Distributed Synthesis for LTL Fragments

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

21 October 2013
An **architecture** is a directed graph describing topology of the system.

- **Communication** is done through variables $V$.
- Communication is **instantaneous**.
- Process $p$ has $I(p)$, $O(p)$, its **input** and **output** variables.
- Process $p$ behaves according to its **local strategy** $\sigma_p : (2^{I(p)})^* \rightarrow 2^{O(p)}$.
- $p_e$ is the **environment**.

- Local strategies give the **collective strategy** $\sigma : (2^{O(p_e)})^* \rightarrow 2^{V \setminus O(p_e)}$.

- Reactive system as a function: The **execution** of $\sigma$ on $\pi = a_1a_2 \ldots \in (2^{O(p_e)})^\omega$ is $\Gamma^\sigma(\pi) = \sigma(a_1)\sigma(a_1a_2) \ldots \in (2^{V \setminus O(p_e)})^\omega$
An **architecture** is a directed graph describing topology of the system.

- **Communication** is done through variables $V$.
- Communication is **instantaneous**.
- Process $p$ has $I(p)$, $O(p)$, its **input** and **output** variables.
- Process $p$ behaves according to its **local strategy** $\sigma_p : (2^{I(p)})^* \rightarrow 2^{O(p)}$.
- $p_e$ is the **environment**.

- Local strategies give the **collective strategy** $\sigma : (2^{O(p_e)})^* \rightarrow 2^{V \setminus O(p_e)}$.

- Reactive system as a function: The **execution** of $\sigma$ on $\pi = a_1a_2 \ldots \in (2^{O(p_e)})^\omega$ is $\Gamma^\sigma(\pi) = \sigma(a_1)\sigma(a_1a_2)\ldots \in (2^{V \setminus O(p_e)})^\omega$. 
Realizability

- A computation of $\sigma$ is the convolution of the environment output $\pi$ and the execution of $\sigma$, i.e., for $\pi = a_1 a_2 \ldots$ and $\Gamma^\sigma(\pi) = b_1 b_2 \ldots$ the computation is: $\pi \otimes \Gamma^\sigma(\pi) = (a_1, b_2)(a_2, b_2)\ldots \in (2^V)^\omega$

Satisfaction

A collective strategy $\sigma$ satisfies an LTL specification $\varphi$ iff its every computation satisfies $\varphi$, i.e., for every $\pi \in (2^{O(p_e)})^\omega$, $\pi \otimes \Gamma^\sigma(\pi) \models \varphi$.

Realizability

Given an architecture $\mathcal{A}$ and an LTL specification $\varphi$, decide whether there exist local strategies $\sigma_p$ for all processes $p$, that generate the collective strategy $\sigma$ that satisfy $\varphi$.

- If so, synthesize them.
Consider a specification

$$\varphi_1 \equiv □(x_1 \implies ♦y_1) \land □(x_2 \implies ♦y_2) \land □¬(y_1 \land y_2)$$

in the architecture:

It is realized by $σ_1, σ_2$ such that:

$σ_1(w) = \{y_1\}$ if $|w|$ is even and $\emptyset$ otherwise, and

$σ_2(w) = \{y_2\}$ if $|w|$ is odd and $\emptyset$ otherwise.
Consider a specification
\( \varphi_1 \equiv \Box(x_1 \implies \Diamond y_1) \land \Box(x_2 \implies \Diamond y_2) \land \Box \neg(y_1 \land y_2) \) in the
architecture:

It is realized by \( \sigma_1, \sigma_2 \) such that:
\( \sigma_1(w) = \{y_1\} \) if \( |w| \) is even and \( \emptyset \) otherwise, and
\( \sigma_2(w) = \{y_2\} \) if \( |w| \) is odd and \( \emptyset \) otherwise.

The following specification is not realizable
\( \varphi_2 \equiv (\Box \Diamond x_1 \implies \Box \Diamond (x_1 \land y_1)) \land (\Box \Diamond x_2 \implies \Box \Diamond (x_2 \land y_2)) \land \Box \neg(y_1 \land y_2). \)
Consider a specification
\[ \varphi_1 \equiv \Box(x_1 \implies \Diamond y_1) \land \Box(x_2 \implies \Diamond y_2) \land \Box \neg(y_1 \land y_2) \] in the architecture:

It is realized by \(\sigma_1, \sigma_2\) such that:
\[ \sigma_1(w) = \{y_1\} \text{ if } |w| \text{ is even and } \emptyset \text{ otherwise, and} \]
\[ \sigma_2(w) = \{y_2\} \text{ if } |w| \text{ is odd and } \emptyset \text{ otherwise.} \]

The following specification is not realizable
\[ \varphi_2 \equiv (\Box \Diamond x_1 \implies \Box \Diamond (x_1 \land y_1)) \land (\Box \Diamond x_2 \implies \Box \Diamond (x_2 \land y_2)) \land \Box \neg(y_1 \land y_2). \]

Suppose it is realizable.
\[
\begin{array}{cccccccc}
  x_1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
  y_1 & & & & & & & \\
  x_2 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
  y_2 & & & & & & & 
\end{array}
\]
Example

Consider a specification
\[ \varphi_1 \equiv \Box(x_1 \implies \Diamond y_1) \land \Box(x_2 \implies \Diamond y_2) \land \Box \neg (y_1 \land y_2) \] in the architecture:

It is realized by \( \sigma_1, \sigma_2 \) such that:
- \( \sigma_1(w) = \{y_1\} \) if \( |w| \) is even and \( \emptyset \) otherwise, and
- \( \sigma_2(w) = \{y_2\} \) if \( |w| \) is odd and \( \emptyset \) otherwise.

The following specification is not realizable
\[ \varphi_2 \equiv (\Box \Diamond x_1 \implies \Box \Diamond (x_1 \land y_1)) \land (\Box \Diamond x_2 \implies \Box \Diamond (x_2 \land y_2)) \land \Box \neg (y_1 \land y_2). \]

Suppose it is realizable.

\[
\begin{array}{cccccccc}
  x_1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
  y_1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
  x_2 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
  y_2 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]
Consider a specification
\[ \varphi_1 \equiv \Box (x_1 \implies \lozenge y_1) \land \Box (x_2 \implies \lozenge y_2) \land \Box \neg (y_1 \land y_2) \]
in the architecture:

It is realized by \( \sigma_1, \sigma_2 \) such that:
\[ \sigma_1(w) = \{y_1\} \text{ if } |w| \text{ is even and } \emptyset \text{ otherwise, and} \]
\[ \sigma_2(w) = \{y_2\} \text{ if } |w| \text{ is odd and } \emptyset \text{ otherwise.} \]

The following specification is not realizable
\[ \varphi_2 \equiv (\Box \lozenge x_1 \implies \Box \lozenge (x_1 \land y_1)) \land (\Box \lozenge x_2 \implies \Box \lozenge (x_2 \land y_2)) \land \Box \neg (y_1 \land y_2). \]

Suppose it is realizable.

\[
\begin{array}{cccccccc}
  x_1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
  y_1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
  x_2 & & & & & & & \\
  y_2 & & & & & & & \\
\end{array}
\]
Consider a specification
\[ \varphi_1 \equiv \Box(x_1 \implies \Diamond y_1) \land \Box(x_2 \implies \Diamond y_2) \land \Box \neg (y_1 \land y_2) \] in the architecture:

It is realized by \( \sigma_1, \sigma_2 \) such that:
\( \sigma_1(w) = \{y_1\} \) if \( |w| \) is even and \( \emptyset \) otherwise, and
\( \sigma_2(w) = \{y_2\} \) if \( |w| \) is odd and \( \emptyset \) otherwise.

The following specification is not realizable
\[ \varphi_2 \equiv (\Box \Diamond x_1 \implies \Box \Diamond (x_1 \land y_1)) \land (\Box \Diamond x_2 \implies \Box \Diamond (x_2 \land y_2)) \land \Box \neg (y_1 \land y_2). \]

Suppose it is realizable.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y_1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Consider a specification
\[ \varphi_1 \equiv \Box(x_1 \implies \Diamond y_1) \land \Box(x_2 \implies \Diamond y_2) \land \Box \neg (y_1 \land y_2) \] in the architecture:

It is realized by \( \sigma_1, \sigma_2 \) such that:
\[ \sigma_1(w) = \{y_1\} \text{ if } |w| \text{ is even and } \emptyset \text{ otherwise, and } \]
\[ \sigma_2(w) = \{y_2\} \text{ if } |w| \text{ is odd and } \emptyset \text{ otherwise.} \]

The following specification is not realizable
\[ \varphi_2 \equiv (\Box \Diamond x_1 \implies \Box \Diamond (x_1 \land y_1)) \land (\Box \Diamond x_2 \implies \Box \Diamond (x_2 \land y_2)) \land \Box \neg (y_1 \land y_2). \]

Suppose it is realizable.
\[
\begin{array}{cccccccc}
  x_1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
  y_1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
  x_2 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
  y_2 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
\end{array}
\]
Consider a specification
\[ \varphi_1 \equiv \Box(x_1 \implies \Diamond y_1) \land \Box(x_2 \implies \Diamond y_2) \land \Box \neg (y_1 \land y_2) \] in the architecture:

It is realized by \( \sigma_1, \sigma_2 \) such that:
\[
\sigma_1(w) = \{y_1\} \text{ if } |w| \text{ is even and } \emptyset \text{ otherwise, and }
\sigma_2(w) = \{y_2\} \text{ if } |w| \text{ is odd and } \emptyset \text{ otherwise.}
\]

The following specification is not realizable
\[ \varphi_2 \equiv (\Box \Diamond x_1 \implies \Box \Diamond (x_1 \land y_1)) \land (\Box \Diamond x_2 \implies \Box \Diamond (x_2 \land y_2)) \land \Box \neg (y_1 \land y_2). \]

Suppose it is realizable.

<table>
<thead>
<tr>
<th></th>
<th>( x_1 )</th>
<th>( y_1 )</th>
<th>( x_2 )</th>
<th>( y_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\( x_2 \) holds infinitely often, but only when \( y_1 \) holds!
Theorem (Pnueli, Rosner)

Realizability of LTL specifications on the following architecture $A_\lambda$ is undecidable.

For every Turing Machine $M$, there is a specification $\tau_M$, that forces $p_1, p_2$ to output the sequence of consecutive configurations of $M(\epsilon)$ terminated by the final configuration.
Undecidability

Theorem (Pnueli, Rosner)

Realizability of LTL specifications on the following architecture $A_\lambda$ is undecidable.

For every Turing Machine $M$, there is a specification $\tau_M$, that forces $p_1, p_2$ to output the sequence of consecutive configurations of $M(\epsilon)$ terminated by the final configuration.
Theorem (Pnueli, Rosner)

Realizability of LTL specifications on the following architecture $A_{\lambda}$ is undecidable.

For every Turing Machine $M$, there is a specification $\tau_M$, that forces $p_1, p_2$ to output the sequence of consecutive configurations of $M(\epsilon)$ terminated by the final configuration.
Undecidability

Theorem (Pnueli, Rosner)

Realizability of LTL specifications on the following architecture $A_\lambda$ is undecidable.

For every Turing Machine $M$, there is a specification $\tau_M$, that forces $p_1, p_2$ to output the sequence of consecutive configurations of $M(\epsilon)$ terminated by the final configuration.
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 $I(p_{-i})$.

**Theorem (Finkbeiner, Schewe)**

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

- LTL formulae that appear in the undecidability proof are complicated.

**Question**

What are the LTL fragments for which the realizability problem is decidable?

- That question can be approached from two directions:
  - Prove that realizability is undecidable in weak LTL fragments.
  - Find LTL fragments for which the realizability problem is decidable.
Reachability specifications \(\text{LTL}^\diamond\)

### \(\text{LTL}^\diamond\)

- \(\psi \in \text{LTL}_1\) iff it is a Boolean combination of \(P\) and \(\mathcal{X}P\), where \(P\) is propositional. (only non-nested \(\mathcal{X}\))
- \(\varphi \in \text{LTL}^\diamond\) iff \(\varphi \equiv Q \rightarrow \diamond \psi\), where \(\psi \in \text{LTL}_1\) and \(Q\) is propositional.

### Theorem

The realizability of specifications from \(\text{LTL}^\diamond\) in architectures containing information fork is undecidable.
Reachability specifications LTL\(\Box\)

\begin{itemize}
  \item \(\psi \in \text{LTL}_1\) iff it is a Boolean combination of \(P\) and \(\Box P\), where \(P\) is propositional. (only non-nested \(\Box\))
  \item \(\varphi \in \text{LTL}\Box\) iff \(\varphi \equiv Q \rightarrow \Box \psi\), where \(\psi \in \text{LTL}_1\) and \(Q\) is propositional.
\end{itemize}

**Theorem**

The realizability of specifications from LTL\(\Box\) in architectures containing information fork is undecidable.

- \(\tau_M\) is a (variant of) formula that forces \(p_1, p_2\) to output a computation of a TM \(M\).
- A safety automaton \(A_{\text{safe}}\) recognizes \(L_{\tau_M}\).
- Specification \(\gamma \in \text{LTL}\Box\) states that eventually
  - \(p_e\) (does not) simulate \(A_{\text{safe}}\) with \(q_1, \ldots, q_k\),
  - \(p_1\) outputs the final configuration.
Safety specifications $\text{LTL}^\Box$ over Overlapping Inputs

### $\text{LTL}^\Box$

- $\psi \in \text{LTL}_1$ iff it is a Boolean combination of $P$ and $\exists P$, where $P$ is propositional. (only non-nested $\exists$)
- $\varphi \in \text{LTL}^\Box$ iff $\varphi \equiv Q \land \Box \psi$, where $\psi \in \text{LTL}_1$ and $Q$ is propositional.

### Theorem

The realizability of specifications from $\text{LTL}^\Box$ in an architecture $A$ containing an information fork-meet is undecidable.

- The proof is as for $\text{LTL}^\Diamond$, but $p_3$ simulates $A_{\text{safe}}$ instead of $p_e$, i.e.:
  - A safety automaton $A_{\text{safe}}$ recognizes $\mathcal{L}_{TM}$.
  - Specification $\gamma \in \text{LTL}^\Box$ ensures that $p_3$ simulates $A_{\text{safe}}$. 
Consider a class of **star architectures with disjoint inputs:**

![Diagram of star architectures](image)

**Lemma**

A formula $\phi = Q \land \Box \psi$ is realizable iff it is realizable by strategies with double exponential memory.

**Sufficiently long plays can be repeated.**

**Theorem**

Realizability of $\text{LTL}_{\Box}$ specifications on star architectures with disjoint inputs is in $\text{EXPSPACE}$. 
Fragments of LTL without $\mathcal{X}$

$LTL_{AG}$

$\varphi \in LTL_{AG}$ if for propositional formulae $P, Q, R_i, F_i$, $\varphi$ is of the form

$$\varphi = \Box P \rightarrow \Box Q \land \bigwedge_i \Box \Diamond R_i \land \bigwedge_i \Diamond F_i$$

**Theorem**

Realizability of $LTL_{AG}$ specifications is NEXPTIME-complete.
Fragments of LTL without $\mathcal{X}$

**LTL$\mathcal{AG}$**

$\varphi \in \text{LTL}^\mathcal{AG}$ if for propositional formulae $P, Q, R_i, F_i$, $\varphi$ is of the form

$$\varphi = \Box P \rightarrow \Box Q \land \bigwedge_i \Box \Diamond R_i \land \bigwedge_i \Diamond F_i$$

**Theorem**

Realizability of LTL$\mathcal{AG}$ specifications is NEXPTIME-complete.

- $\varphi \in \text{LTL}^\mathcal{AG}$ is realizable iff every formula $\Box(P \rightarrow Q \land R_i)$ and every $\Box(P \rightarrow Q \land F_i)$ are realizable.
- $\Box Q$ is realizable iff it is realizable by memoryless strategies.
- Realizability of LTL$\mathcal{AG}$ is in NEXPTIME.
Fragments of LTL without $\mathcal{X}$

$LTL_{AG}$

$\varphi \in LTL_{AG}$ if for propositional formulae $P, Q, R_i, F_i$, $\varphi$ is of the form

$$\varphi = \square P \rightarrow \square Q \land \bigwedge_i \square \lozenge R_i \land \bigwedge_i \lozenge F_i$$

Theorem

Realizability of $LTL_{AG}$ specifications is $\text{NEXPTIME}$-complete.

Dependency Quantified Boolean Formulas (DQBF) are propositional formulae with Henkin quantifiers.

$$\forall x_1 \forall x_2 \exists y_1(x_1) \exists y_2(x_2). Q(x_1, x_2, y_1, y_2)$$

- Validity of DQBF is $\text{NEXPTIME}$-complete.
- DQBF reduces to realizability of $LTL_{AG}$
Conclusions

Our contributions:

- Distributed synthesis is undecidable, even restricted to simple LTL fragments: $\text{LTL}^{\lozenge}$, $\text{LTL}^{\square}$.

- $\text{LTL}^{\square}$ is decidable in NEXPSpace on the class of star architectures with disjoint inputs.

- $\text{LTL}_{AG}$ is NEXPTIME-complete.

- $\text{LTL}_{AG}$ reduces to DQBF and vice versa.

Thank you!