Two Synthesis Approaches for CTL*

Roderick Bloem[1], Ayrat Khalimov[1], Sven Schewe[2]

TU Graz[1] Graz University of Technology

RiSE[1] Rigorous Systems Engineering

University of Liverpool[2]

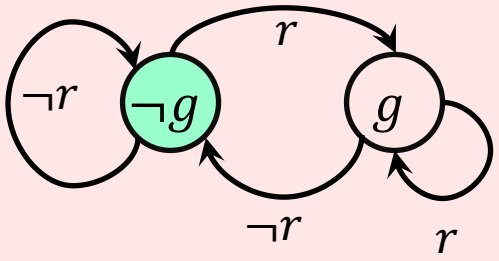
LTL/CTL* synthesis problem

Specification:

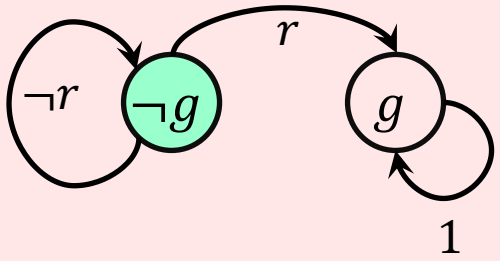
- LTL formula: $\boldsymbol{G}(r \rightarrow \boldsymbol{F}\ g)$
- Inputs: r, outputs: g

Find a state machine with such inputs/outputs that satisfies the formula.

An example solution



Another solution



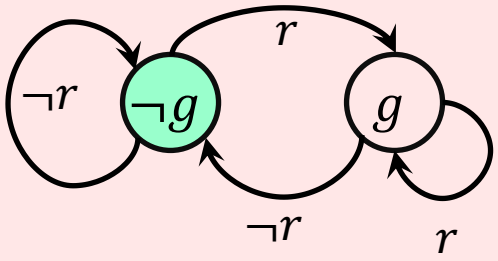
LTL/CTL* synthesis problem

Specification:

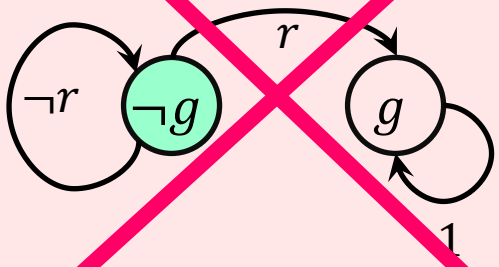
- CTL* formula: $\boldsymbol{AG}(r \rightarrow \boldsymbol{F}\ g) \wedge \boldsymbol{AGEF}\neg g$
- Inputs: r, outputs: g

Find a state machine with such inputs/outputs that satisfies the formula.

An example solution



Another solution



Talk outline

- SMT-based bounded CTL* synthesis
  - "model checking, but with unknown system functions" (bounded synthesis)
- Reducing CTL* synthesis to LTL synthesis
  - explicit models
- Conclusion

CTL* synthesis: approach #1

bottom-up CTL* model checking with uninterpreted functions

- Encode CTL* *model checking* into SMT
  - the query is satisfiable iff the system is correct
- Replace the known system with UFs
  - possible if we bound the number of system states

Encoding CTL^* model checking into SMT

$$system \models \overbrace{\mathbf{A}G\,\underbrace{\mathbf{E}FG\,\overline{g}}_{p_E}}^{p_A}$$

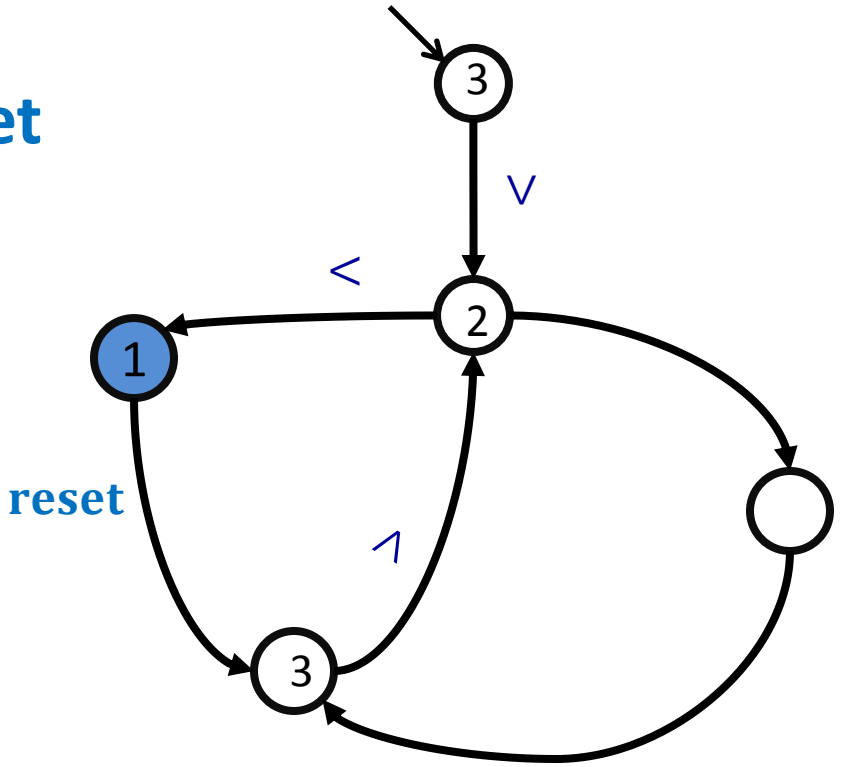
- Proposition for each sub-formula
- For every s and sub-formula ϕ, encode into SMT "$p_\phi(s) \rightarrow s \models \phi$"
  - $p_A(s) \rightarrow s \models AGp_E$
  - $p_E(s) \rightarrow s \models EFG\,\overline{g}$

How to encode into SMT?

- Require $p_{top}(s_0) = true$
  - $p_A(s_0) = true$

Encode $s \models E\varphi$ into SMT

- Build the product graph $system \times automaton_{\varphi}$
  - Buchi automaton
- $s \models E\varphi \quad \Leftrightarrow \quad$ the product has an accepting path
- Buchi ranking
  - exit normal state: $<$
  - exit accepting state: **reset**
- SMT query is satisfiable
  iff the product is accepting



From model checking to synthesis

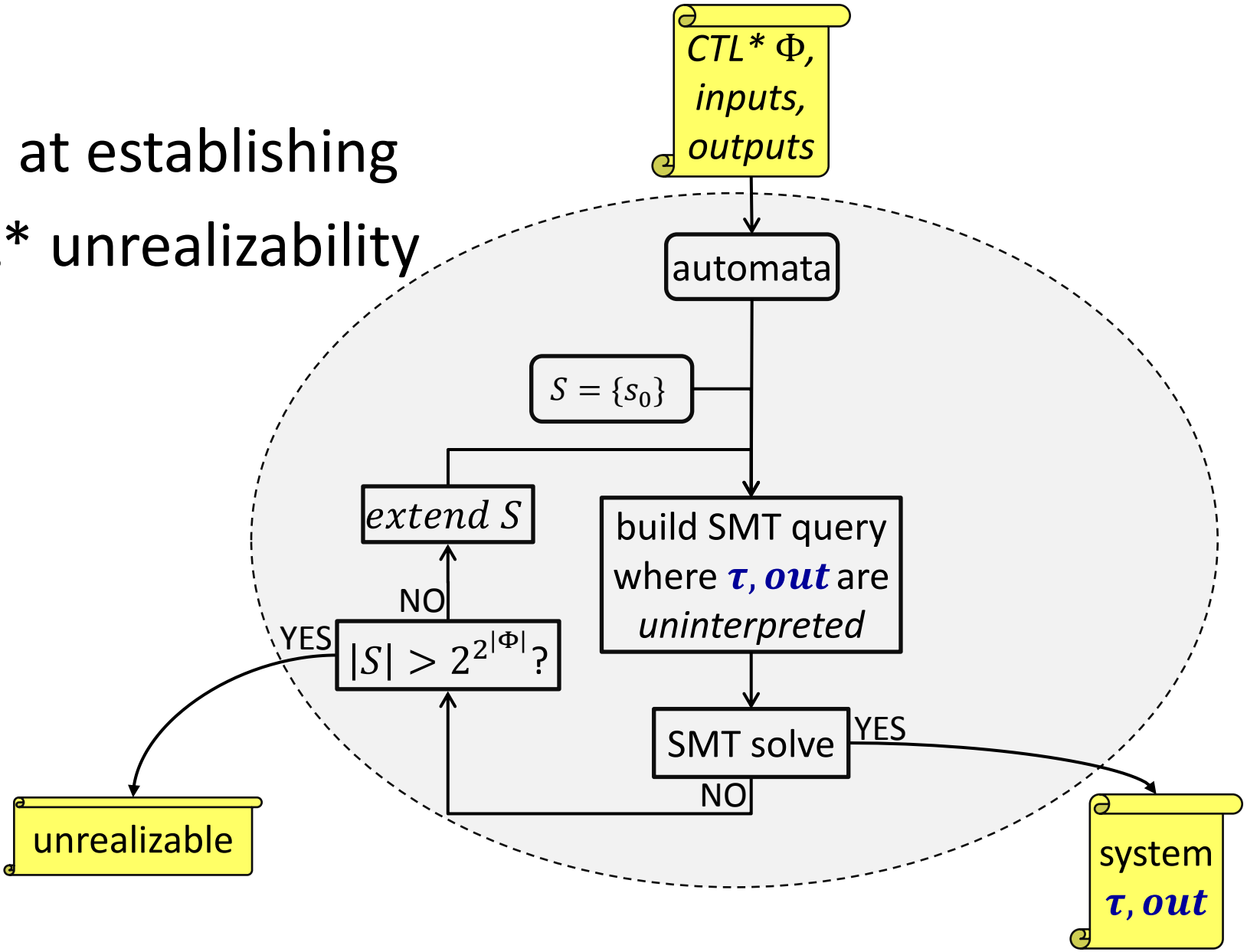
- SMT constraints look like this:

$$\bigwedge_{s \in S,\ r \in B} rch(q,s) \wedge \boldsymbol{grant}(s) \rightarrow rch\big(q, \boldsymbol{\tau}(s,r)\big) \wedge \rho(q,s) > \rho(q', \boldsymbol{\tau}(s,r))$$

- To do synthesis, replace given system functions ($\boldsymbol{grant}$ and $\boldsymbol{\tau}$) with uninterpreted functions!

CTL* bounded synthesis: summary

- bad at establishing CTL* unrealizability



CTL* synthesis: approach #2

reduce CTL* synthesis to LTL synthesis

- Overcome the bounded synthesis limitation
  - efficiently handle unrealizable CTL*
- \+ Avoid building specialized CTL* synthesizers
- \+- Be fast by using state-of-the-art LTL synthesizers

Idea of reduction CTL* -> LTL

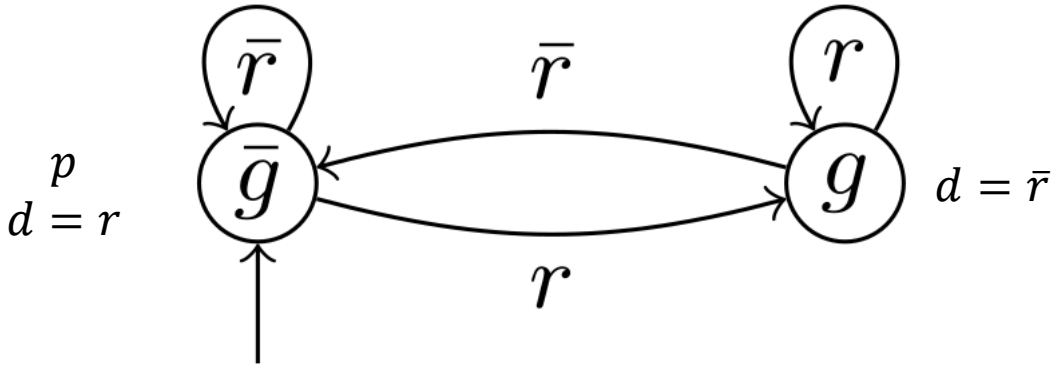
- Synthesize *explicit* models
  - for each sub-formula $A\varphi$ or $E\varphi$, introduce new system outputs $p_{A\varphi}$ or $p_{E\varphi}$
  - for each $E\varphi$, introduce direction-output $d_{E\varphi}$ that encodes system path that satisfies φ
- LTL formula says:
  - $\mathbf{G}\left(p_{A\varphi} \rightarrow \varphi \right)$
  - "$\mathbf{G}\left(p_{E\varphi} \rightarrow \left(\mathbf{G} d_{E\varphi} \rightarrow \varphi \right) \right)$" (roughly)
  - The top-level proposition holds initially

Example

- The top-level proposition holds initially
- $\mathbf{G}(\, p_{A\varphi} \rightarrow \varphi \,)$
- "$\mathbf{G}(\, p_{E\varphi} \rightarrow (\mathbf{G} d_{E\varphi} \rightarrow \varphi) \,)$" (roughly)

- $\boldsymbol{\Phi_{\mathbf{CTL}*} = \mathbf{EX}(g \wedge \mathbf{F}\bar{g})}$, inputs={r}, outputs={g}
- inputs={r}, outputs=$\{g, p, d\}$

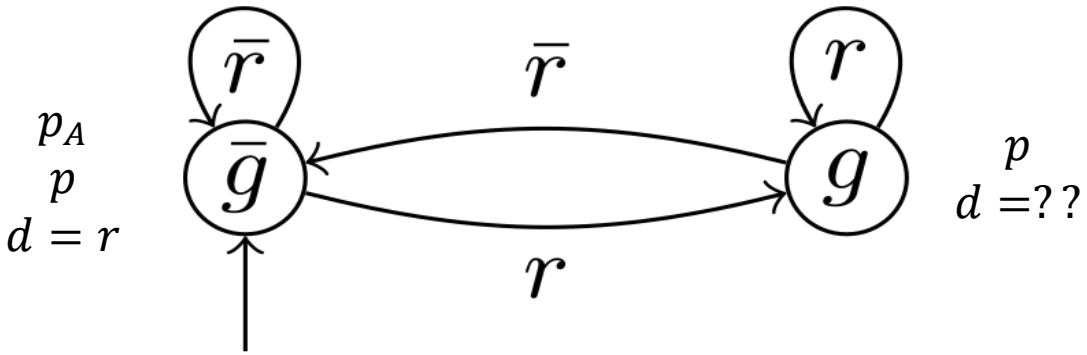
$$\boldsymbol{\Phi_{\mathbf{LTL}} = p \wedge G(p \rightarrow Gd \rightarrow X(g \wedge F\bar{g}))}$$



Counterexample to 'rough' Eϕ reduction

- $\boldsymbol{\Phi_{CTL*} = \mathbf{AG}\,\mathbf{EX}(g \wedge F\overline{g})}$
- outputs=$\{g, p_A, p, d\}$

$$\boldsymbol{\Phi_{LTL} = p_A \wedge G(p_A \rightarrow Gp) \wedge G(p \rightarrow Gd \rightarrow X(g \wedge F\overline{g}))}$$



Correct translation of E-formulas

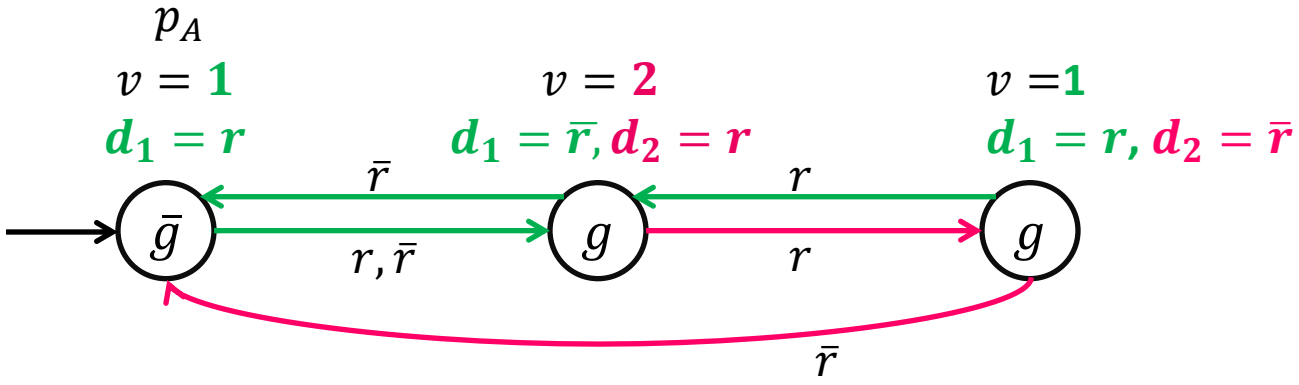
- For each $E\varphi$, add outputs $d_1, \ldots, d_{|Q|}$, $v: \{0 \ldots |Q|\}$
- Add LTL formula:

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

Example

- $\boldsymbol{\Phi_{CTL*} = AGEX(g \wedge F\bar{g})}$
- outputs=$\{g, p_A, v{:}\{0 \ldots 4\}, d_1, d_2, d_3, d_4\}$

$$\boldsymbol{\Phi_{LTL} = p_A \wedge G(p_A \rightarrow Gv \neq 0) \wedge \bigwedge_{i \in \{1 \ldots 4\}} G(v = i \rightarrow Gd_i \rightarrow X(g \wedge F\bar{g}))}$$



CTL* via LTL synthesis: summary

- For each sub-formula $E\varphi$:

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

- For each sub-formula $A\varphi$:

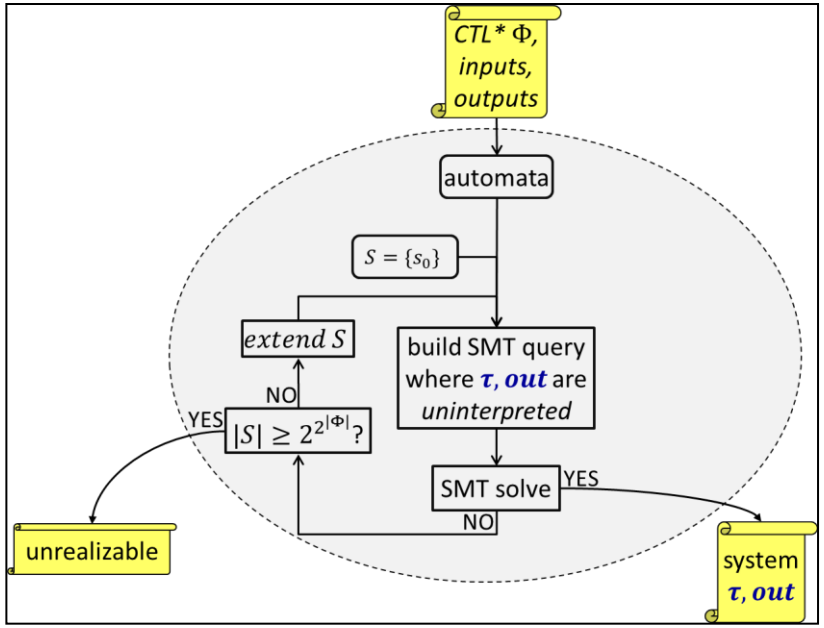
$$\mathbf{G}[\ p_{A\varphi} \ \rightarrow \ \varphi'\ ] \qquad (2)$$

- The LTL formula is

$$\bigwedge_{\mathbf{E}\varphi} Eq.1 \ \wedge \bigwedge_{A\varphi} Eq.2 \wedge \Phi'$$

- Φ_{LTL} is realizable $\Leftrightarrow$ Φ_{CTL^*} is realizable
- $|\Phi_{LTL}| \approx 2^{|\Phi_{CTL*}|}$
- Yet the synthesis complexity stays in 2EXPTIME
- Systems can get larger
- Experiments: faster when the # of E-formulas is small

Conclusion



SMT-based bounded CTL* synthesis

- For each sub-formula $E\varphi$:

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

- For each sub-formula $A\varphi$:

$$\mathbf{G}[\ p_{A\varphi} \ \rightarrow \ \varphi'\ ] \qquad (2)$$

- The LTL formula is

$$\bigwedge_{\mathbf{E}\varphi} Eq.1 \ \wedge \bigwedge_{A\varphi} Eq.2 \wedge \Phi'$$

CTL*-via-LTL synthesis

Future directions:

- How to establish unrealizability of CTL*?
- Synthesizers for ATL*
- Satisfiability of CTL*