Assume-Guarantee Synthesis for Prompt Linear Temporal Logic

Bastien Maubert\textsuperscript{1}

joint work with
Nathanaël Fijalkow\textsuperscript{2}, Aniello Murano\textsuperscript{1} & Moshe Vardi\textsuperscript{3}

\textsuperscript{1} University of Naples
\textsuperscript{2} CNRS & LaBRI, Bordeaux
\textsuperscript{3} Rice University, Houston
Reactive synthesis

- $I$: finite set of inputs,
- $O$: finite set of outputs

Game between system and environment
- Environment chooses an input in $I$
- System chooses an output in $O$
Reactive synthesis

- $I$: finite set of inputs,
- $O$: finite set of outputs

**Game between system and environment**

- Environment chooses an input in $I$
- System chooses an output in $O$
Reactive synthesis

- $I$: finite set of inputs,
- $O$: finite set of outputs

Game between system and environment

- Environment chooses an input in $I$
- System chooses an output in $O$

$i_0$

$o_0$
Reactive synthesis

- $I$: finite set of inputs,
- $O$: finite set of outputs

Game between system and environment

- Environment chooses an input in $I$
- System chooses an output in $O$

$$i_0, i_1$$
$$o_0$$
Reactive synthesis

- $I$: finite set of inputs,
- $O$: finite set of outputs

Game between system and environment
- Environment chooses an input in $I$
- System chooses an output in $O$

$i_0$ $i_1$
$o_0$ $o_1$
Reactive synthesis

- $I$: finite set of inputs,
- $O$: finite set of outputs

Game between system and environment

- Environment chooses an input in $I$
- System chooses an output in $O$

\[
\begin{array}{ccc}
i_0 & i_1 & i_2 \\
o_0 & o_1 &
\end{array}
\]
Reactive synthesis

- $I$: finite set of inputs,
- $O$: finite set of outputs

Game between system and environment

- Environment chooses an input in $I$
- System chooses an output in $O$

\[
\begin{align*}
i_0 & \quad i_1 & \quad i_2 \\
o_0 & \quad o_1 & \quad o_2
\end{align*}
\]
Reactive synthesis

- $I$: finite set of inputs,
- $O$: finite set of outputs

**Game between system and environment**

- Environment chooses an input in $I$
- System chooses an output in $O$

$i_0 \quad i_1 \quad i_2 \quad \ldots$

$o_0 \quad o_1 \quad o_2 \quad \ldots$
Reactive synthesis

- $I$: finite set of inputs,
- $O$: finite set of outputs

Game between system and environment

- Environment chooses an input in $I$
- System chooses an output in $O$

\[
i_0 \ i_1 \ i_2 \ \ldots \\
o_0 \ o_1 \ o_2 \ \ldots
\]

LTL synthesis problem

Given a specification $\varphi \in \text{LTL}$ over $I \cup O$, synthesize a system $S : I^* \rightarrow O$ such that all resulting executions satisfy $\varphi$. 
Prompt LTL

LTL + operator of *bounded eventuality*

**Syntax:**

\[ \varphi ::= p \mid \neg p \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid X \varphi \mid \varphi U \varphi \mid \varphi R \varphi \mid F^{\leq B} \varphi \]

**Semantics:**

Semantics of \( \varphi \) in an infinite word \( w \), at position \( i \in \mathbb{N} \), with bound \( b \in \mathbb{N} \): all cases as for LTL, plus

\[
\begin{align*}
    w, i, b & \models F^{\leq B} \varphi \quad \text{if} \quad \exists j \in [i, i + b] \text{ such that } w, j, b \models \varphi \\
    w & \models \varphi \quad \text{if} \quad \exists b \in \mathbb{N} \text{ such that } w, 0, b \models \varphi
\end{align*}
\]

**Example**

Bound the waiting time of a printer server

\[ G(\text{request} \Rightarrow F\text{granted}) \]
Prompt LTL

LTL + operator of *bounded eventuality*

**Syntax:**

\[ \varphi ::= p \mid \neg p \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid X\varphi \mid \varphi U \varphi \mid \varphi R \varphi \mid F^{\leq B} \varphi \]

**Semantics:**

Semantics of \( \varphi \) in an infinite word \( w \), at position \( i \in \mathbb{N} \), with bound \( b \in \mathbb{N} \): all cases as for LTL, plus

\[
w, i, b \models F^{\leq B} \varphi \quad \text{if} \quad \exists j \in [i, i + b] \text{ such that } w, j, b \models \varphi
\]

\[
w \models \varphi \quad \text{if} \quad \exists b \in \mathbb{N} \text{ such that } w, 0, b \models \varphi
\]

**Example**

Bound the waiting time of a printer server

\[ G(\text{request } \Rightarrow F^{\leq B} \text{granted}) \]
Assume-guarantee synthesis for Prompt-LTL

**Input**
- Assumption: Prompt-LTL formula \( \varphi \) over \( I \)
- Guarantee: Prompt-LTL formula \( \psi \) over \( I \times O \)

**Question**
Is it the case that for all \( b \in \mathbb{N} \), there exists \( b' \in \mathbb{N} \) and a system \( S : I^* \rightarrow O \) such that for all \( w \in I^\omega \),

\[
\text{if } w, b \models \varphi \text{ then } S(w), b' \models \psi
\]

Problem open for a decade.
Contribution

Main result
The assume-guarantee synthesis problem for PROMPT-LTL is 2EXPTIME-complete.

Approach:
Similar to classical automata-based approach to LTL synthesis:
1. Build an automaton $A_\varphi$ for the specification $\varphi$
2. Determinise $A_\varphi$
3. Solve a game on $A^{\text{det}}_\varphi$

Crucial elements:
- Use of cost functions and cost automata (Colcombet et al.)
- New result on history-determinisation of a subclass of cost automata on infinite words
Contribution

Main result
The assume-guarantee synthesis problem for PROMPT-LTL is 2EXPTIME-complete.

Approach:
Similar to classical automata-based approach to LTL synthesis:
1. Build automata $A_\varphi$ and $A_\psi$ for $\varphi$ and $\psi$
2. Determinise $A_\varphi$
3. Solve a game on $A_\varphi^{\text{det}}$

Crucial elements:
- Use of cost functions and cost automata (Colcombet et al.)
- New result on history-determinisation of a subclass of cost automata on infinite words
Main result
The assume-guarantee synthesis problem for PROMPT-LTL is 2EXPTIME-complete.

Approach:
Similar to classical automata-based approach to LTL synthesis:
1. Build automata $A_\varphi$ and $A_\psi$ for $\varphi$ and $\psi$
2. History-Determinise $A_\varphi$ and $A_\psi$
3. Solve a game on $A_{\varphi}^{\text{det}}$

Crucial elements:
- Use of cost functions and cost automata (Colcombet et al.)
- New result on history-determinisation of a subclass of cost automata on infinite words
Main result
The assume-guarantee synthesis problem for PROMPT-LTL is 2EXPTIME-complete.

Approach:
Similar to classical automata-based approach to LTL synthesis:
1. Build automata $A_\varphi$ and $A_\psi$ for $\varphi$ and $\psi$
2. History-Determinise $A_\varphi$ and $A_\psi$
3. Solve a game on $A^\text{det}_\varphi$ and $A^\text{det}_\psi$

Crucial elements:
- Use of cost functions and cost automata (Colcombet et al.)
- New result on history-determinisation of a subclass of cost automata on infinite words