Distributed Synthesis for LTL Fragments

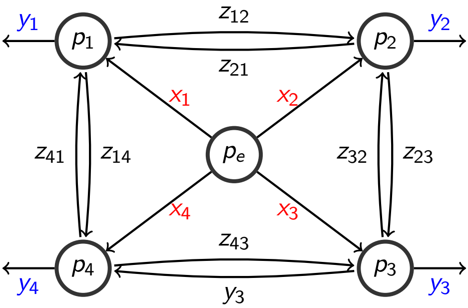
Krishnendu Chatterjee, Thomas A. Henzinger, Jan Otop,
Andreas Pavlogiannis

July 18, 2013

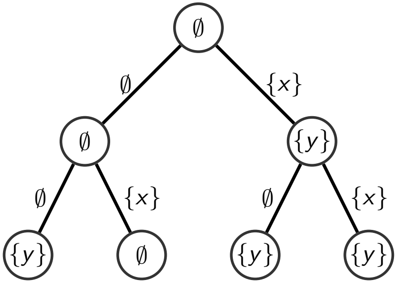
Reactive Distributed Systems

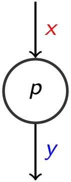
Synchronous architecture $\mathcal{A} = (\mathcal{P}, p_e, V, E)$

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



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





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

Realizability

- Every infinite path $\pi = (a_1, a_2 \dots)$ of T_σ defines a *computation* $\ell_\sigma(\pi) = (c_1, c_2 \dots)$.
  - $\ell_\sigma(\pi)[i]$ is the set of variables being True at that node.
- Acceptable computations are specified by LTL specifications.

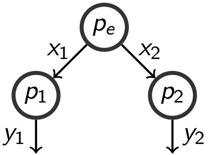
Realizability

Given an architecture $\mathcal{A}$ and an LTL specification ϕ, decide whether there exist local strategies σ_p for all processes p, such that for every path π in the corresponding collective strategy tree T_σ, it holds $\ell_\sigma(\pi) \models \phi$.

- If so, synthesize them.

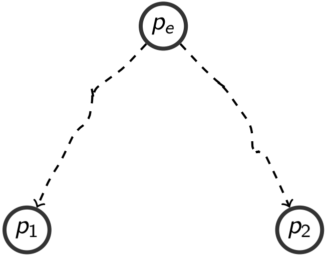
Distributed Realizability is Undecidable

- 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 $C_1 C_1'$ and $C_2 C_2'$ 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).

The Fragment $\mathrm{LTL}_{\Diamond}$

$\mathrm{LTL}_{\Diamond}$

For propositional formulae P and Q, we consider $\phi \in \mathrm{LTL}_{\Diamond}$ of the form

$$\begin{aligned}\theta &= P \mid \mathcal{X}P \\ \psi &= \theta_1 \wedge \theta_2 \mid \theta_1 \vee \theta_2 \mid \neg\theta \\ \phi &= Q \to \Diamond\psi\end{aligned}$$

Theorem

The realizability of specifications from $\mathrm{LTL}_{\Diamond}$ in some architecture $\mathcal{A}$ is decidable iff $\mathcal{A}$ does not have information fork.

Turing Machine Configurations

- Fix a Turing machine M, with tape alphabet $\{0, 1, \sqcup\}$ and set of states $\mathbf{Q}$. Let $\Sigma = \{0, 1, \sqcup, \perp, \#\} \cup \mathbf{Q}$.
- Configurations of M are words over $\Sigma \setminus \{\perp\}$ and start with $\#$.
- Projection $\pi_\perp : \Sigma^* \to (\Sigma \setminus \{\perp\})^*$ omits the $\perp$ symbols.
- A *scattered configuration* C is a word over Σ such that $\pi_\perp(C)$ is a configuration of M. Denote with $\perp(C) = \{i : C[i] = \perp\}$.
- A *scattered preconfiguration* is a word over Σ which is a prefix of some scattered configuration.
- We write $C_1 \parallel C_2$ if $|\perp(C_1) \triangle \perp(C_2)| \leq 1$.

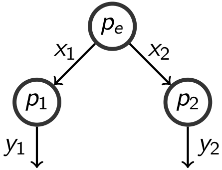
$C_1 \vdash C_2$

For scattered preconfigurations C_1 and C_2 we write $C_1 \vdash C_2$ if

1. $\pi_\perp(C_1) \vdash \pi_\perp(C_2)$, or
2. C_1 and C_2 are infinite and every finite prefix of $\pi_\perp(C_i)$ can be extended to C_i' such that $C_1' \vdash C_2'$.

Proof Idea

We consider the following architecture



- p_e sends *next* and *stall* signals.
- Processes output infinite words over Σ.
- Undecidability obtained through reduction from the halting problem of M.
- We first describe a safety property φ.

Proof Idea (Cont.)

$$\varphi = \mathcal{L} \to \bigwedge_{0 \leq i \leq 4} \mathit{Cond}_i$$

$\mathcal{L}$: for every process, every *stall* input signal is followed by a *next* signal.

Cond_0: each process outputs $\perp$ when its input is *stall*, otherwise it outputs a letter from $\Sigma \setminus \{\perp\}$,

Cond_1: each process produces a sequence of scattered preconfigurations,

Cond_2: initially each process produces two scattered configurations of M, whose projections are the first two valid configurations of M,

Cond_3: if starting from some position, p_1 outputs consecutively C_1, C_2 and p_2 outputs consecutively C'_1, C'_2, then $C_1 \vdash C'_1$ implies $C_2 \vdash C'_2$ or $C'_2 \nparallel C_2$,

Cond_4: if D, D' are outputs of p_1, p_2 up to some positions such that $D \parallel D'$ and $|\pi_\perp(D)| = |\pi_\perp(D')|$, then $\pi_\perp(D) = \pi_\perp(D')$.

Proof Idea (Cont)

- φ can be expressed by a safety automaton A_{safe}.

Lemma

Strategies realizing A_{safe} output a valid computation of M.

- $Cond_2$ requires the first two configurations.
- If C_1 and C_1' are the i and $i+1$ configurations of M, then p_e can synchronize them as outputs of p_1 and p_2.
- Needs to send at most i stalls to p_1, without violating $\mathcal{L}$.
- If no more stalls follow, by $Cond_3$, $C_1' \vdash C_1$ implies $C_2' \vdash C_2$.
- Due to $Cond_4$ we can show that for any execution under $\mathcal{L}$, $\pi_\perp(C_1')$ and $\pi_\perp(C_2')$ are the $i+1$ and $i+2$ configurations of M.

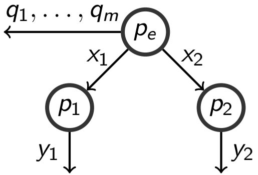
Proof Idea (Cont)

$$\phi = Q \rightarrow \Diamond(\psi_1 \vee \psi_2)$$

Q The first state of A_{safe} according to the output variables $\{q_1, \dots q_m\}$ corresponds to the first step of the computation.

ψ_1 p_e cheats in simulating A_{safe} .

ψ_2: The current state of A_{safe} is not rejecting, and p_1 or p_2 output a halting state of M.



ϕ is realizable iff M halts.

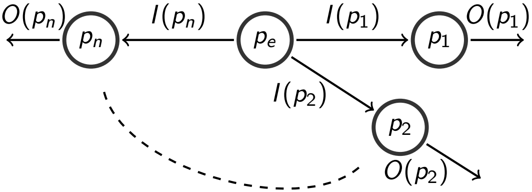
The Fragment $\text{LTL}_\square$

$\text{LTL}_\square$

For propositional formulae P and Q, we consider $\phi \in \text{LTL}_\square$ of the form

$$\psi = P \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \neg\psi \mid \mathcal{X}\psi$$
$$\phi = Q \wedge \square\psi$$

- Consider star architectures with p_e the central process.
- We distinguish between overlapping inputs (inter process communication), and disjoint inputs.



Undecidability for Overlapping Inputs

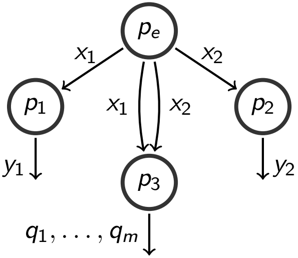
$$\phi = Q \wedge \Box(\psi_1 \wedge \psi_2 \wedge \psi_3)$$

Q: The first state of A_{safe} according to the output variables $\{q_1, \ldots q_m\}$ corresponds to the first step of the computation.

ψ_1: p_3 simulates A_{safe} faithfully in $\{q_1, \ldots q_m\}$.

ψ_2: p_1 and p_2 do not output a halting state of M.

ψ_3: A_{safe} does not reach a rejecting state.



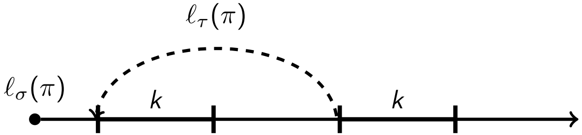
Decidability for Disjoint Inputs

- Let k be the nesting depth of $\mathcal{X}$ operators in ψ.

Lemma

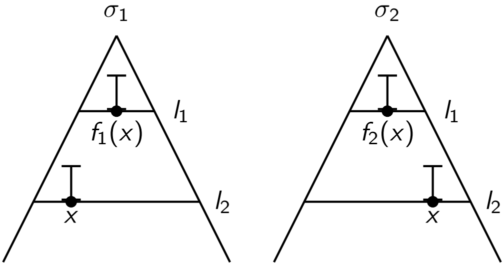
A formula $\phi = Q \land \Box\psi$ is realizable iff it is realizable by bounded strategies of depth $k + 2^{2^{k|V|}}$.

- Assume ϕ is realizable by local strategies σ_i.
- If some good computation $\ell_\sigma(\pi)$ repeats a k-segment then another computation $\ell_\tau(\pi)$ that loops between the two segments is also good.



Decidability for Disjoint Inputs (Cont.)

- The *type* of a local node is the unique history of inputs and outputs k steps back.
  - There exist at most $2^{k|V|}$ unique types of nodes.
- The type of a level is the set of types of all local nodes in that level.
  - There exist at most $2^{2^{k|V|}}$ unique types of levels.



Define folding functions f_i that folds levels with the same types, wrt the types of the nodes.