Games and synthesis: going distributed

Anca Muscholl

GAMES Winter School Champèry, February 2013

Motivation

- Verification of distributed programs is challenging. Adding "dimensions" like recursion, communication, etc easily leads to undecidability. Many approximated techniques.
- Synthesis of distributed programs is even more challenging: we don't know how to proceed in a systematic way. Techniques are sparse and ad-hoc.
- Why do we need to synthesize programs? Lack of expertise, laziness, experimenting, ... Very popular to "learn" from examples.

Motivation

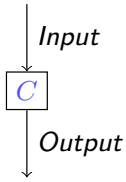
- Verification of distributed programs is challenging. Adding "dimensions" like recursion, communication, etc easily leads to undecidability. Many approximated techniques.
- Synthesis of distributed programs is even more challenging: we don't know how to proceed in a systematic way. Techniques are sparse and ad-hoc.
- Why do we need to synthesize programs? Lack of expertise, laziness, experimenting, ... Very popular to "learn" from examples.

Outline

1. Basics on synthesis and control: Church's problem & Ramadge/Wonham supervisory control
2. Distributed synthesis: Pnueli/Rosner model
3. Control for distributed automata. New decidability results.

I. Basics on synthesis and control

Church's problem (1963) "Logic, arithmetic and automata"

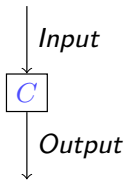


Problem

- *Given*: specification $S \subseteq (\{0,1\} \times \{0,1\})^\omega$ relating inputs/outputs.
- *Output*: I/O device $C : \{0,1\}^* \to \{0,1\}$ s.t. $(x, C(x)) \in S$ for all inputs x.

Device C must react correctly on every input.

Church's problem (1963) "Logic, arithmetic and automata"



Problem

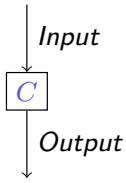
- *Given*: specification $S \subseteq (\{0,1\} \times \{0,1\})^\omega$ relating inputs/outputs.
- *Output*: I/O device $C : \{0,1\}^* \to \{0,1\}$ s.t. $(x, C(x)) \in S$ for all inputs x.

Device C must react correctly on every input.

Remarks

- The specification S is provided in an effective way, by an MSO formula or a Büchi automaton.
- The problem is more complicated than just requiring $\forall x \, \exists y \, .(x, y) \in S$: device C must react continuously on inputs.

Church's problem (1963) "Logic, arithmetic and automata"



Problem

- *Given*: specification $S \subseteq (\{0,1\} \times \{0,1\})^\omega$ relating inputs/outputs.
- *Output*: I/O device $C : \{0,1\}^* \to \{0,1\}$ s.t. $(x, C(x)) \in S$ for all inputs x.

Device C must react correctly on every input.

Remarks

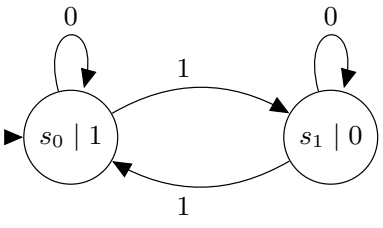
- The specification S is provided in an effective way, by an MSO formula or a Büchi automaton.
- The problem is more complicated than just requiring $\forall x \, \exists y \, .(x, y) \in S$: device C must react continuously on inputs.

Church's problem:
Synthesis of open systems: systems reacting on input from environment.

Examples

Ex. 1

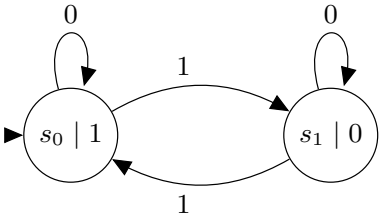
S: "*the output is 1 iff the number of previous inputs equal to 1, is even*"



Examples

Ex. 1

S: "*the output is 1 iff the number of previous inputs equal to 1, is even*"



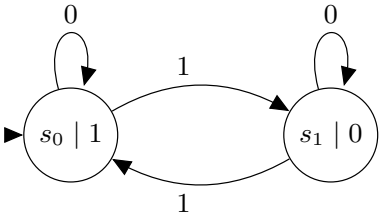
Ex. 2

S: "*the output is 1 iff some future input is 1*"

Examples

Ex. 1

S: "*the output is 1 iff the number of previous inputs equal to 1, is even*"



Ex. 2

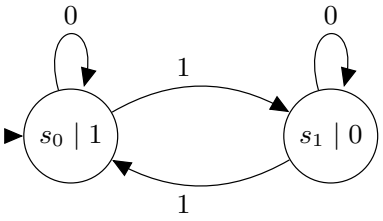
S: "*the output is 1 iff some future input is 1*"

No solution.

Examples

Ex. 1

S: "*the output is 1 iff the number of previous inputs equal to 1, is even*"



Ex. 2

S: "*the output is 1 iff some future input is 1*"

No solution.

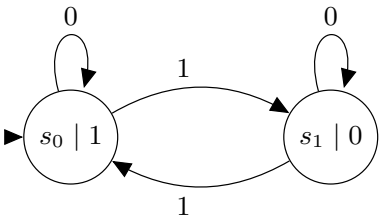
Ex. 3

S: "*The output has infinitely many 1's if the input has infinitely many 1's*".

Examples

Ex. 1

S: "*the output is 1 iff the number of previous inputs equal to 1, is even*"



Ex. 2

S: "*the output is 1 iff some future input is 1*"

No solution.

Ex. 3

S: "*The output has infinitely many 1's if the input has infinitely many 1's*".

Various solutions (e.g. copying the input, outputting always 1,...).

Examples (cont.)

Ex. 4

S: "*The output has finitely many 1's iff the input has infinitely many 1's*".

Examples (cont.)

Ex. 4

S: "*The output has finitely many 1's iff the input has infinitely many 1's*".

- on 0^ω the output should contain at least one 1, say after k_1 steps;
- on $0^{k_1}10^\omega$ the output should contain at least one more 1, say after another k_2 steps;
- In the limit: on $0^{k_1}10^{k_2}1\cdots$ the output will contain infinitely many 1's.

Examples (cont.)

Ex. 4

S: "*The output has finitely many 1's iff the input has infinitely many 1's*".

- on 0^ω the output should contain at least one 1, say after k_1 steps;
- on $0^{k_1}10^\omega$ the output should contain at least one more 1, say after another k_2 steps;
- In the limit: on $0^{k_1}10^{k_2}1\cdots$ the output will contain infinitely many 1's.

No solution.

Church's problem and logic

Specifications

Specification $S \subseteq (\{0,1\} \times \{0,1\})^\omega$: finite description, say Büchi automaton.

Church's problem and logic

Specifications

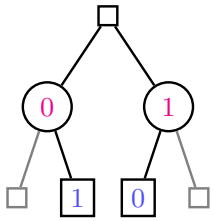
Specification $S \subseteq (\{0,1\} \times \{0,1\})^\omega$: finite description, say Büchi automaton.

Trees

Synthesis is concerned with trees:

$$t : \{0,1\}^\omega \to \Sigma$$

$C : \{0,1\}^* \to \{0,1\}$ is a subset of the tree.



Church's problem and logic

Specifications

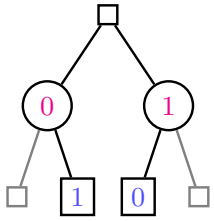
Specification $S \subseteq (\{0,1\} \times \{0,1\})^\omega$: finite description, say Büchi automaton.

Trees

Synthesis is concerned with trees:

$$t : \{0,1\}^\omega \to \Sigma$$

$C : \{0,1\}^* \to \{0,1\}$ is a subset of the tree.



Monadic second-order logic (MSO)

- quantification over nodes and sets of nodes in the tree
- atomic predicates: labels on nodes, descendant order

Church's problem and logic

Specifications

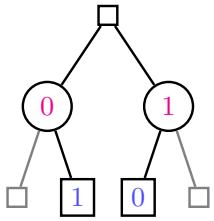
Specification $S \subseteq (\{0,1\} \times \{0,1\})^\omega$: finite description, say Büchi automaton.

Trees

Synthesis is concerned with trees:

$$t : \{0,1\}^\omega \to \Sigma$$

$C : \{0,1\}^* \to \{0,1\}$ is a subset of the tree.



Monadic second-order logic (MSO)

- quantification over nodes and sets of nodes in the tree
- atomic predicates: labels on nodes, descendant order

The existence of a device C satisfying S can be expressed by an MSO formula. Apply then Rabin's theorem (1972) on the decidability of MSO. If a solution C exists, then it is a finite automaton.

Church's problem and games

2-player games

- Game arena: graph with vertex set V, edge set E.
- Two players P_0 (system) et P_1 (environment). The set of vertices is partitioned into two disjoint subsets: V_0 belongs to P_0 and V_1 to P_1.
- Play = path is the graph. Owner of the current vertex chooses an outgoing edge.
- Winning condition on plays (e.g. reachability, Büchi, parity,...).
- Strategies: $\sigma_0 : V^* V_0 \to V$, $\sigma_1 : V^* V_1 \to V$.

Church's problem rephrased as a game: McNaughton, Büchi-Landweber 1969.

Church's problem and games

2-player games

- Game arena: graph with vertex set V, edge set E.
- Two players P_0 (system) et P_1 (environment). The set of vertices is partitioned into two disjoint subsets: V_0 belongs to P_0 and V_1 to P_1.
- Play = path is the graph. Owner of the current vertex chooses an outgoing edge.
- Winning condition on plays (e.g. reachability, Büchi, parity,...).
- Strategies: $\sigma_0 : V^* V_0 \to V$, $\sigma_1 : V^* V_1 \to V$.

Church's problem rephrased as a game: McNaughton, Büchi-Landweber 1969.

Complexity

- For regular specifications one needs to solve parity games.
- Linear temporal logic specifications: 2-EXPTIME (Pnueli/Rosner).
- Branching-time specifications: EXPTIME for CTL, 2-EXPTIME for CTL* (Kupferman/Vardi).

Supervisory control: Ramadge/Wonham

Setting

Partition the set Σ of actions of a given "plant" P into controllable actions from Σ^{sys} and uncontrollable actions from Σ^{env}.

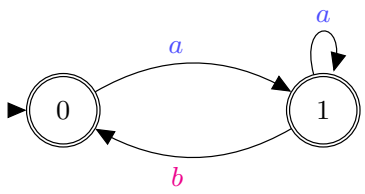
Supervisory control: Ramadge/Wonham

Setting

Partition the set Σ of actions of a given "plant" P into controllable actions from Σ^{sys} and uncontrollable actions from Σ^{env}.

Example

Plant P with $\Sigma^{env} = \{b\}$:



S: at most 2 consecutive a's

- Controller: mapping $C : \text{Path}(P) \to 2^{\Sigma}$ s.t. $\Sigma^{env} \subseteq C(w)$ for all w.
- Controlled plant $P \times C$ must satisfy S.
- Here: C counts a up to 2 and $P \times C = ((a + aa)b)^*(\epsilon + a + aa)$

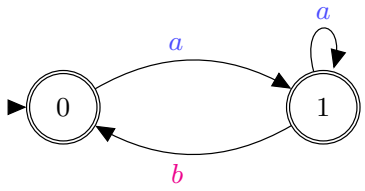
Supervisory control: Ramadge/Wonham

Setting

Partition the set Σ of actions of a given "plant" P into controllable actions from Σ^{sys} and uncontrollable actions from Σ^{env}.

Example

Plant P with $\Sigma^{env} = \{b\}$:



S: at most 2 consecutive a's

- Controller: mapping $C : \text{Path}(P) \to 2^{\Sigma}$ s.t. $\Sigma^{env} \subseteq C(w)$ for all w.
- Controlled plant $P \times C$ must satisfy S.
- Here: C counts a up to 2 and $P \times C = ((a + aa)b)^*(\epsilon + a + aa)$

Rem.

Control subsumes synthesis. The control problem statement is more flexible w.r.t. auxiliary constraints on the solution (e.g., the controlled plant should be non-blocking).

Ramadge and Wonham

Problem

Given:

- A *plant* P (DFA, all states final) over alphabet partitioned into controllable actions Σ^{sys} and uncontrollable actions Σ^{env}.
- A specification S (DFA, all states final).

Ramadge and Wonham

Problem

Given:

- A *plant* P (DFA, all states final) over alphabet partitioned into controllable actions Σ^{sys} and uncontrollable actions Σ^{env}.
- A specification S (DFA, all states final).

Compute C such that:

- $P \times C \subseteq S$, and
- $w \in C$ and $a \in \Sigma^{env}$ implies $wa \in C$.

Ramadge and Wonham

Problem

Given:

- A *plant* P (DFA, all states final) over alphabet partitioned into controllable actions Σ^{sys} and uncontrollable actions Σ^{env}.
- A specification S (DFA, all states final).

Compute C such that:

- $P \times C \subseteq S$, and
- $w \in C$ and $a \in \Sigma^{env}$ implies $wa \in C$.

Solution

Build $P \times S$ and remove all states (p, s) such that for some $w \in (\Sigma^{env})^*$: $p \cdot w$ is defined and $s \cdot w$ is not. The output is the largest solution.

Ramadge and Wonham

Problem

Given:

- A *plant* P (DFA, all states final) over alphabet partitioned into controllable actions Σ^{sys} and uncontrollable actions Σ^{env}.
- A specification S (DFA, all states final).

Compute C such that:

- $P \times C \subseteq S$, and
- $w \in C$ and $a \in \Sigma^{env}$ implies $wa \in C$.

Solution

Build $P \times S$ and remove all states (p, s) such that for some $w \in (\Sigma^{env})^*$: $p \cdot w$ is defined and $s \cdot w$ is not. The output is the largest solution.

Further requirements

- Non-blocking controller: if $w \in P \times C$ has some extension in P, then also one in $P \times C$.
- Final states in P: every $w \in P \times C$ is prefix of some $w' \in P \times C$ ending in a final state for P.
- Unobservable actions for the controller.