Circuit Transformer: A Transformer That Preserves Logical Equivalence

Xihan Li[1], Xing Li[2], Lei Chen[2], Xing Zhang[2], Mingxuan Yuan[2] and Jun Wang[1]

1. Centre for Artificial Intelligence, University College London
2. Huawei Noah's Ark Lab, Hong Kong, China

Problem Setting & Challenges

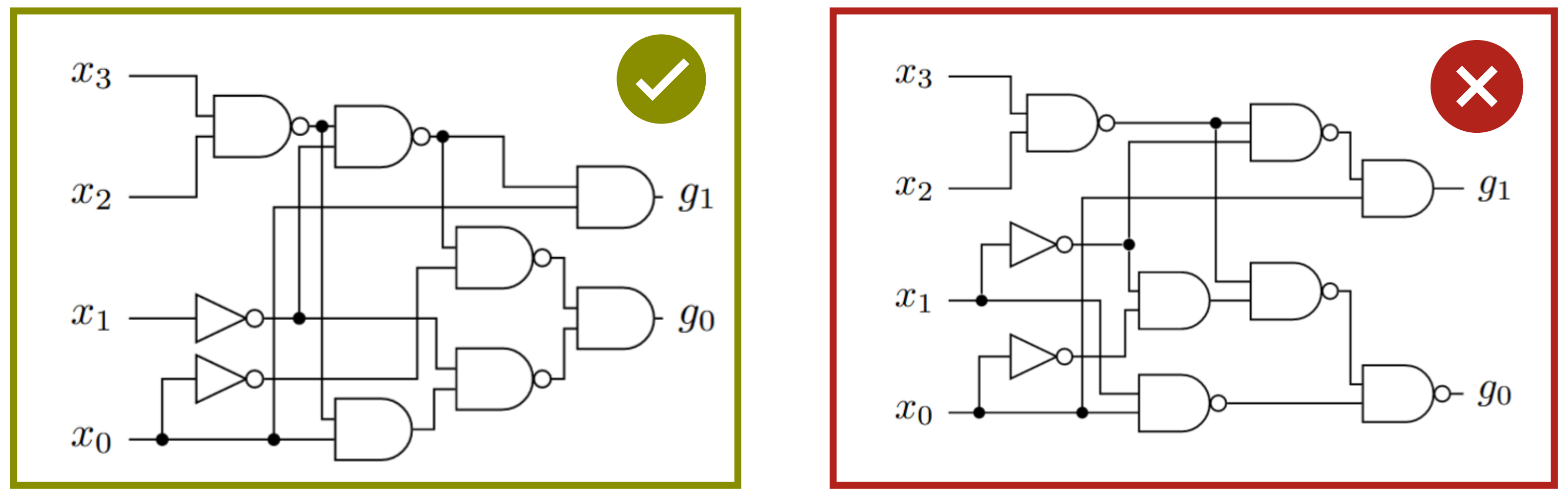
Implementing Boolean functions with logic circuits
(A fundamental problem in digital design!)

Boolean function f

x_3	x_2	x_1	x_0	y_1	y_0	x_3	x_2	x_1	x_0	y_1	y_0
0	0	0	0	0	1	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	1	0	0
0	0	1	0	0	0	1	0	1	0	0	0
0	0	1	1	1	1	1	0	1	1	1	1
0	1	0	0	0	1	1	1	0	0	0	0
0	1	0	1	0	0	1	1	0	1	1	1
0	1	1	0	0	0	1	1	1	0	0	0
0	1	1	1	1	1	1	1	1	1	1	1

Must be **exactly** equivalent
(i.e., without a single bit of error!)

Logic circuit g



Outputs match for all 16 possible inputs

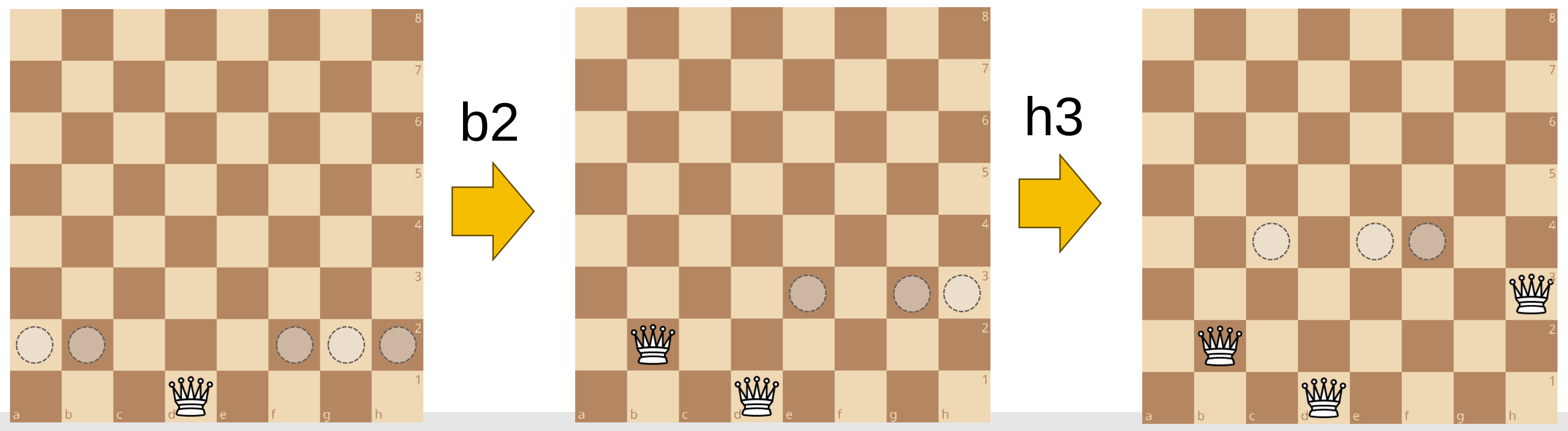
Outputs not exactly match (15 match, 1 not match)

Question: Can generative neural models directly generate strictly equivalent circuits for a Boolean function?

Typical answer is No (generative models occasionally make mistakes). Previous AI approaches strengthens traditional symbolic methods.

Motivation

Complex constraints can be satisfied step-by-step!
(e.g., the 8-queen problem)



Valid choices: $a2, b2, f2, g2, h2$

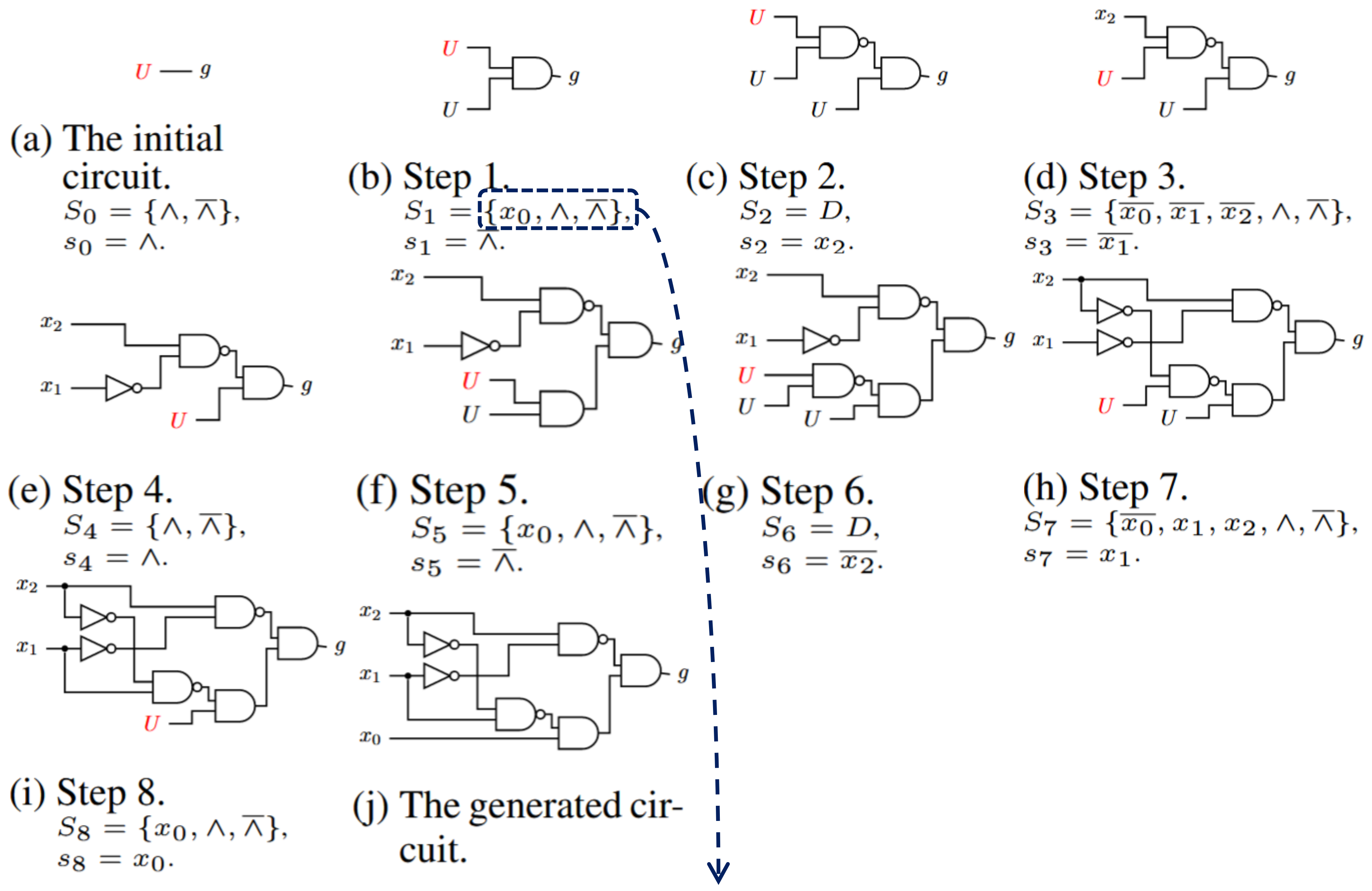
Valid choices: $e3, g3, h3$

Valid choices: $c4, e4, f4$

Our Method

Generate circuits step-by-step with equivalence preserved

- Construct from outputs to inputs, start from a wildcard node U
- In each step, replace a wildcard node U with a new gate / input
- In each step t, only choose from **the valid choices S_t**



(a) The initial circuit. $S_0 = \{\wedge, \overline{\wedge}\}$, $s_0 = \wedge$.

(b) Step 1. $S_1 = \{x_0, \wedge, \overline{\wedge}\}$, $s_1 = \overline{\wedge}$.

(c) Step 2. $S_2 = D$, $s_2 = x_2$.

(d) Step 3. $S_3 = \{\overline{x_0}, \overline{x_1}, \overline{x_2}, \wedge, \overline{\wedge}\}$, $s_3 = \overline{x_1}$.

(e) Step 4. $S_4 = \{\wedge, \overline{\wedge}\}$, $s_4 = \wedge$.

(f) Step 5. $S_5 = \{x_0, \wedge, \overline{\wedge}\}$, $s_5 = \overline{\wedge}$.

(g) Step 6. $S_6 = D$, $s_6 = \overline{x_2}$.

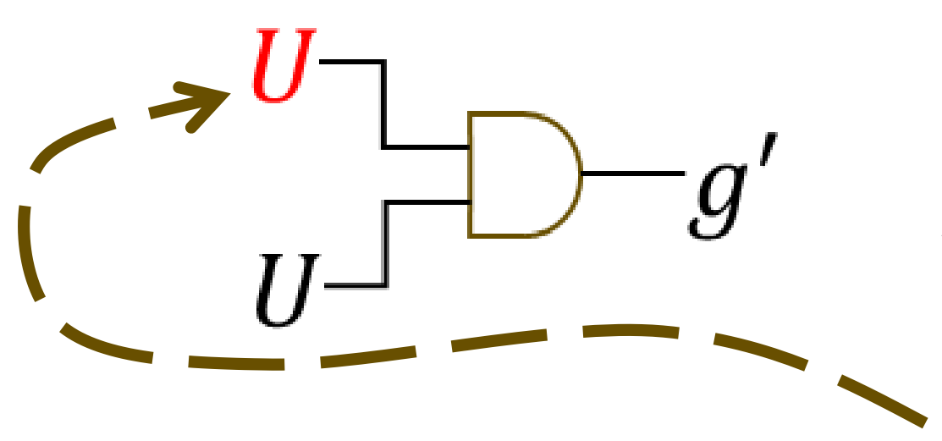
(h) Step 7. $S_7 = \{\overline{x_0}, x_1, x_2, \wedge, \overline{\wedge}\}$, $s_7 = x_1$.

(i) Step 8. $S_8 = \{x_0, \wedge, \overline{\wedge}\}$, $s_8 = x_0$.

(j) The generated circuit.

How to find valid choices in each step

Valid choices for U:
$x_0, \wedge, \overline{\wedge}$



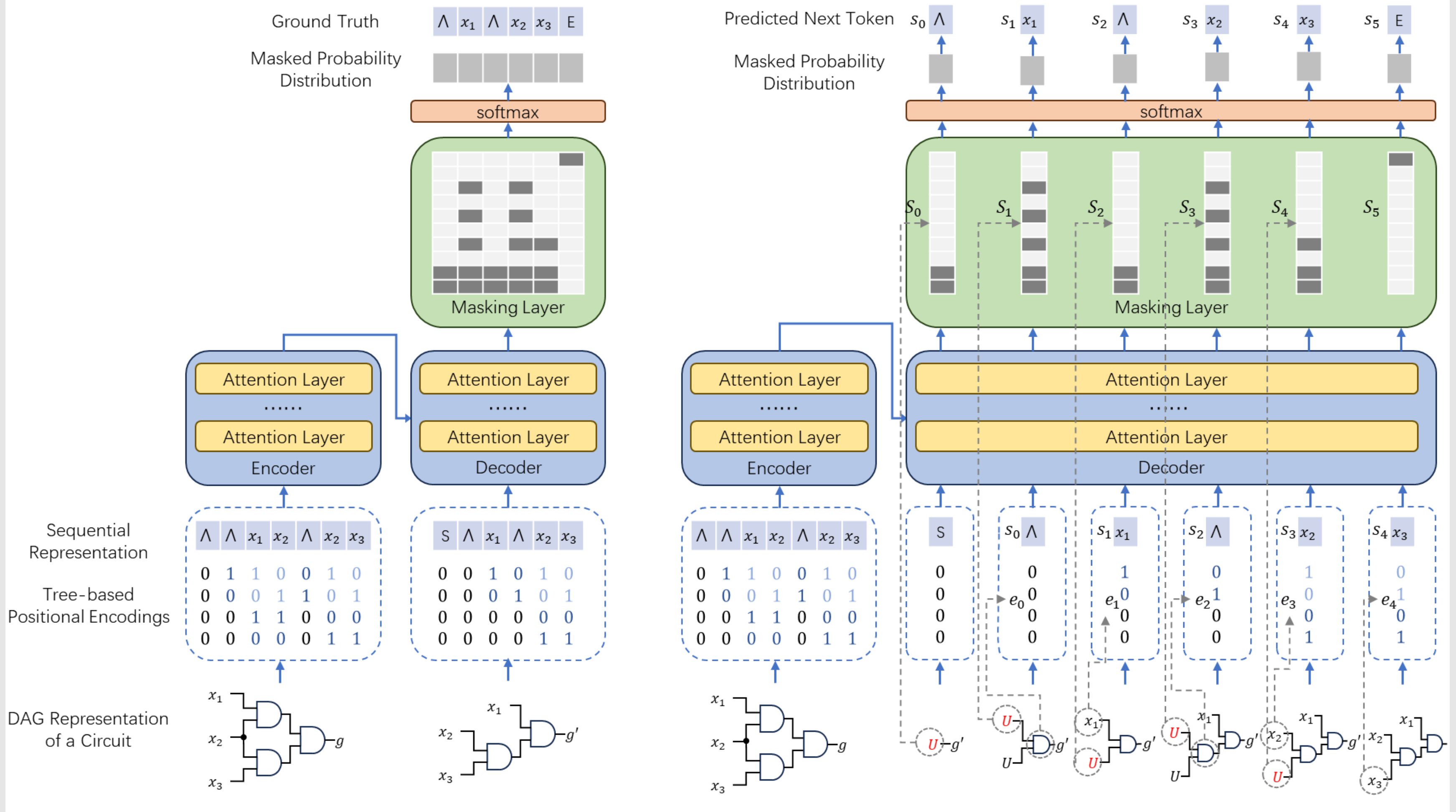
Replace U with each possible input / gate to see whether any equivalence conflict occurs

x_2	x_1	x_0	f	The value of g' when U is replaced by							
				x_0	$\overline{x_0}$	x_1	$\overline{x_1}$	x_2	$\overline{x_2}$	$\wedge$	$\overline{\wedge}$
0	0	0	**0**	0	U	0	U	0	U	U	U
0	0	1	**1**	U	0	0	U	0	U	U	U
0	1	0	**0**	0	U	U	0	0	U	U	U
0	1	1	**0**	U	0	U	0	0	U	U	U
1	0	0	**0**	0	U	0	U	U	0	U	U
1	0	1	**0**	U	0	0	U	U	0	U	U
1	1	0	**0**	0	U	U	0	U	0	U	U
1	1	1	**1**	U	0	U	0	U	0	U	U
				✓						✓	✓

U = Unknown
Red = Conflict with f

The Circuit Transformer

- The Boolean function is encoded by the Transformer encoder
- A masking layer is added before the softmax layer of the Transformer decoder
- Only allows tokens in valid choices S_t to be predicted



The Training Stage

The Inference Stage

Experiments

Train a Circuit Transformer with 88M parameters to generate minimized circuits for 8-input, 2-output Boolean functions.

Methods	In distribution: Random circuits		Out of distribution: IWLS FFWs	
	Unsuccessful cases	Avg. size	Unsuccessful cases	Avg. size
Boolean Chain	5.07% (5.07%)	15.25	11.36% (11.26%)	17.24
Boolean Chain (beam size = 16)	2.16% (2.16%)	14.89	6.34% (6.29%)	17.15
Boolean Chain (beam size = 128)	1.91% (1.91%)	14.87	5.97% (5.94%)	17.15
AIGER	4.32% (4.32%)	15.14	8.35% (7.77%)	17.19
AIGER (beam size = 16)	1.85% (1.85%)	14.87	4.62% (4.37%)	17.12
AIGER (beam size = 128)	1.71% (1.71%)	14.86	4.24% (3.99%)	17.12
Circuit Transformer w/o TPE	2.14% (0%)	15.02	6.63% (0%)	17.33
Circuit Transformer	1.14% (0%)	14.79	4.76% (0%)	17.17
Circuit Transformer ($K = 10$)	0.20% (0%)	14.02	2.83% (0%)	16.92
Circuit Transformer ($K = 100$)	**0.17%** (0%)	**13.73**	**2.63%** (0%)	**16.73**
Resyn2 (ground truth for training)	/	14.56	/	16.82

Better performance with MCTS enabled

Zero violation of equivalence constraints

Paper

Code

Slide, Poster, Video and Online Demo: https://snowkylin.github.io/publications

Install: `pip install circuit-transformer`

Correspondence email: xihan.li@cs.ucl.ac.uk