CTL*-via-LTL Synthesis

Roderick Bloem, Ayrat Khalimov, TU Graz
Sven Schewe, University of Liverpool

Intro & Motivation

CTL* Synthesis Problem

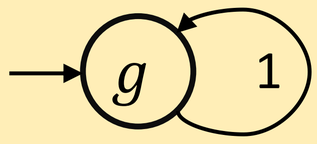
Input: [CTL* or LTL] formula φ, inputs I, outputs O

Output: I/O machine satisfying φ or "unrealisable"

CTL* allows the designer to write *structural* properties, but LTL synthesizers are prevalent. Hence we want to turn state-of-the-art LTL synthesizers into CTL* synthesizers.

LTL

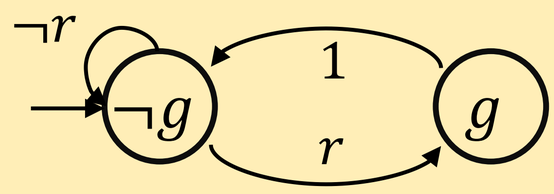
$\mathbf{G}(r \rightarrow \mathbf{F}g)$



this system is boring

CTL*

$\mathbf{AG}(r \rightarrow \mathbf{F}g) \wedge \mathbf{AGEFG}\neg g$



this system is more interesting

Reducing to LTL Synthesis

Idea

We will synthesize *explicit* models:

- for each sub-formula $\mathbf{A}\varphi$ or $\mathbf{E}\varphi$, introduce *new* Boolean outputs $p_{A\varphi}$ or $p_{E\varphi}$
- for each $\mathbf{E}\varphi$, introduce direction-output $d_{E\varphi} \in 2^I$ that encodes path that satisfies φ

LTL formula says:

a) The top-level proposition holds in the initial state

b) $\mathbf{G}(\, p_{A\varphi} \rightarrow \varphi \,)$

c) "$\mathbf{G}(\, p_{E\varphi} \rightarrow (\mathbf{G}d_{E\varphi} \rightarrow \varphi)\,)$" (roughly*)

*roughly, because one direction-output per sub-formula might be not enough.

Correct reduction

For each $\mathbf{E}\varphi$, add outputs $d_1, \dots, d_{|Q|}, v: \{0 \dots |Q|\}$, where Q are the states of an NBW for φ.

Use (a), (b), but replace (c) with:

$$\bigwedge_{i \in \{1 \dots |Q|\}} \mathbf{G}[\; v_{E\varphi} = i \;\rightarrow\; (\mathbf{G}d_i \rightarrow \varphi)\;]$$

$|Q|$ number of direction-outputs suffice, because the (memory-less) verifier can pass through a tree node in up to $|Q|$ different automaton states.

CTL* specification

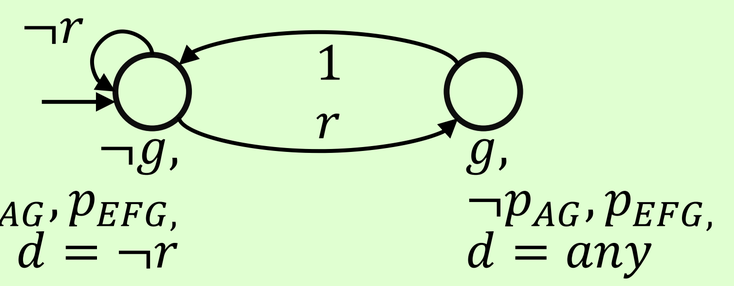
$I = \{r\}, O = \{g\}, \varphi = \mathbf{AGEFG}\neg g$

is translated into LTL specification:

$I = \{r\}, O = \{g, p_{AG}, p_{EFG}, d_{EFG}\},$

$\varphi = p_{AG} \wedge \mathbf{G}(p_{AG} \rightarrow \mathbf{G}p_{EFG}) \wedge$

$\mathbf{G}(p_{EFG} \wedge \mathbf{G}d_{EFG} \rightarrow \mathbf{FG}\neg g)$



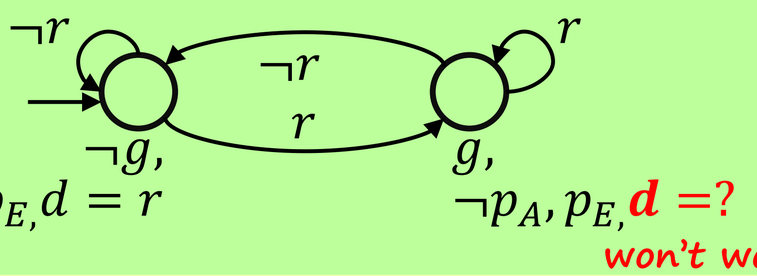
$I = \{r\}, O = \{g\}, \mathbf{AG}\,\mathbf{EX}(g \wedge \mathbf{F}\neg g)$

becomes

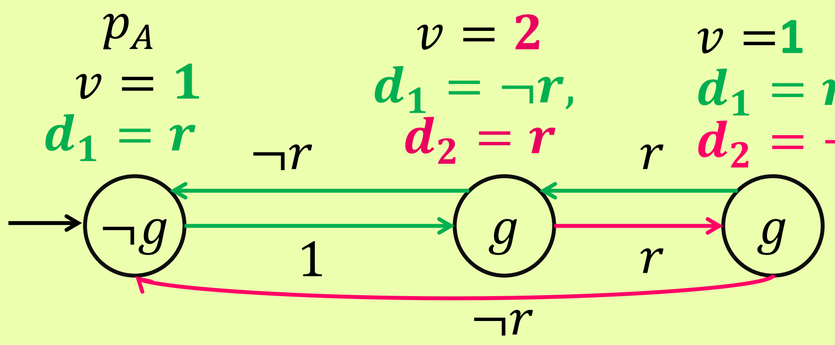
$I = \{r\}, O = \{g, p_A, p_E, d_E\}$

$p_A \wedge \mathbf{G}(p_A \rightarrow \mathbf{G}p_E) \wedge$

$\mathbf{G}(p_E \wedge \mathbf{G}d_E \rightarrow \mathbf{X}(g \wedge \mathbf{F}\neg g))$



won't work



Properties of the Reduction

- Φ_{LTL} is realizable $\Leftrightarrow$ Φ_{CTL^*} is realizable
- $|\Phi_{LTL}| = EXP(|\Phi_{CTL*}|)$
- … but the complexity stays in 2EXPTIME
- Systems can get larger
- Experiments: fast when the number of $\mathbf{E}$ sub-formulas is small

Why can we reduce CTL* synthesis to LTL synthesis, but cannot reduce CTL* MC to LTL MC?

We introduce *new* outputs and they are used by the LTL formula. The synthesizer has to generate them, but MC is not aware of them.