Open / Reactive system

Synthesis problem
- Given a specification \( \varphi \), decide whether there exists a program \( P \) such that \( P \parallel E \models \varphi \) for all environment \( E \).
- Build such a program \( P \) (if one exists).

Specification

Example: Elevator
- Inputs: call for level \( i \).
- Outputs: open/close door \( i \), move 1 level up/down.

Linear time: LTL, FO, MSO, regular, ...
- Safety: \( G(\text{level} \neq i \rightarrow \text{is\_closed}) \)
- Liveness: \( G(\text{is\_called}, \rightarrow F(\text{level} = i \land \text{is\_open})) \)

Branching time: CTL, CTL\( ^* \), \( \mu \)-calculus, ...
- \( AG(c\text{all}_i)T \) (\( c\text{all}_i \) is uncontrollable)
- \( AGF(\text{level} = 0 \land \text{is\_open}_0) \)
Synthesis of reactive programs

Reactive program

\[ x \xrightarrow{f} y \]

- \(Q_x\): domain for input variable \(x\)
- \(Q_y\): domain for output variable \(y\)
- Program: \(f : Q_x^+ \to Q_y\)
- Input: \(x_1 x_2 \cdots \in Q_x^+\)
- Behavior: \((x_1, y_1)(x_2, y_2)(x_3, y_3) \cdots \text{ with } y_n = f(x_1 \cdots x_n)\) for all \(n > 0\).

Chruch problem (implementability) 1962

- Given a linear time specification \(\varphi\) over the alphabet \(\Sigma = Q_x \times Q_y\).
  - Does there exist a program \(f\) such that all \(f\)-behaviors satisfy \(\varphi\)?

Implementability \(\neq\) Satisfiability

- \(Q_x = \{0, 1\}\) and \(\varphi := F(x = 1)\)
- \(\varphi\) is satisfiable: \((1, 0)^\omega \models \varphi\)
- \(\varphi\) is not implementable since the input is not controllable.

Implementability \(\neq\) Validity of \(\forall \vec{x} \exists \vec{y} \varphi\)

- \(Q_x = Q_y = \{0, 1\}\) and \(\varphi := (y = 1) \longrightarrow F(x = 1)\)
- \(\forall \vec{x} \exists \vec{y} \varphi\) is valid.
- \(\varphi\) is not implementable by a reactive program.

For non-reactive terminating programs, Implementability = Validity of \(\forall \vec{x} \exists \vec{y} \varphi\)

Synthesis of reactive programs

Chruch problem (implementability) 1962

- Given a linear time specification \(\varphi\) over the alphabet \(\Sigma = Q_x \times Q_y\).
  - Does there exist a program \(f\) such that all \(f\)-behaviors satisfy \(\varphi\)?

Theorem (Pnueli-Rosner 89)

- The specification \(\varphi \in \text{LTL}\) is implementable iff the formula
  \[
  A \varphi \land AG( \bigwedge_{a \in Q_x} EX(x = a) )
  \]
  is satisfiable.

When \(\varphi\) is implementable, we can construct a finite state implementation (program) in time doubly exponential in \(\varphi\).

Control problem

Open system: Transitions system \(\mathcal{A} = (Q, \Sigma, q_0, \delta)\)

- \(Q\): finite or infinite set of states,
- \(\delta\): deterministic or non-deterministic transition function.

Control problem

- Given a system \(S\) and a specification \(\varphi\), decide whether there exists a controller \(C\) such that \((S \otimes C)[E] \models \varphi\).
- Build such a controller \(C\) (if one exists).
Control versus Game

Correspondance

<table>
<thead>
<tr>
<th>Transition system</th>
<th>= Game arena (graph).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controllable events</td>
<td>= Actions of player 1 (controller).</td>
</tr>
<tr>
<td>Uncontrollable events</td>
<td>= Action of player 0 (opponent, environment).</td>
</tr>
<tr>
<td>Behavior</td>
<td>= Play.</td>
</tr>
<tr>
<td>Controller</td>
<td>= Strategy.</td>
</tr>
<tr>
<td>Specification</td>
<td>= Winning condition.</td>
</tr>
<tr>
<td>Finding a controller</td>
<td>= finding a winning strategy.</td>
</tr>
</tbody>
</table>

Theorem: Büchi - Landweber 1969
If the system is finite state and the specification is regular then the control problem is decidable.
Moreover, when $(S, \varphi)$ is controllable, we can synthesize a finite state controller.

Outline

Synthesis and control for sequential systems

- Synthesis and control for distributed systems

Well-connected architectures

Program synthesis versus System control

Equivalence

The implementability problem for

\[ x \rightarrow \ Q_a \rightarrow \ Q_y \]

is equivalent to the control problem for the system

\[ Q_x \rightarrow \ Q_y \]

Distributed synthesis

Distributed synthesis problem

- Decide whether there exists a distributed program st.
- \[ P_1 \parallel \cdots \parallel P_n \parallel \ E \models \varphi. \]
- Synthesis: If so, compute such a distributed program.

Peterson-Reif 1979, Pnueli-Rosner 1990

In general, the problem is undecidable.
Distributed control

Main parameters
- Which subclass of architectures?
- Which semantics? synchronous (with or without delay), asynchronous
- What kind of specification? LTL, CLT*, μ-calculus Rational, Recognizable word/tree
- What kind of memory for the programs? memoryless, local memory, causal memory finite or infinite memory

Distributed Synthesis or control

Architecture with shared variables

Example

Programming with local memory: \( f_x : Q_u^* \rightarrow Q_x \) and \( f_z : (Q_x \times Q_o)^* \rightarrow Q_z \).
- Input: \( \begin{pmatrix} u_1 & u_2 & u_3 & \cdots \\ v_1 & v_2 & v_3 & \cdots \end{pmatrix} \in (Q_u \times Q_v)\omega \).
- Behavior: \( \begin{pmatrix} u_1 & u_2 & u_3 & \cdots \\ v_1 & v_2 & v_3 & \cdots \\ x_1 & x_2 & x_3 & \cdots \\ z_1 & z_2 & z_3 & \cdots \end{pmatrix} \)
with \( x_n = f_x(u_1 \cdots u_n) \) \( z_n = f_z((x_1, v_1) \cdots (x_n, v_n)) \) for all \( n > 0 \).
**Undecidability proof 1**

**SPEC₁**: processes $a$ and $b$ must output configurations

$$\begin{align*}
\boxdot & \quad \boxslash \quad 0^q 1^q 0 \cdots \quad : n(v) = p \\
\downarrow & \quad \downarrow \\
\phi & \quad \psi \\
\tau & \quad \eta \\
\#^q \mu C \# & : \text{where } C \in \Gamma^+ Q \Gamma^+
\end{align*}$$

$$(v = 0 \land y = \#) \mathsf{W} (v = 1 \land (v = 1 \land y = \#) \mathsf{W} (v = 0 \land y \in \Gamma^+ Q \Gamma^+ \#^\omega))$$

where

$$y \in \Gamma^+ Q \Gamma^+ \#^\omega \quad \text{def} \quad y \in \Gamma \cup \{ y \in Q \land X (y \in \Gamma \cup (y \in \Gamma \land G y = \#)) \}$$

**Undecidability proof 2**

**SPEC₂**: processes $a$ and $b$ must start with the first configuration

$$\begin{align*}
\boxdot & \quad \boxslash \quad 0^q 1^q \cdots \quad : n(v) = 1 \\
\downarrow & \quad \downarrow \\
\phi & \quad \psi \\
\tau & \quad \eta \\
\#^q \mu C \# & : \text{where } C \in \Gamma^+ Q \Gamma^+
\end{align*}$$

$$(v = 0 \land y = \#) \mathsf{W} (v = 1 \land (v = 1 \land y = \#) \mathsf{W} (v = 0 \land y \in \Gamma^+ Q \Gamma^+ \#^\omega))$$

$$v = 0 \mathsf{W} (v = 1 \land X (v = 0 \rightarrow y \in C_1 \#^\omega))$$
Undecidability proof 3

**SPEC₃:** if \( n(u) = n(v) \) are synchronized then \( x = y \)

\[
\begin{array}{ccc}
0^{q+1} p 0 \ldots & a & b \\
\#^{q+1} p C \#^ω & y & \#^{q+1} p C \#^ω \\
\end{array}
\]

where \( n(u) = n(v) \equiv (u = v = 0) \lor (u = v = 1 \land (u = v = 1 \lor u = v = 0)) \)

Undecidability proof 4

**SPEC₄:** if \( n(u) = n(v) + 1 \) are synchronized then \( C_y \vdash C_y \)

\[
\begin{array}{ccc}
0^{q+1} p + 1 0 \ldots & a & b \\
\#^{q+1} p C_{x} \#^ω & y & \#^{q+1} p C_{x} \#^ω \\
\end{array}
\]

where \( n(u) = n(v) + 1 \rightarrow x = y \lor (\text{Trans}(y, x) \land X^3 G x = y) \)

Undecidability proof 5

**Lemma:** winning strategies must simulate the Turing machine

For each \( p \geq 1 \), if \( n(u) = p \) then \( C_x = C_p \) is the \( p \)-th configuration of the Turing machine starting from the empty tape.

**Proof**

\[
\begin{array}{ccc}
0^{q+1} p + 1 0 \ldots & a & b \\
\#^{q+1} p C_{p} \#^ω & y & \#^{q+1} p C_{p} \#^ω \\
\end{array}
\]

Corollary

Specifications 1-4 and 5: \( G x \neq \text{stop} \) are implementable iff the Turing machine does not halt starting from the empty tape.
Decidability of distributed synthesis

Some examples

Undecidable \quad Decidable \quad Undecidable

Decidability

Kupferman-Vardi (LICS’01)

The synthesis problem is non-elementary decidable for
one-way chain, one-way ring, two-way chain and two-way ring,
CTL* specifications (or tree-automata specifications) on all variables,
synchronous, 1-delay semantics,
local strategies.

1-delay synchronous semantics

Example

| x | u | l
|---|---|---
| a | b | c

Programs: \( f_x : Q^*_u \rightarrow Q_x \) and \( f_z : (Q_x \times Q_z)^* \rightarrow Q_z \).

Input:
\[
\begin{pmatrix}
  u_1 & u_2 & u_3 & \cdots \\
  v_1 & v_2 & v_3 & \cdots
\end{pmatrix}
\in (Q_u \times Q_v)^\omega.
\]

Behavior:
\[
\begin{pmatrix}
  u_1 & u_2 & u_3 & \cdots \\
  v_1 & v_2 & v_3 & \cdots \\
  x_1 & x_2 & x_3 & \cdots \\
  z_1 & z_2 & z_3 & \cdots
\end{pmatrix}
\]

with
\[
\begin{align*}
x_{n+1} &= f_x(u_1, \ldots, u_n) \\
z_{n+1} &= f_z((x_1, v_1) \cdots (x_n, v_n)) \quad \text{for all } n > 0.
\end{align*}
\]
Decidability

Adequately connected sub-architecture

\[ Q_x = Q \text{ for all } x \in V \]

Pnueli-Rosner (FOCS'90)

An adequately connected architecture is equivalent to a singleton architecture.

The synthesis problem is decidable for LTL specifications and pipelines of adequately connected architectures.

Outline

Synthesis and control for sequential systems

Synthesis and control for distributed systems

Uniformly well connected architectures

Definition

For an output variable \( y \), \( \text{View}(y) \) is the set of input variables \( x \) such that there is a path from \( x \) to \( y \).

Definition

An architecture is uniformly well connected if there is a uniform way to route variables in \( \text{View}(y) \) to \( y \) for each output variable \( y \).

Example
Uniformly well connected architectures

Definition
An architecture is uniformly well connected if there is a uniform way to route variables in View(\(v\)) to \(v\) for each output variable \(v\).

- If the capacity of internal variables is big enough then the architecture is uniformly well-connected.
- If the architecture is uniformly well-connected then we can use causal strategies instead of local ones.

Proposition
Checking whether a given architecture is uniformly well connected is NP-complete.

Proof
Reduction to the multicast problem in Network Information Flow.
The multicast problem is NP-complete (Rasala Lehman-Lehman 2004).

Uncomparable information

Definition
An architecture has uncomparable information if there exist \(y_1, y_2\) output variables such that View(\(y_2\)) \ \\ View(\(y_1\)) \ \neq \ \emptyset \ and \ View(\(y_1\)) \ \\ View(\(y_2\)) \ \neq \ \emptyset \.

Otherwise it is said to have preorder information.

Uncomparable information yields undecidability

Theorem
Architectures with uncomparable information are undecidable for LTL or CTL input-output specifications.

Proof.
Uniformly well connected architectures

Theorem (PG, Nathalie Sznajder, Marc Zeitoun)

Uniformly well connected architectures with preordered information are decidable for CTL* external specifications.

Proof.

Well-connected architectures

Definition

An architecture is well connected if, for each output variable \( y \), the subarchitecture formed by \((E^*)^{-1}(y)\) is uniformly well connected.

Example: well-connected but not UWC

Robust specifications

Definition

A specification \( \varphi \) is robust if it can be written \( \varphi = \bigvee_{x \in \text{Out}} \varphi_x \) where \( \varphi_x \) depends only on \( \text{View}(x) \cup \{ z \} \).

Theorem

The synthesis problem for uniformly well-connected architectures and external and robust CTL* specifications is decidable.

Proof.

Well-connected architectures

Definition

An architecture is well connected if, for each output variable \( y \), the subarchitecture formed by \((E^*)^{-1}(y)\) is uniformly well connected.

Rasala Lehman–Lehman 2004

One can solve the network information flow in the special case where there is a unique sink in polynomial time.

Corollary

One can decide whether an architecture is well-connected in polynomial time.
Well connected preordered architectures

Theorem
The synthesis problem for LTL specifications and well connected architectures with preordered information is undecidable.

Specification and routing

One bit of $u$ is hidden to $p_6$

Open problem

- Find a decidability criterion for external specifications and well-connected architectures.
- Find a decidability criterion for external specifications and arbitrary architectures.
- Decidability of the distributed control/synthesis problem for robust and external specifications.