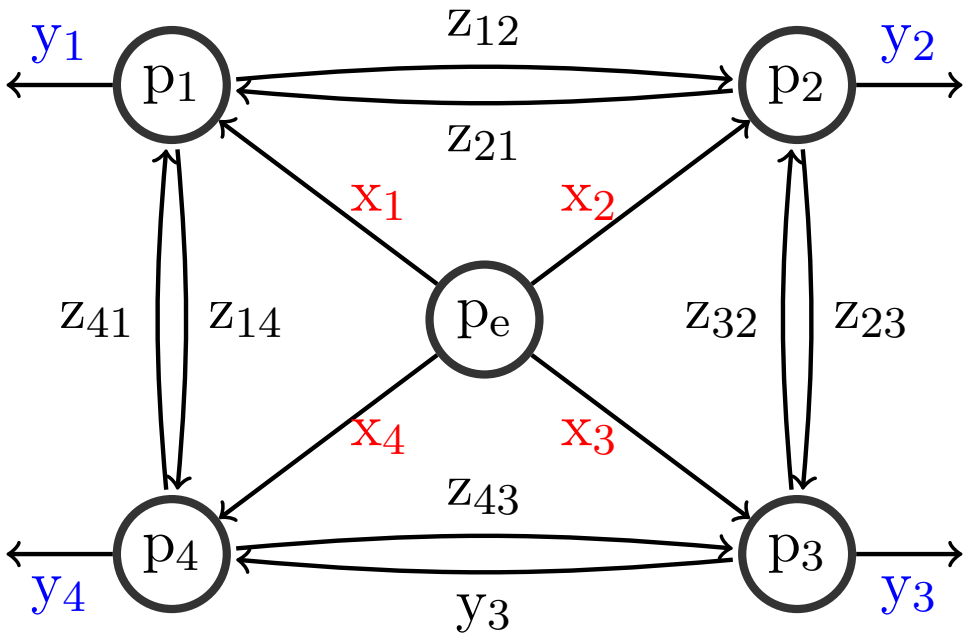
Parametric approach

1. Construct a NBW $\mathcal{M}_{\mathrm{B}}$ for $\neg\varphi$.
2. Complement $\mathcal{M}_{\mathrm{B}}$ into an UCW $\mathcal{M}_{\mathrm{U}}$.
3. Construct an UCT $\mathcal{M}_{\mathrm{T}}$ based on $\mathcal{M}_{\mathrm{U}}$.
4. Consider UCT $\mathcal{M}_{\mathrm{T}}$ as k-UCT $\mathcal{M}_{\mathrm{T,k}}$.
5. Check emptiness of $\mathcal{M}_{\mathrm{T,k}}$.
6. If the language of $\mathcal{M}_{\mathrm{T,k}}$ is non-empty, take a regular tree and transform it into a transducer M.

Reactive Distributed Systems

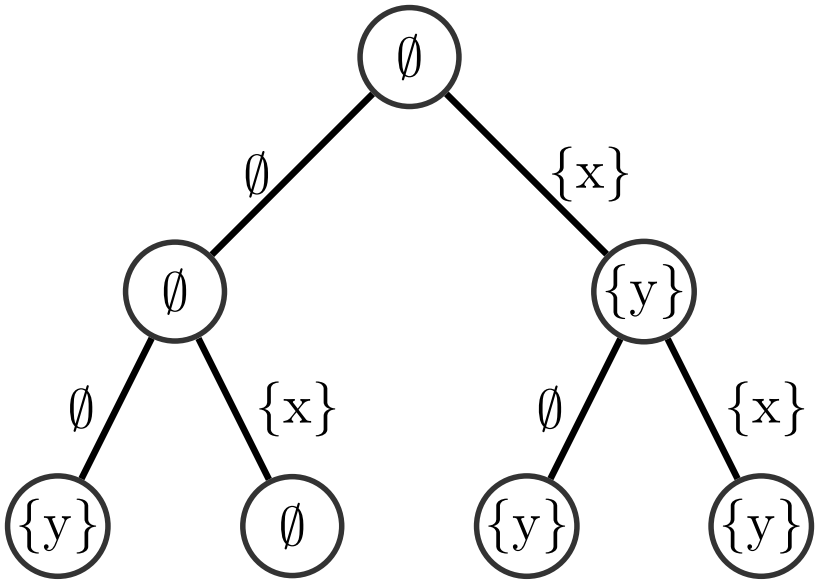
Synchronous architecture $\mathcal{A} = (\mathcal{P}, \mathrm{p_e}, \mathrm{V}, \mathrm{E})$

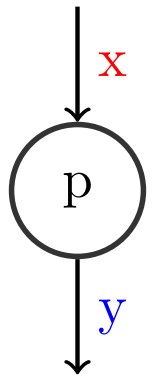
- $\mathcal{P}$ is a set of $\mathrm{n}+1$ processes.
- $\mathrm{p_e} \in \mathcal{P}$ is the environment.
- V is a set of binary variables.
- $\mathrm{E} : \mathcal{P} \times \mathcal{P} \to 2^{\mathrm{V}}$ defines the communication.
- For $\mathrm{p} \in \mathcal{P}$ denote input variables with $\mathrm{I(p)}$, output variables with $\mathrm{O(p)}$.



Strategies

- Process p behaves according to local strategy $\sigma_{\mathrm{p}} : \left(2^{\mathrm{I(p)}}\right)^* \to 2^{\mathrm{O(p)}}$.
- Can be viewed as the labeling of an infinite $2^{\mathrm{I(p)}}$-tree, $\mathrm{T}_{\sigma_{\mathrm{p}}}$.

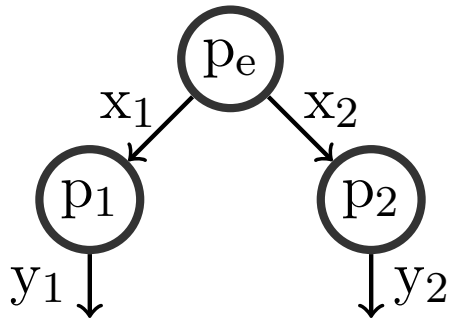




- The collective strategy $\sigma : \left(2^{\mathrm{O(p_e)}}\right)^* \to 2^{\mathrm{V}\setminus\mathrm{O(p_e)}}$ determines the distributed behavior of the system.
- Can be viewed as the labeling of an infinite $2^{\mathrm{O(p_e)}}$-tree, T_{σ}.

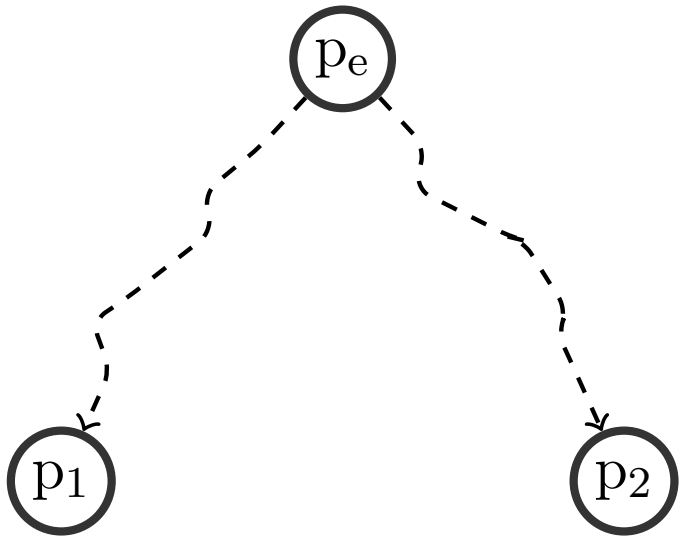
Distributed Realizability is Undecidable

- Distributed realizability was shown to be undecidable for the following architecture.
- Reduction from the halting problem.
- For any Turing machine M, construct ϕ_M which requires that p_1, p_2 output a legal sequence of configurations of M, and M halts.
  1. When p_i receives a start signal, it outputs a sequence of legal configurations of M.
  2. Initially p_i outputs the first two configurations of M.
  3. If p_1, p_2 output C_1C_1' and C_2C_2' and $C_1 \vdash C_2$, then $C_1' \vdash C_2'$.



Parametric on the Architecture

- For which classes of architectures is realizability decidable?
- Complete characterization base on the information fork criterion.
- Processes p_1, p_2 form an information fork in architecture $\mathcal{A}$ if there exist paths $p_e \rightsquigarrow p_i$ in $\mathcal{A}$ such that do not traverse edges in $I(p_{-i})$.



- Every architecture either:
  - Has an information fork (undecidable).
  - Can be reduced to a pipeline (decidable).

Synthesis from components

A component C is a transducer.
Given C_1, C_2 such that $\Sigma^1_O = \Sigma^2_I$, we construct a component $C_1 \circ C_2$.

Realizability from components

Given a set $\{C_i \mid i = 1, \ldots, k\}$ and an LTL spec. φ decide whether there exists $x[1], \ldots, x[n]$ such that $C_{x[1]} \circ \ldots \circ C_{x[n]}$ is defined and satisfies φ.

Synthesis from components

A component C is a transducer.
Given C_1, C_2 such that $\Sigma^1_O = \Sigma^2_I$, we construct a component $C_1 \circ C_2$.

Realizability from components

Given a set $\{C_i \mid i = 1, \ldots, k\}$ and an LTL spec. φ decide whether there exists $x[1], \ldots, x[n]$ such that $C_{x[1]} \circ \ldots \circ C_{x[n]}$ is defined and satisfies φ.

Theorem

Realizability from components is undecidable.

Acknowledgements

Slides on distributed synthesis have been prepared by Andreas Pavlogiannis (http://lara.epfl.ch/~pavlogia/).