Assume-Guarantee Synthesis for Prompt Linear Temporal Logic

Bastien Maubert[1]

joint work with
Nathanaël Fijalkow[2], Aniello Murano[1] & Moshe Vardi[3]

[1] University of Naples
[2] CNRS & LaBRI, Bordeaux
[3] Rice University, Houston

UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

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

$$i_0$$

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

$$\begin{array}{ll} i_0 & i_1 \\ o_0 & \end{array}$$

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

$$\begin{array}{cc} i_0 & i_1 \\ o_0 & o_1 \end{array}$$

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

$$\begin{array}{ccc} i_0 & i_1 & i_2 \\ o_0 & o_1 & \end{array}$$

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

$$\begin{array}{ccc} i_0 & i_1 & i_2 \\ o_0 & o_1 & o_2 \end{array}$$

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

$$\begin{array}{cccc} i_0 & i_1 & i_2 & \dots \\ o_0 & o_1 & o_2 & \dots \end{array}$$

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

$$\begin{array}{cccc} i_0 & i_1 & i_2 & \dots \\ o_0 & o_1 & o_2 & \dots \end{array}$$

LTL synthesis problem

Given a specification $\varphi \in$ LTL over $I \cup O$, synthesize a system $S : I^* \to O$ such that all resulting executions satisfy φ.

Prompt LTL

LTL + operator of *bounded eventuality*

Syntax:

$$\varphi ::= p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U} \varphi \mid \varphi \mathbf{R} \varphi \mid \mathbf{F}^{\leq B} \varphi$$

Semantics:

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

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

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

Example

Bound the waiting time of a printer server

$$\mathbf{G}(\text{request} \Rightarrow \mathbf{F}\text{granted})$$

Prompt LTL

LTL + operator of *bounded eventuality*

Syntax:

$$\varphi ::= p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U} \varphi \mid \varphi \mathbf{R} \varphi \mid \mathbf{F}^{\leq B} \varphi$$

Semantics:

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

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

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

Example

Bound the waiting time of a printer server

$$\mathbf{G}(\text{request} \Rightarrow \mathbf{F}^{\leq B} \text{granted})$$

Assume-guarantee synthesis for Prompt-LTL

Input

- Assumption: PROMPT-LTL formula φ over I
- Guarantee: PROMPT-LTL formula ψ 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 $S : I^* \to O$ such that for all $w \in I^\omega$,

$$w, b \models \varphi \implies S(w), b' \models \psi \ ?$$

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 $\mathcal{A}_\varphi$ for the specification φ
2. Determinise $\mathcal{A}_\varphi$
3. Solve a game on $\mathcal{A}_\varphi^{\text{det}}$

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

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 $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$ for φ and ψ
2. Determinise $\mathcal{A}_\varphi$
3. Solve a game on $\mathcal{A}_\varphi^{\text{det}}$

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

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 $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$ for φ and ψ
2. History-Determinise $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$
3. Solve a game on $\mathcal{A}_\varphi^{\text{det}}$

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

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 $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$ for φ and ψ
2. History-Determinise $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$
3. Solve a game on $\mathcal{A}_\varphi^{\text{det}}$ and $\mathcal{A}_\psi^{\text{det}}$

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