CTL* synthesis via LTL synthesis

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

TU Graz[1] Graz University of Technology
RiSE[1] Rigorous Systems Engineering
University of Liverpool[2]

in the next 30 minutes

- LTL/CTL* synthesis problem
- Why reduce CTL* synthesis to LTL synthesis?
  - unrealizable specifications
- Reduction
  - annotating trees with strategies
- Conclusion

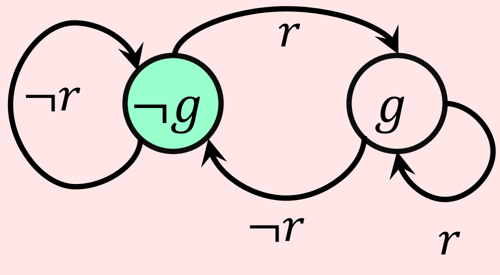
LTL/CTL* synthesis problem by example

Specification:

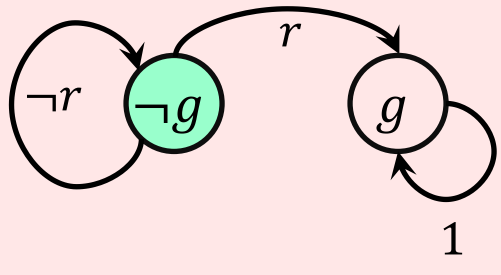
- LTL formula: $\boldsymbol{G}(r \rightarrow \boldsymbol{F}\, g)$
- Inputs: r, outputs: g

Find a state machine with such inputs/outputs whose all executions satisfy the formula.

An example solution



Another solution

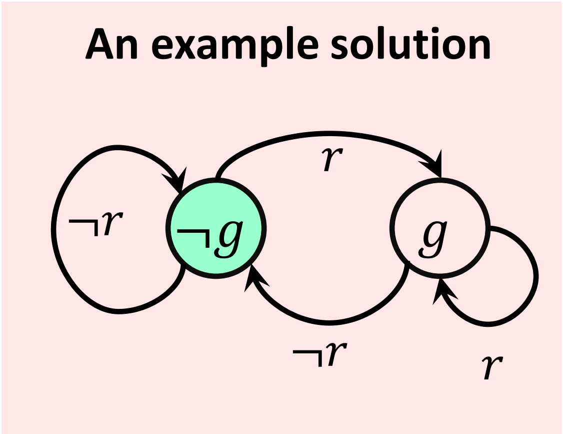


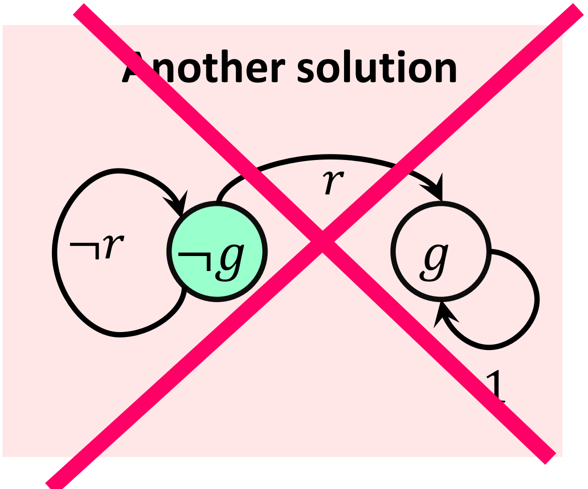
LTL/CTL* synthesis problem by example

Specification:

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

Find a state machine with such inputs/outputs whose all executions satisfy the formula.





why reduce CTL* synth. to LTL synthesis?

1. Handle unrealizable CTL* *efficiently*
2. Avoid building specialized CTL* synthesizers
   - re-use state-of-the-art LTL synthesizers

unrealizable specifications: LTL

$[\Phi_{LTL}, I, O, type]$ is unrealizable $\Leftrightarrow$
$[\neg\Phi_{LTL}, O, I, \neg type]$ is realizable

Example:

- $g \leftrightarrow \mathbf{X}r,\ I = \{r\}, O = \{g\}$ is *un*realizable.
- $\neg(g \leftrightarrow \mathbf{X}r),\ I = \{g\}, O = \{r\}$ is realizable:

output the negated first value of g.

unrealizable specifications: CTL*

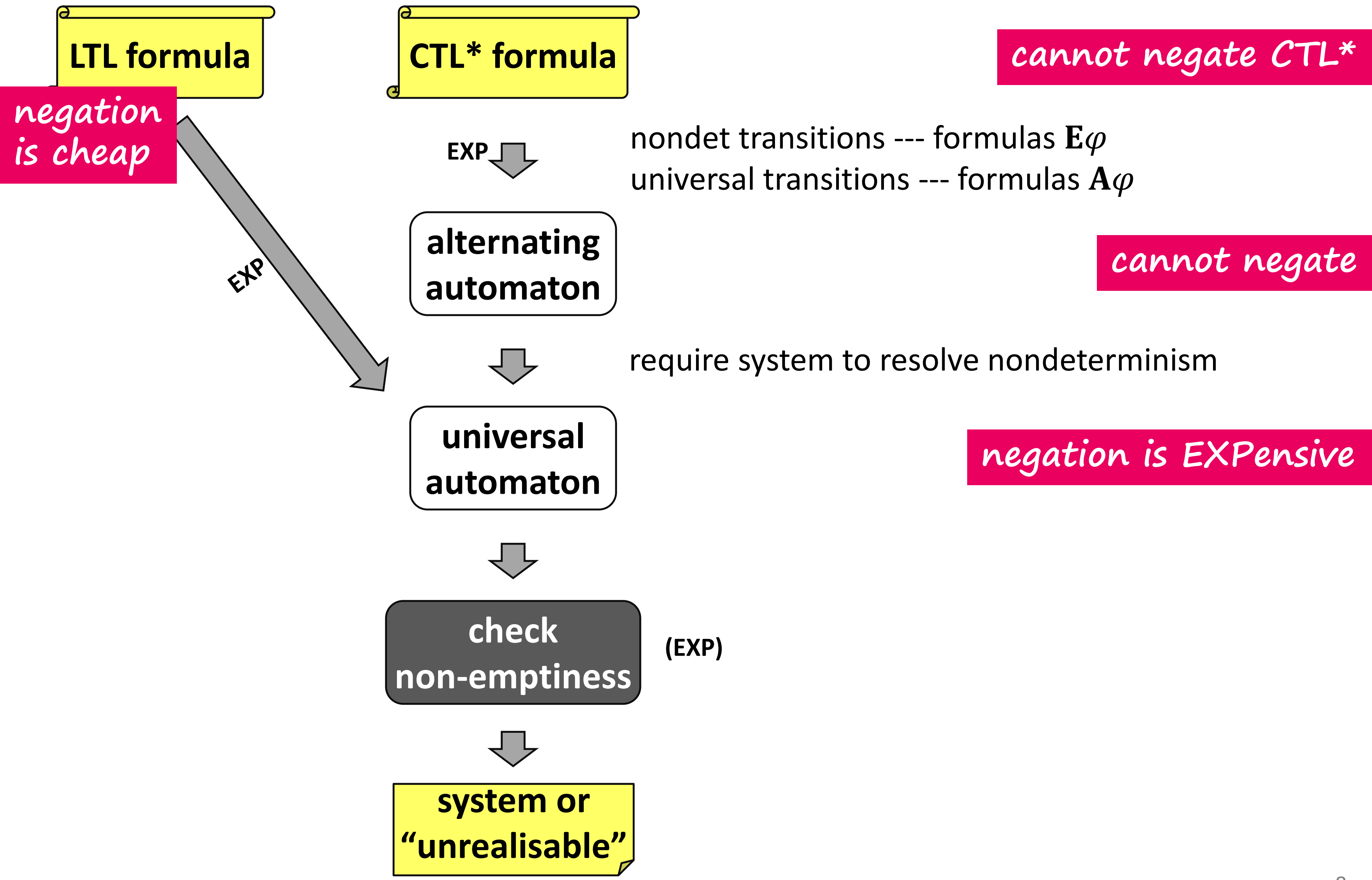
$[\Phi_{CTL^*}, I, O, type]$ is unrealizable $\Leftrightarrow$
$[\neg\Phi_{CTL^*}, O, I, \neg type]$ is realizable

wrong!

Counterexample:

- $\mathbf{AG}o, I = \{i\}, O = \{o\}$ is realizable:
  always output o.
- $\mathbf{EF}\neg o, I = \{o\}, O = \{i\}$ is realizable:

steps in standard LTL/CTL* synthesis

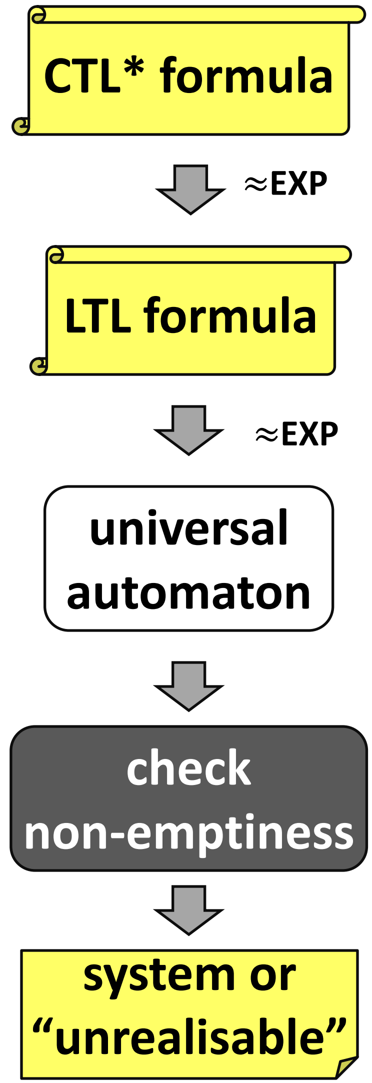


our reduction

Φ_{CTL^*} **is realizable** $\Leftrightarrow$ Φ_{LTL} **is realizable**

the *total* blow-up is as before: EXP

system size can grow



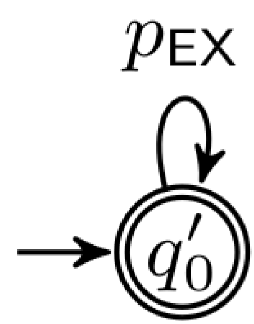
require system to resolve nondeterminism

(EXP)

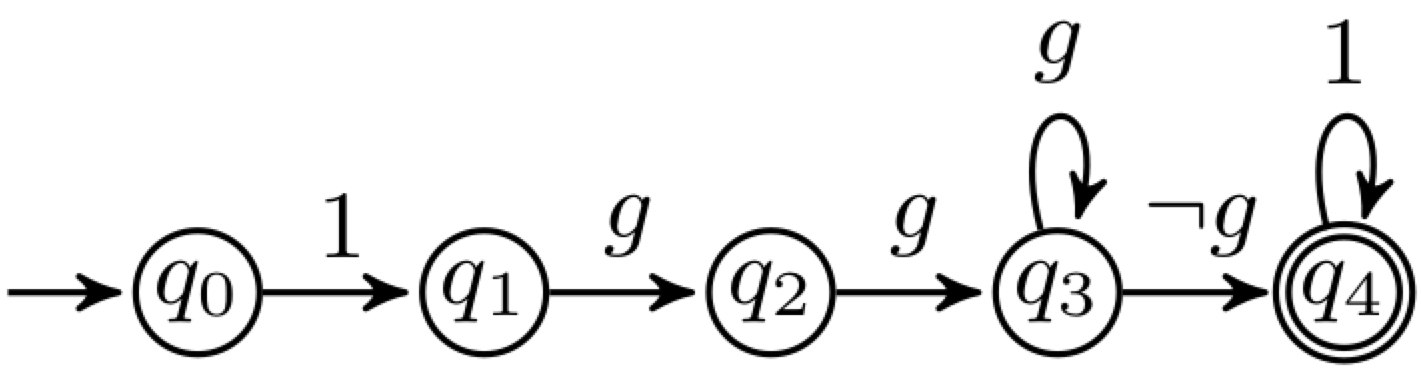
negation is cheap

automata for CTL*

- $\mathbf{EG}\,\mathbf{EX}\big(g\ \wedge\ \mathbf{X}(g\ \wedge\ \mathbf{F}\,\neg g)\big)$
- $p_{EX} \equiv \mathbf{EX}\big(g\ \wedge\ \mathbf{X}(g\ \wedge\ \mathbf{F}\,\neg g)\big)$
- $p_{EG} \equiv \mathbf{EG}p_{EX}$

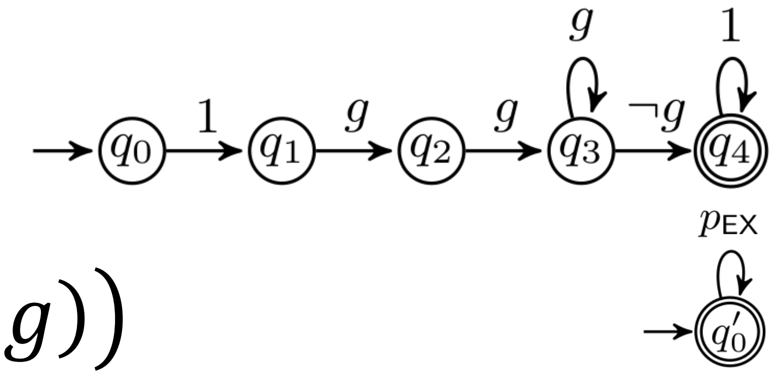


NBW for $\mathbf{G}\mathrm{p}_{\mathrm{EX}}$

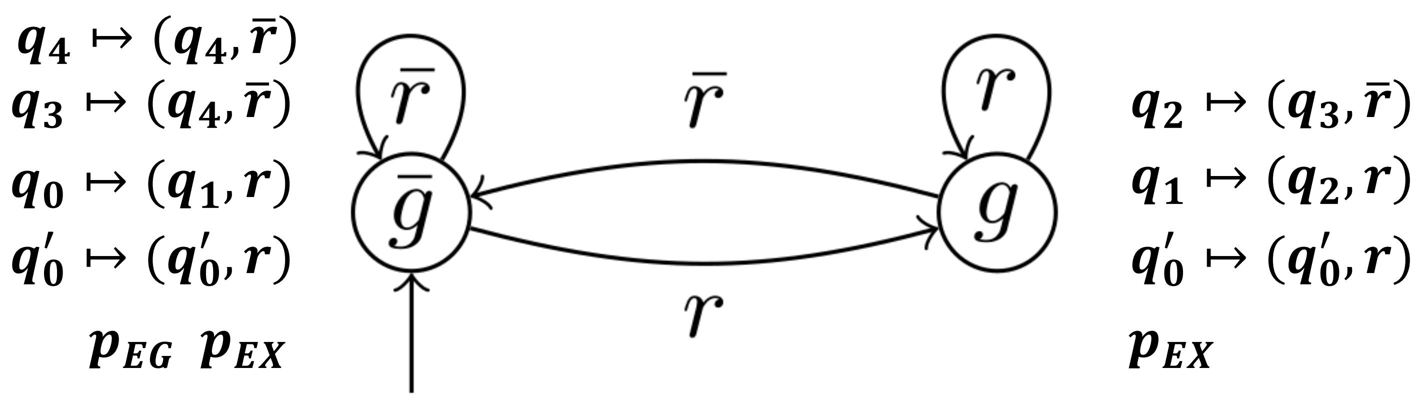


NBW for $\mathbf{X}(\boldsymbol{g} \wedge \cdots)$

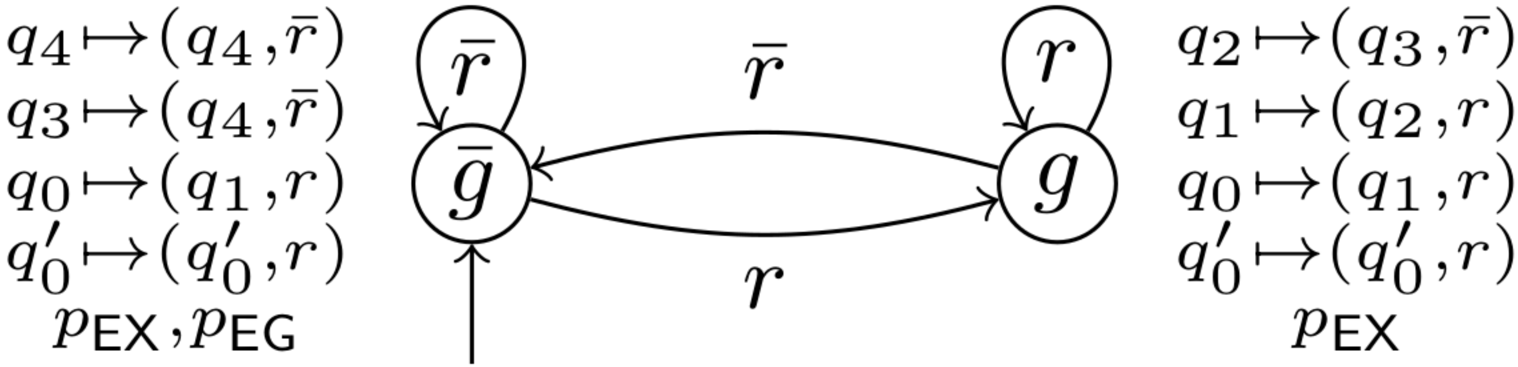
model checking $\mathbf{EG}\,\mathbf{EX}(g \wedge \mathbf{X}(g \wedge \mathbf{F}\,\neg g))$



- $p_{EX} \equiv \mathbf{EX}(g \wedge \mathbf{X}(g \wedge \mathbf{F}\,\neg g))$
- $p_{EG} \equiv \mathbf{EG} p_{EX}$



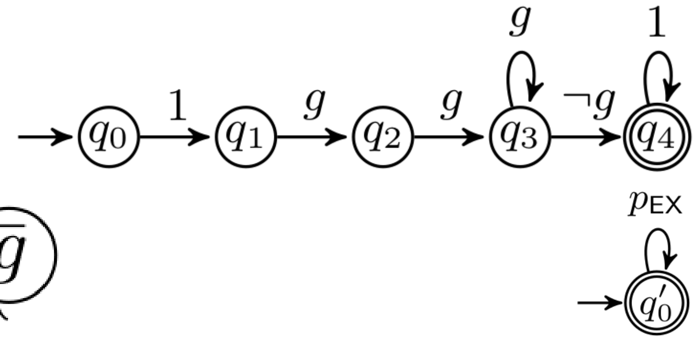
annotated model

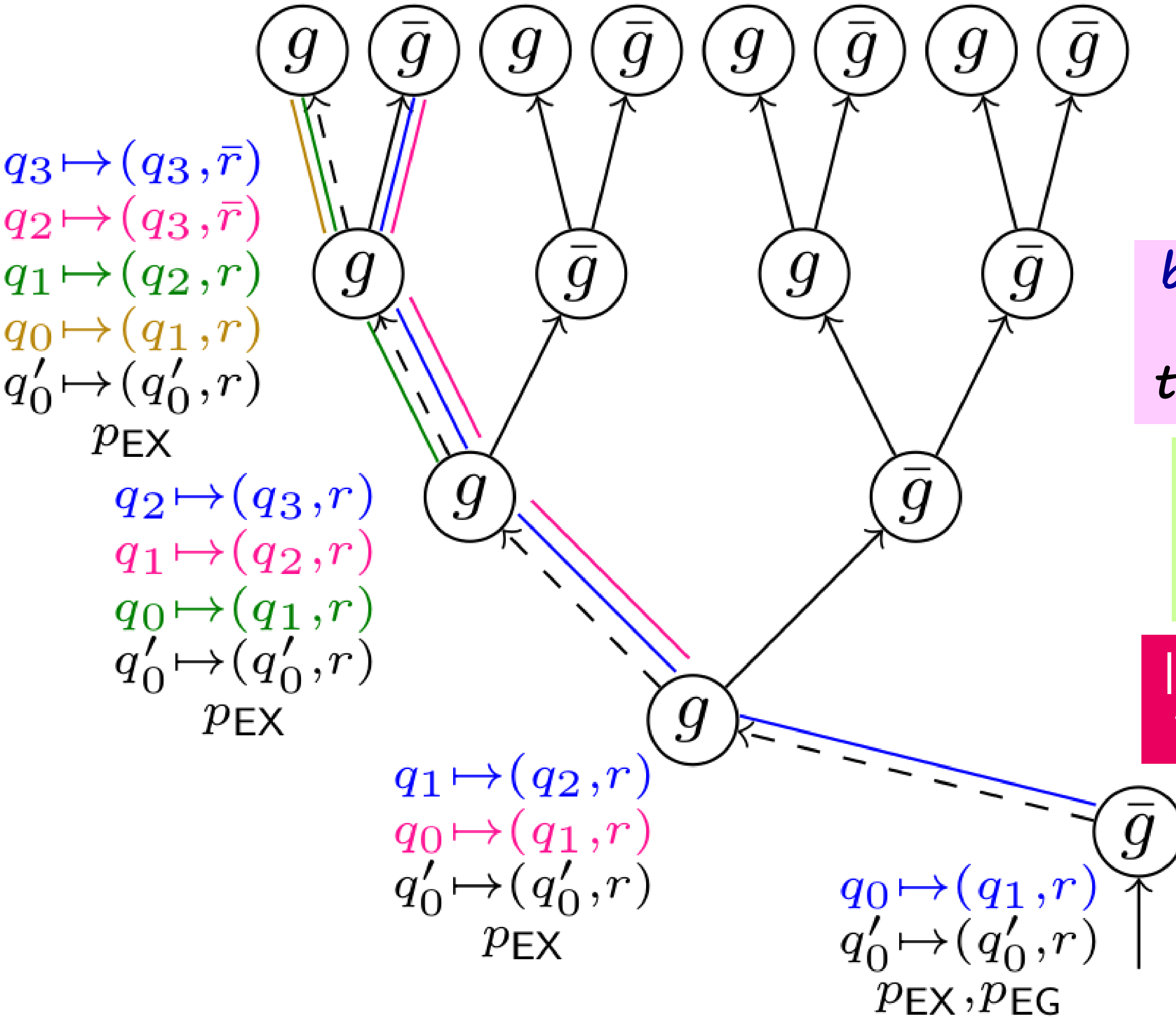


Every state is additionally labeled with:

- $subformulas \rightarrow \{true, false\}$
- $Q \rightarrow Q \times Direction$

annotated tree





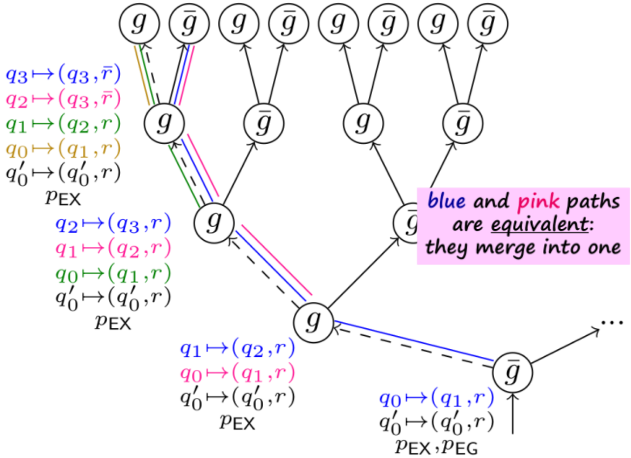
blue and pink paths are equivalent: they merge into one

how many different paths can pass a node?

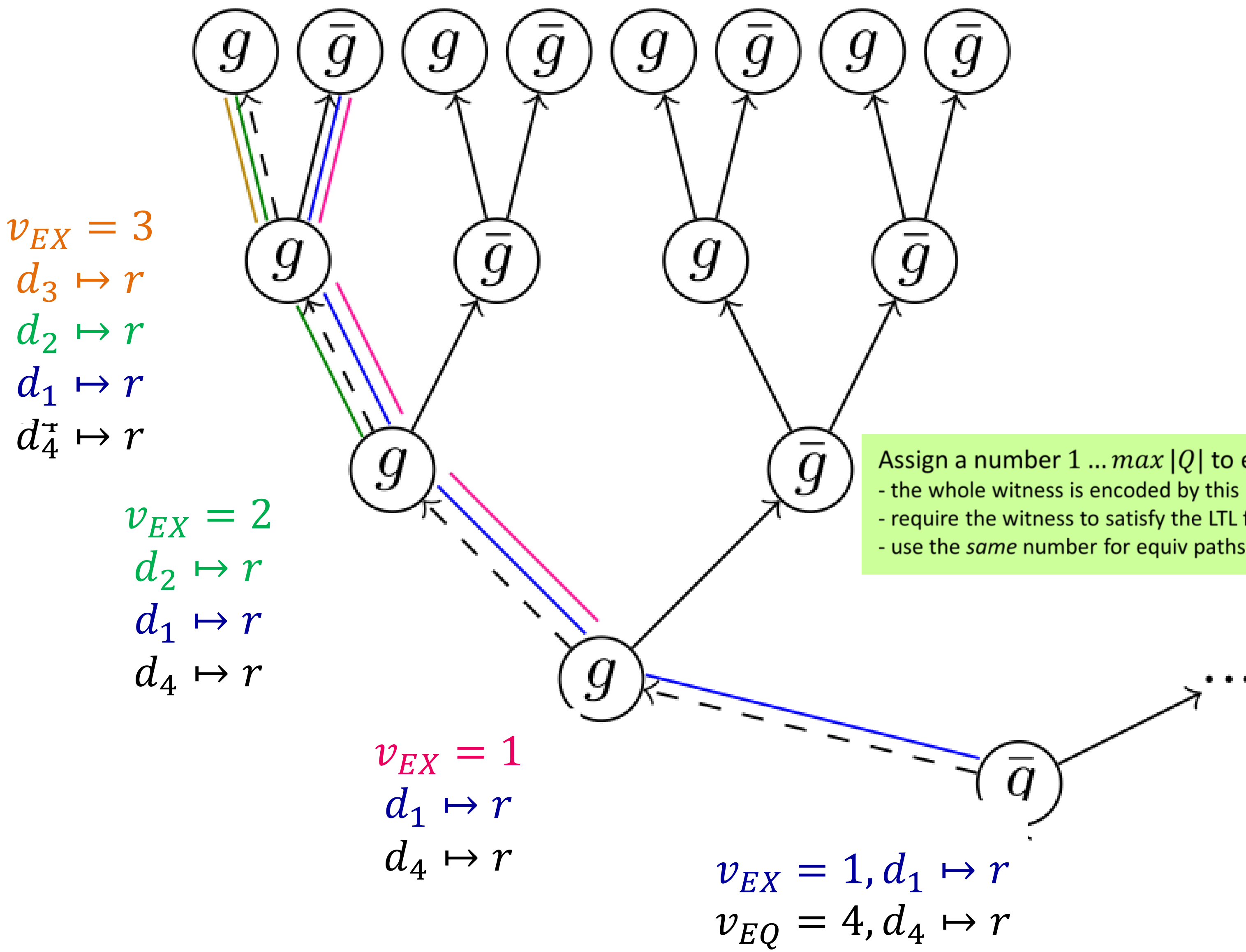
|Q|: the number of the nondet states!

core ideas of reduction

- "merging" paths are *equivalent*
  - max $|Q|$ non-equiv paths can pass through a node
- Assign a number $1 \dots max\ |Q|$ to each witness of p_{EX}
  - the whole witness is encoded by this number
  - require the witness to satisfy the LTL formula of p_{EX}
  - use the *same* number for equiv paths



newly annotated tree



Assign a number 1 … $max\,|Q|$ to each witness of p_{EX}
- the whole witness is encoded by this number
- require the witness to satisfy the LTL formula of p_{EX}
- use the *same* number for equiv paths

LTL formula

- For each subformula $E\varphi$:

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

- For each subformula $A\varphi$:

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

- The LTL formula is

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

our result

- For each subformula $E\varphi$:

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

- For each subformula $A\varphi$:

$$\mathbf{G}[\ p_{A\varphi} \ \rightarrow \ \varphi'\ ]$$

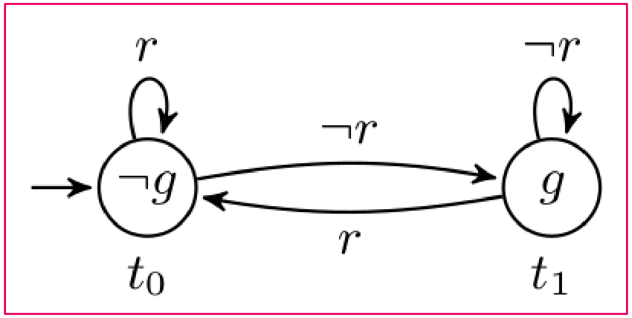
- The LTL formula is

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

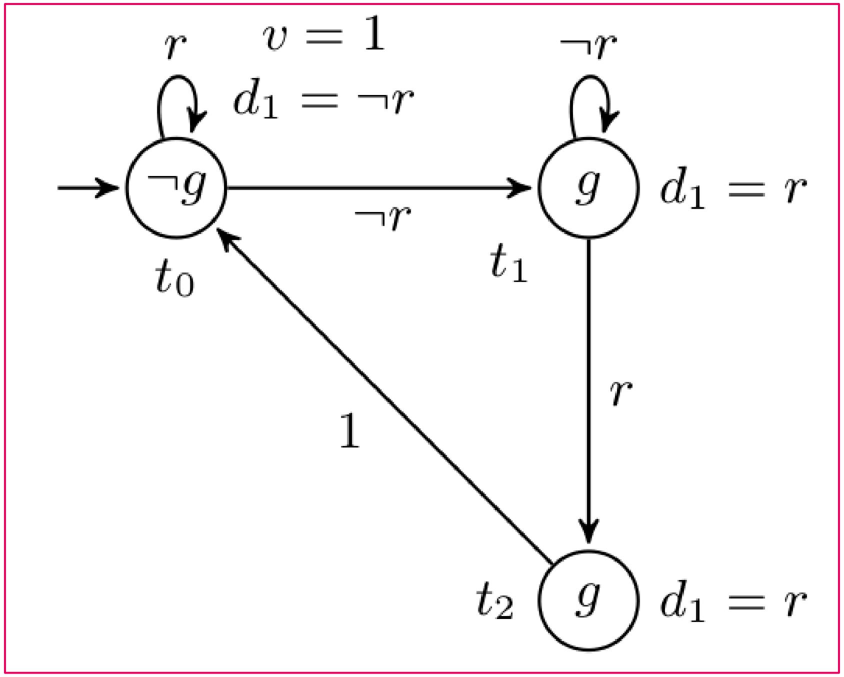
- Φ_{LTL} is realizable $\Leftrightarrow$ Φ_{CTL^*} is realizable
- The complexity stays in 2EXP
- The system can get larger!

example: $\mathbf{EX}(g \wedge \mathbf{X}(g \wedge \mathbf{X}\neg g))$

$$v \neq 0 \wedge \bigwedge_{i \in \{1,\ldots,5\}} \mathbf{G}[\, v = i \;\rightarrow\; (\; \mathbf{G} d_i \;\rightarrow \mathbf{X}(g \wedge \mathbf{X}(g \wedge \mathbf{X}\neg g\,))\,)\,]$$



a smallest system satisfying Φ_{CTL*}



a smallest system satisfying Φ_{LTL}

conclusion

We reduced CTL* synthesis to LTL synthesis without incurring a blow up.

Now we can use the reduction to handle unrealizable CTL* specifications and to re-use LTL synthesizers.