Distributed synthesis over LTL fragment

Giuseppe Perelli

Rice University

Comp607 Seminary
Houston, February 14th - 2014
(Happy S.Valentine Day!)
Computations

Definition
A computation is an infinite path of assignments over a (finite) set $\mathbf{V}$ of binary variables.

Example
For $\mathbf{V} = \{x_1, x_2, x_3, y_1, y_2, y_3\}$, then a possible computation is of the form

$$\pi = \{x_1, y_2, y_3\} \cdot \{x_2, y_1, y_2, y_3\} \cdot \ldots \cdot \{x_1, x_2, y_1, y_3\} \cdot \ldots,$$

where the variables in each set $\mathbf{V}_i \subseteq \mathbf{V}$ of the sequence are assigned to true and the others in $\mathbf{V} \setminus \mathbf{V}_i$ are assigned to false.
Temporal properties of computations

A computation $\pi$ can be evaluated under certain temporal properties.

Example

- Two variables can never be both true at the same time. (Critics avoidance);
Temporal properties of computations

A computation $\pi$ can be evaluated under certain temporal properties.

Example

- Two variables can never be both true at the same time. (Critics avoidance);
- Every occurrence of variable $x$ is followed by an occurrence of variable $y$ (System guarantees);
Temporal properties of computations

A computation $\pi$ can be evaluated under certain temporal properties.

Example

- Two variables can never be both true at the same time. (Critics avoidance);
- Every occurrence of variable $x$ is followed by an occurrence of variable $y$ (System guarantees);
- The variable $x$ is never true in the computation $\pi$ (Safety);
- The variable $x$ occurs infinitely often in the computation $\pi$ (Safety).
Temporal properties of computations

A computation $\pi$ can be evaluated under certain temporal properties.

Example

- Two variables can never be both true at the same time. (Critics avoidance);
- Every occurrence of variable $x$ is followed by an occurrence of variable $y$ (System guarantees);
- The variable $x$ is never true in the computation $\pi$ (Safety);
- The variable $x$ is true until the variable $y$ occurs in the computation $\pi$ (Safety);
Temporal properties of computations

A computation $\pi$ can be evaluated under certain temporal properties.

Example

- Two variables can never be both true at the same time. (Critics avoidance);
- Every occurrence of variable $x$ is followed by an occurrence of variable $y$ (System guarantees);
- The variable $x$ is never true in the computation $\pi$ (Safety);
- The variable $x$ is true until the variable $y$ occurs in the computation $\pi$ (Safety);
- The variable $x$ occurs infinitely often in the computation $\pi$ (Safety).
Temporal properties of computations

A computation $\pi$ can be evaluated under certain temporal properties.

**Example**

- Two variables can never be both true at the same time. *(Critics avoidance)*;
- Every occurrence of variable $x$ is followed by an occurrence of variable $y$ *(System guarantees)*;
- The variable $x$ is never true in the computation $\pi$ *(Safety)*;
- The variable $x$ is true until the variable $y$ occurs in the computation $\pi$ *(Safety)*;
- The variable $x$ occurs infinitely often in the computation $\pi$ *(Safety)*.
Temporal properties of computations

A computation \( \pi \) can be evaluated under certain temporal properties.

**Example**

- Two variables can never be both true at the same time. (Critics avoidance);
- Every occurrence of variable \( x \) is followed by an occurrence of variable \( y \) (System guarantees);
- The variable \( x \) is never true in the computation \( \pi \) (Safety);
- The variable \( x \) is true until the variable \( y \) occurs in the computation \( \pi \) (Safety);
- The variable \( x \) occurs infinitely often in the computation \( \pi \) (Safety).

**Question**

How to formally represent such temporal properties?
Linear Temporal Logic (\(\text{LTL}, [\text{Pnueli, '77}]\))

Syntax

\[
\phi ::= p \mid \neg \phi \mid \phi \land \phi \mid \phi \lor \phi
\]
Linear Temporal Logic (LTL, [Pnueli, ’77])

Syntax

\[ \varphi ::= p \mid \neg \varphi \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X \varphi \mid \lozenge \varphi \mid \square \varphi \mid \varphi U \varphi \]

Semantics
Linear Temporal Logic (LTL, [Pnueli, ’77])

Syntax
ϕ ::= p | ¬ϕ | ϕ ∧ ϕ | ϕ ∨ ϕ | Xϕ | ◊ϕ | □ϕ | ϕUϕ

Semantics

\[ Xϕ \]
Linear Temporal Logic (\textbf{LTL}, [Pnueli, ’77])

Syntax

\[ \varphi ::= p \mid \neg \varphi \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X \varphi \mid \Diamond \varphi \mid \Box \varphi \mid \varphi U \varphi \]

Semantics

\[ \begin{array}{l}
X \varphi \\
\Diamond \varphi
\end{array} \]

\[ \begin{array}{l}
\varphi \\
\varphi
\end{array} \]
Linear Temporal Logic (LTL, [Pnueli, ’77])

Syntax

\[ \varphi ::= p \mid \neg \varphi \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X \varphi \mid \Diamond \varphi \mid \Box \varphi \mid \varphi U \varphi \]

Semantics

- **X \varphi**
  - \( V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow \ldots \rightarrow V_i \rightarrow V_k \)

- **Diamond \varphi**
  - \( V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow \ldots \rightarrow V_i \rightarrow V_k \)

- **Box \varphi**
  - \( V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow \ldots \rightarrow V_i \rightarrow V_k \)
Linear Temporal Logic (LTL, [Pnueli, ’77])

Syntax

\( \varphi ::= p \mid \neg \varphi \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X \varphi \mid \Diamond \varphi \mid \square \varphi \mid \varphi U \varphi \)

Semantics

- **\( X \varphi \)**
  - \( V_0 \) \( \rightarrow \) \( V_1 \) \( \rightarrow \) \( V_2 \) \( \rightarrow \) \( V_i \) \( \rightarrow \) \( V_k \)

- **\( \Diamond \varphi \)**
  - \( V_0 \) \( \rightarrow \) \( V_1 \) \( \rightarrow \) \( V_2 \) \( \rightarrow \) \( V_i \) \( \rightarrow \) \( V_k \)

- **\( \square \varphi \)**
  - \( V_0 \) \( \rightarrow \) \( V_1 \) \( \rightarrow \) \( V_2 \) \( \rightarrow \) \( V_i \) \( \rightarrow \) \( V_k \)

- **\( \varphi U \psi \)**
  - \( V_0 \) \( \rightarrow \) \( V_1 \) \( \rightarrow \) \( V_2 \) \( \rightarrow \) \( V_i \) \( \rightarrow \) \( V_k \)
Example of LTL specifications

- Two variables $x$ and $y$ can never be both true at the same time.
Example of LTL specifications

- Two variables $x$ and $y$ can never be both true at the same time.
  $\Box \neg (x \land y)$
Example of LTL specifications

- Two variables $x$ and $y$ can never be both true at the same time.
  $$\square \neg (x \land y)$$
- Each occurrence of variable $x$ is followed by an occurrence of variable $y$. 
  $$x \rightarrow x U y$$
Example of LTL specifications

- Two variables $x$ and $y$ can never be both true at the same time.
  $\Box \neg(x \land y)$

- Each occurrence of variable $x$ is followed by an occurrence of variable $y$.
  $\Box (x \rightarrow X \Diamond y)$
Example of LTL specifications

- Two variables $x$ and $y$ can never be both true at the same time.
  \[ \Box \neg (x \land y) \]
- Each occurrence of variable $x$ is followed by an occurrence of variable $y$.
  \[ \Box (x \rightarrow X \Diamond y) \]
- The variable $x$ is never true in the computation $\pi$.
  \[ \neg x \]
- The variable $x$ occurs infinitely often in the computation $\pi$.
  \[ x U y \]
Example of LTL specifications

- Two variables $x$ and $y$ can never be both true at the same time.
  $\square \neg(x \land y)$

- Each occurrence of variable $x$ is followed by an occurrence of variable $y$.
  $\square (x \rightarrow X \diamond y)$

- The variable $x$ is never true in the computation $\pi$.
  $\square \neg x$
Example of LTL specifications

- Two variables $x$ and $y$ can never be both true at the same time.
  \[ \square \neg (x \land y) \]

- Each occurrence of variable $x$ is followed by an occurrence of variable $y$.
  \[ \square (x \rightarrow X \diamond y) \]

- The variable $x$ is never true in the computation $\pi$.
  \[ \square \neg x \]

- The variable $x$ is true until the variable $y$ occurs in the computation $\pi$.
  \[ x U y \]
Example of LTL specifications

- Two variables $x$ and $y$ can never be both true at the same time.
  $\Box \neg (x \land y)$
- Each occurrence of variable $x$ is followed by an occurrence of variable $y$.
  $\Box (x \rightarrow X \diamond y)$
- The variable $x$ is never true in the computation $\pi$.
  $\Box \neg x$
- The variable $x$ is true until the variable $y$ occurs in the computation $\pi$.
  $x U y$
Example of LTL specifications

- Two variables $x$ and $y$ can never be both true at the same time.
  $\square \neg (x \land y)$

- Each occurrence of variable $x$ is followed by an occurrence of variable $y$.
  $\square (x \rightarrow X \diamond y)$

- The variable $x$ is never true in the computation $\pi$.
  $\square \neg x$

- The variable $x$ is true until the variable $y$ occurs in the computation $\pi$.
  $x U y$

- The variable $x$ occurs infinitely often in the computation $\pi$. 
Example of LTL specifications

- Two variables $x$ and $y$ can never be both true at the same time.
  $\square \neg (x \land y)$

- Each occurrence of variable $x$ is followed by an occurrence of variable $y$.
  $\square (x \rightarrow X \Diamond y)$

- The variable $x$ is never true in the computation $\pi$.
  $\square \neg x$

- The variable $x$ is true until the variable $y$ occurs in the computation $\pi$.
  $xU y$

- The variable $x$ occurs infinitely often in the computation $\pi$.
  $\square \Diamond x$
Other examples

- $\diamond \square X$
- $X \land \square (X \rightarrow X X) - x$ is true on at least all the odd positions;
- $X \land \square (X \rightarrow X X) - x$ is true on at least all the even positions.
Other examples

- $\Diamond \Box x$ - from a certain point $x$ will be always true;
Other examples

- $\Diamond \Box x$ - from a certain point $x$ will be always true;
- $x \land \Box (x \rightarrow XX x)$

Giuseppe Perelli
Rice University
Distributed synthesis over LTL fragment
Other examples

- $\Diamond \Box x$ - from a certain point $x$ will be always true;
- $x \land \Box (x \rightarrow XX x)$ - $x$ is true on at least all the odd positions;
Other examples

- \( \Diamond \Box x \) - from a certain point \( x \) will be always true;
- \( x \land \Box (x \rightarrow XX x) \) - \( x \) is true on at least all the odd positions;
- \( XX x \land \Box (x \rightarrow XX x) \) - \( x \) is true on at least all the even positions.
Satisfiability

Definition
A computation \( \pi \) that follows the LTL specification \( \varphi \) is said to satisfy \( \varphi \).

Definition
A set of computations \( \Pi \) satisfies an LTL specification \( \varphi \) if all the computations \( \pi \in \Pi \) satisfy \( \varphi \).
Architectures

Definition
An **architecture** is a labeled directed graph describing the topology of the system.

- $p_e$ is the **environment** process;
- **Instantaneous communication** among processes is managed with a set $V$ of binary variables;
- Process $p$ reads its input variables $I(p)$ and writes its output variables $O(p)$;
- Every sequence of truth-assignments of variables over time-steps corresponds to a computation over $V$;
Architectures

Definition

An **architecture** is a labeled directed graph describing the topology of the system.

- A **local strategy** for the system process $p$ establishes what to output according to the history of the input. $\sigma_p : (2^{I(p)})^* \rightarrow 2^{O(p)}$;
- By combining local strategies of system processes, we obtain a **collective strategy**, which depends on the environment input histories $\sigma : (2^{O(\pi_e)})^* \rightarrow 2^{V \setminus O(\pi_e)}$
Architectures

Definition
An architecture is a labeled directed graph describing the topology of the system.

\[
\begin{align*}
\sigma_1(\alpha \cdot a) &= (a, 0); \\
\sigma_2(\alpha \cdot a, \beta) &= (a, 1 - a); \\
\sigma_3(\alpha, \beta, \gamma) &= (1).
\end{align*}
\]
Architectures

Definition
An architecture is a labeled directed graph describing the topology of the system.

\[ \sigma_1(\alpha \cdot a) = (a, 0); \]
\[ \sigma_2(\alpha \cdot a, \beta) = (a, 1 - a); \]
\[ \sigma_3(\alpha, \beta, \gamma) = (1). \]
Architectures

Definition
An architecture is a labeled directed graph describing the topology of the system.

Definition
An architecture is a labeled directed graph describing the topology of the system.

\[ \sigma_1(\alpha \cdot a) = (a, 0); \]
\[ \sigma_2(\alpha \cdot a, \beta) = (a, 1 - a); \]
\[ \sigma_3(\alpha, \beta, \gamma) = (1). \]

\[ x_1 \quad 0 \]
\[ x_2 \quad 1 \]
\[ y_1 \quad 0 \]
\[ y_2 \quad 0 \]
\[ y_3 \quad 1 \]
\[ y_4 \]
\[ y_5 \quad 1 \]
Architectures

Definition
An architecture is a labeled directed graph describing the topology of the system.

\[
\begin{align*}
\sigma_1(\alpha \cdot a) &= (a, 0); \\
\sigma_2(\alpha \cdot a, \beta) &= (a, 1 - a); \\
\sigma_3(\alpha, \beta, \gamma) &= (1).
\end{align*}
\]
## Architectures

### Definition

An **architecture** is a labeled directed graph describing the topology of the system.

\[
\sigma_1(\alpha \cdot a) = (a, 0);
\]
\[
\sigma_2(\alpha \cdot a, \beta) = (a, 1 - a);
\]
\[
\sigma_3(\alpha, \beta, \gamma) = (1).
\]
A collective strategy $\sigma$ identifies a set of computations $\Pi(\sigma)$, given by the all possible behaviors over time of the environment process.

**Definition**
A collective strategy $\sigma$ satisfies an LTL formula $\varphi$ if the set $\Pi(\sigma)$ of computations satisfies $\varphi$.

**Realizability problem**
Given an architecture $\mathcal{A}$ and an LTL formula $\varphi$, decide whether there exist local strategies $\sigma_p$, for all processes $p$, which generate the collective strategy $\sigma$ that satisfies $\varphi$. Moreover, if so, synthesize it.
Consider a specification

\[ \varphi_1 \equiv \square(x_1 \implies \Diamond y_1) \land \square(x_2 \implies \Diamond y_2) \land \square \neg(y_1 \land y_2) \]

in the architecture:

![Diagram](image)

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.
Consider a specification
\( \varphi_1 \equiv [\neg x_1 \implies \lozenge y_1] \land [\neg x_2 \implies \lozenge y_2] \land [\neg (y_1 \land y_2)] \) 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
\( \varphi_2 \equiv ([\neg \lozenge x_1 \implies [\neg \lozenge (x_1 \land y_1)] \land ([\neg \lozenge x_2 \implies [\neg \lozenge (x_2 \land y_2)] \land [\neg (y_1 \land y_2)]). \)
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:
\[
\sigma_1(w) = \{ y_1 \} \text{ if } |w| \text{ is even and } \emptyset \text{ otherwise, and }
\sigma_2(w) = \{ y_2 \} \text{ if } |w| \text{ is odd and } \emptyset \text{ otherwise.}
\]

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.
\[
\begin{array}{cccccccc}
  x_1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
  y_1 & & & & & & & \\
  x_2 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
  y_2 & & & & & & & 
\end{array}
\]
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:

\[
\begin{array}{c}
p_e \\
p_1 & p_2 \\
\downarrow y_1 & \downarrow y_2 \\
x_1 & x_2 \\
p_1 \end{array}
\]

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.

\[
\begin{array}{cccccccc}
x_1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
y_1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
x_2 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
y_2 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]
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:

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

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.

\[
\begin{array}{cccccccc}
  x_1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
  y_1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
  y_2 & & & & & & & \\
  x_2 & & & & & & & \\
\end{array}
\]
Example

Consider a specification
\[ \varphi_1 \equiv \Box(x_1 \Rightarrow \Diamond y_1) \land \Box(x_2 \Rightarrow \Diamond y_2) \land \Box \neg (y_1 \land y_2) \] 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
\[ \varphi_2 \equiv (\Box \Diamond x_1 \Rightarrow \Box \Diamond (x_1 \land y_1)) \land (\Box \Diamond x_2 \Rightarrow \Box \Diamond (x_2 \land y_2)) \land \Box \neg (y_1 \land y_2). \]

Suppose it is realizable.

| \( x_1 \) | 1 | 0 | 1 | 0 | 1 | 0 | 1 |   |
| \( y_1 \) | 1 | 0 | 0 | 0 | 1 | 0 | 1 |   |
| \( x_2 \) | \downarrow | \downarrow | \downarrow | \downarrow | \downarrow | \downarrow | \downarrow | \downarrow |
| \( y_2 \) |   |   |   |   |   |   |   |   |
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:

\( \sigma_1(w) = \{ y_1 \} \) 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.

\[
\begin{array}{cccccccc}
  x_1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
  y_1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
  x_2 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
  y_2 & & & & & & & \\
\end{array}
\]
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.

\[
\begin{array}{ccccccccc}
x_1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
y_1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
x_2 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
y_2 & & & & & & & &
\end{array}
\]

\( x_2 \) holds infinitely often, but only when \( y_1 \) holds!
The power of sharing input variables

Consider the LTL formula $\varphi = \square((x_1 \leftrightarrow y_1) \land (y_1 \leftrightarrow \neg y_2))$ and the two architectures below.

![Diagram of two architectures](image-url)
The power of sharing input variables

Consider the LTL formula $\varphi = \Box ((x_1 \leftrightarrow y_1) \land (y_1 \leftrightarrow \neg y_2))$ and the two architectures below.

$$\sigma_{p_1}(\alpha \cdot a) = a$$
The power of sharing input variables

Consider the LTL formula $\varphi = \square ((x_1 \leftrightarrow y_1) \land (y_1 \leftrightarrow \neg y_2))$ and the two architectures below.

$$\sigma_{p_1}(\alpha \cdot a) = a$$
$$\sigma_{p_2}(\alpha \cdot a) = 1 - a.$$
The power of sharing input variables

Consider the LTL formula $\varphi = \Box((x_1 \leftrightarrow y_1) \land (y_1 \leftrightarrow \neg y_2))$ and the two architectures below.

$$\sigma_{p_1}(\alpha \cdot a) = a$$
$$\sigma_{p_2}(\alpha \cdot a) = 1 - a.$$

Not realizable. Process $p_2$ cannot deduce anything on variable $y_1$. 
For which classes of architectures is realizability decidable?

Complete characterization based on the *information fork* criterion.

Processes $p_1$, $p_2$ form an information fork in architecture $A$ if there exist paths $p_e \sim p_i$ in $A$ 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 \( \text{LTL} \downarrow \)

### \( \text{LTL} \downarrow \)

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

### Theorem

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

<table>
<thead>
<tr>
<th>( \text{LTL} \diamond )</th>
</tr>
</thead>
<tbody>
<tr>
<td>- ( \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} ))</td>
</tr>
<tr>
<td>- ( \varphi \in \text{LTL} \diamond ) iff ( \varphi \equiv Q \to \diamond \psi ), where ( \psi \in \text{LTL}_1 ) and ( Q ) is propositional.</td>
</tr>
</tbody>
</table>

### Theorem

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

\[
\begin{aligned}
&\text{\( q_1, \ldots, q_m \)} \\
&\text{\( p_e \)} \\
&\text{\( p_1 \)} \\
&\text{\( p_2 \)} \\
&\text{\( y_1 \)} \\
&\text{\( y_2 \)} \\
&\text{\( \tau_M \)} \\
&\text{A safety automaton} \ A_{\text{safe}} \text{ recognizes} \ L_{\tau_M}. \\
&\text{Specification} \ \gamma \in \text{LTL} \diamond \text{ states that eventually} \\
&\text{\( p_e \) (does not) simulate} \ A_{\text{safe}} \text{ with} \ q_1, \ldots, q_k, \\
&\text{\( p_1 \) outputs the final configuration.}
\end{aligned}
\]
Safety specifications LTL\(\square\) over Overlapping Inputs

**LTL\(\square\)**

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

**Theorem**

The realizability of specifications from LTL\(\square\) in an architecture \(A\) containing an information fork-meet is undecidable.

The proof is as for LTL\(\Diamond\), but \(p_3\) simulates \(A_{\text{safe}}\) instead of \(p_e\), i.e.:

- A safety automaton \(A_{\text{safe}}\) recognizes \(L_{\tau_M}\).
- Specification \(\gamma \in \text{LTL}\_\square\) ensures that \(p_3\) simulates \(A_{\text{safe}}\).
Consider a class of star architectures with disjoint inputs:

\[ O(p_n) \rightarrow I(p_n) \leftarrow I(p_1) \rightarrow O(p_1) \rightarrow I(p_2) \rightarrow O(p_2) \]

**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 $\mathcal{X}$

$LTL_{AG}$

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

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

Theorem

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

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

**LTL\textsubscript{AG}**

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

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

**Theorem**

Realizability of LTL\textsubscript{AG} specifications is 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 NEXPTIME-complete.
- DQBF reduces to realizability of LTL\textsubscript{AG}
Conclusions

Our contributions:

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