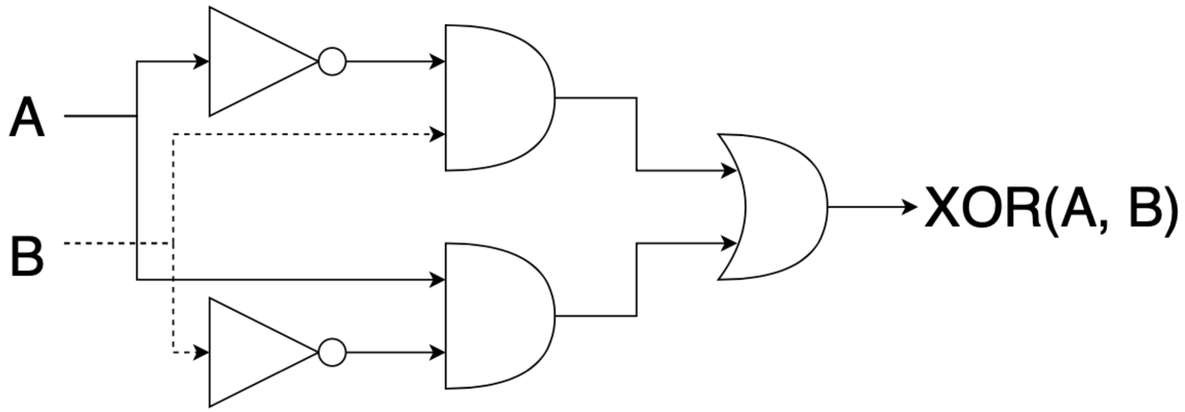
The focus of combinatorial logic circuit design is minimization of Boolean expressions

- Combinatorial logic circuits consist of Boolean gates
  - AND, OR, NOT
- Outputs depend on current inputs only
  - No state or memory
- Want correct circuits with minimal size

Combinatorial logic circuit design is difficult with more than 4 input variables

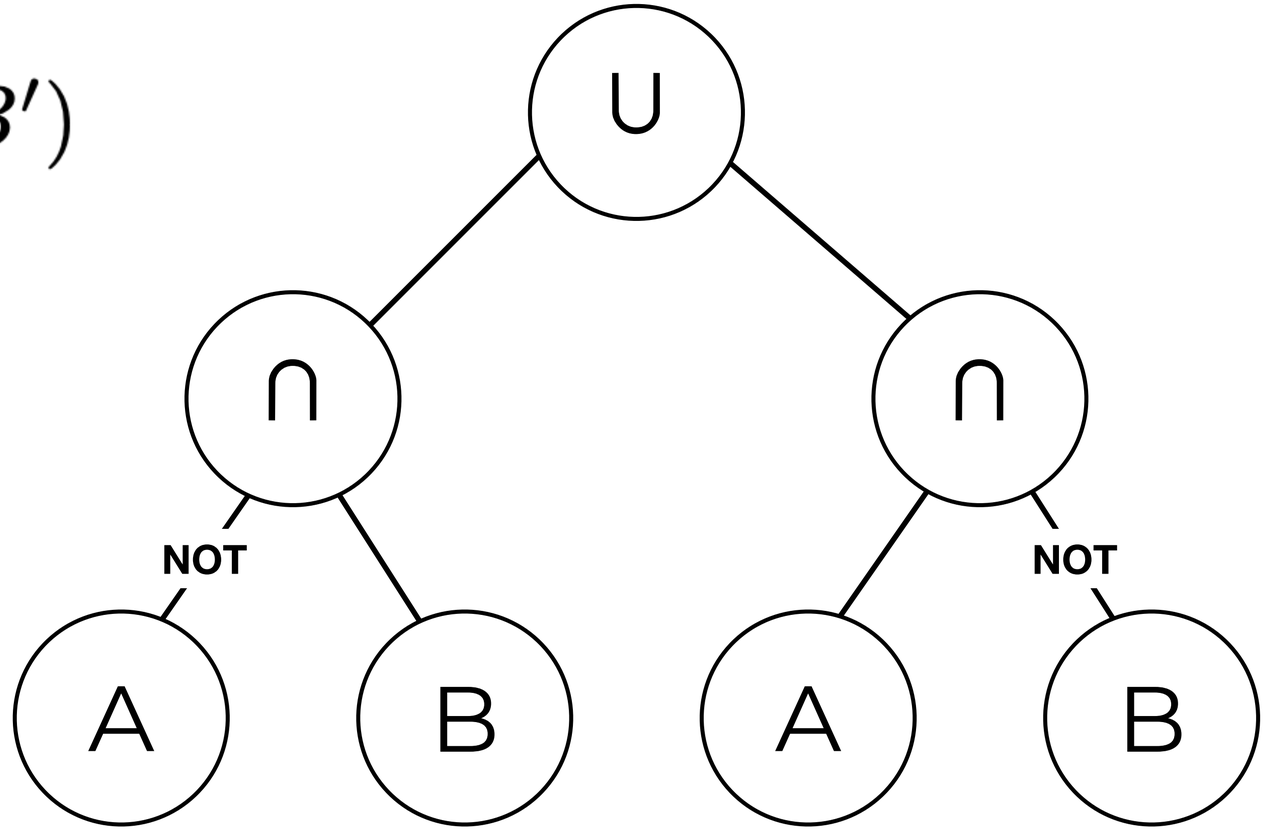
- Design typically done using deterministic algorithms
  - e.g. Karnaugh maps, Quine-McClusky method
  - Do not scale well beyond n=4, n=10 respectively

A	B	XOR(A, B)
0	0	0
0	1	1
1	0	1
1	1	0



Boolean expressions can be represented in the form of binary trees

$$(A' \cap B) \cup (A \cap B')$$



Binary tree representation naturally leads to genetic programming (GP) for optimization

- GP is an evolutionary optimization technique for tree data structures
- Reproduction operations may include:
    - Mutation, subtree crossover, permutation, editing, lifting, …
- Define objective functions for correctness and size
- Inputs and outputs defined in truth table

High-level pseudocode for algorithm

```
input hyperparameters
input truth table

create initial population using truth table

for number of generations:
    evaluate population correctness
    evaluate population sizes
    assign fitnesses
    select mating pool
    perform reproduction

extract best program from population
```

Ran algorithm 10 times for simple problem with 3 inputs

		in	out
A	B	C	
0	0	X	0
0	1	X	1
1	X	0	1
1	X	1	0

- Yielded several unique correct solutions
  - Sizes range from 5 to 6 gates
- Median wall time = 2 seconds

$$((B \cap A') \cup (A \cap C'))$$

Next steps include improving the algorithm and using more test cases

- Add more reproduction operators
    - Permutation, editing, lifting, encapsulation
- Test algorithm on different input sizes (e.g. n=4, n=10, n >> 10)
- Compare results with other methods