Miguel Alcón Doganoc
Combinatorial Problem Solving
May 7, 2019

Logic Synthesis

1 Description of the problem

In this project, our goal is to solve the *NOR Logic Synthesis Problem* (NLSP): given a specification of a Boolean function $f(x_1, ..., x_n)$ in the form of a truth table, find a NOR-circuit satisfying the specification that minimizes depth (and, in case of a tie in depth, with minimum size). An instance of NLSP consists in:

- $\mathbf{n}$:= "Number of input signals"
- $\mathbf{y_t}$:= "Desired output signal, described by row t in the truth table", where $t \in \{0, 1, ..., 2^n - 1\}$

2 Decision variables

Given the number of input signals n, the depth d, and the truth table of the logical circuit, I defined the following variables:

- $\mathbf{Z_{i,j}}$:= "The node (i,j) contains a constant zero", where
  - $0 \leq i \leq d$
  - $0 \leq j < 2^i$
- $\mathbf{N_{i,j}}$:= "The node (i,j) contains a NOR gate", where
  - $0 \leq i \leq d$
  - $0 \leq j < 2^i$
- $\mathbf{I_{i,j,k}}$:= "The node (i,j) contains the input k", where
  - $0 \leq i \leq d$
  - $0 \leq j < 2^i$
  - $1 \leq k \leq n$
- $\mathbf{B_{i,j}^{(t)}}$:= "Boolean value of the node (i, j) for the row t of the truth table", where
  - $0 \leq i \leq d$
  - $0 \leq j < 2^i$
  - $0 \leq t < 2^n$

For example, for a NOR-circuit that implements the functionality of an AND gate (see figure 1), with $n = d = 2$, one possible solution for the variables $c_{i,j}$ and $B_{i,j}^{(0)}$ is shown in figures 2e and 2f, respectively.

x_1	x_2	y	t
0	0	0	0
0	1	0	1
1	0	0	2
1	1	1	3

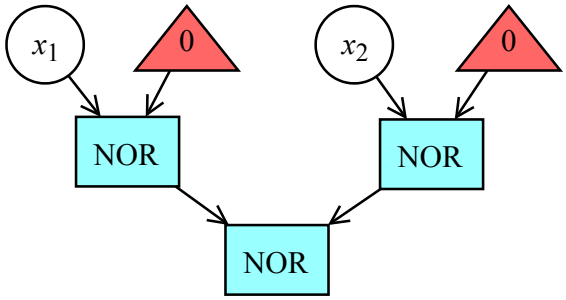
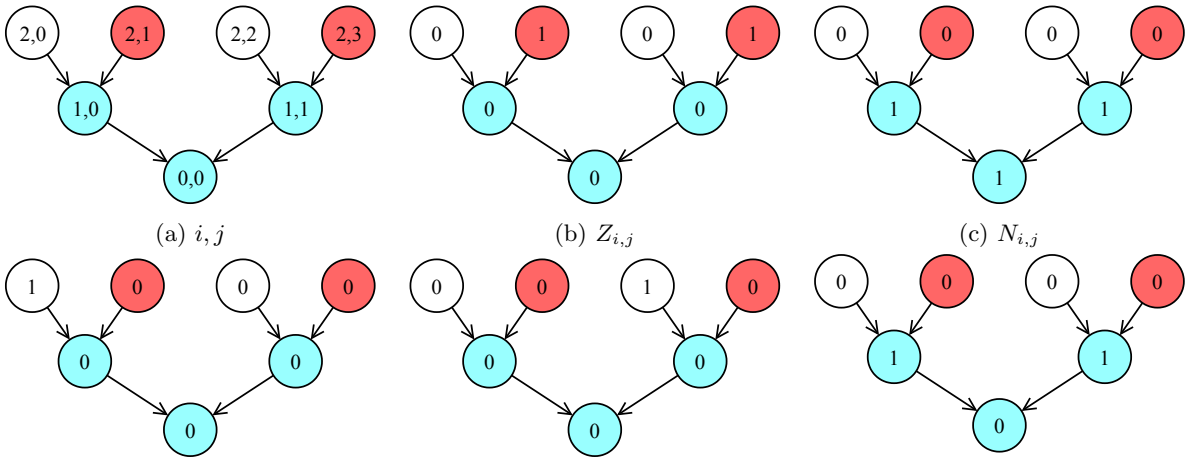


Figure 1: Truth table of $y = \mathrm{AND}(x_1, x_2)$ and NOR-circuit implementing it.



(a) i, j (b) $Z_{i,j}$ (c) $N_{i,j}$

(d) $I_{i,j,1}$ (e) $I_{i,j,2}$ (f) $B_{i,j}^{(0)}$

Figure 2: Visual representation of (i, j) and the variables.

3 Constraints

In order to simplify the definition of the constraints, I define three functions. Given the variable $v_{i,j}$, with $v_{i,j} \in \{Z_{i,j}; N_{i,j}; I_{i,j,k}; B_{i,j}^{(t)}\}$,

- **left**$(v_{i,j})$:= "Variable corresponding to the one on the left of $v_{i,j}$" $= v_{i+1,2\times j}$.
- **right**$(v_{i,j})$:= "Variable corresponding to the one on the right of $v_{i,j}$" $= v_{i+1,2\times j+1}$.
- **bit**(k,t) := "Boolean value of x_k in the t row of the truth table" = "Value of the position k of the binary representation of t (i.e. $t_k \in \{t_1 t_2 ... t_n\}$)".

So, the constraints are the following:

- The output of the circuit is equal to the desired value for each row t of the truth table.

$$b_{0,0}^{(t)} = y_t$$
$$\forall t < 2^n$$

- NOR gates are not allowed on the leaves of the circuit.

$$c_{d,j} \geq 0$$
$$\forall j < 2^d$$

- Force children (if any) of each node to be 0 if the node is not a NOR gate.

$$c_{i,j} \geq 0 \Rightarrow (\textbf{left}(c_{i,j}) = 0 \wedge \textbf{right}(c_{i,j}) = 0)$$
$$\forall i < d \; \forall j < 2^i$$

- Force the left child of NOR gates to be greater or equal than the right one (non-symmetry of NOR gates).

$$c_{i,j} = -1 \Rightarrow (\textbf{left}(c_{i,j}) \geq \textbf{right}(c_{i,j}))$$
$$\forall i < d \; \forall j < 2^i$$

- Force the left child of NOR gates to be greater than the right one if one of them is an input signal (non-symmetry of NOR gates).

$$(c_{i,j} = -1 \wedge (\textbf{left}(c_{i,j}) > 0 \vee \textbf{right}(c_{i,j}) > 0)) \Rightarrow (\textbf{left}(c_{i,j}) > \textbf{right}(c_{i,j}))$$
$$\forall i < d \; \forall j < 2^i$$

- Link each NOR gate with its corresponding value, which is the NOR operation between both children.

$$c_{i,j} = -1 \Rightarrow (B_{i,j}^{(t)} = \neg(\textbf{left}(B_{i,j}^{(t)}) \vee \textbf{right}(B_{i,j}^{(t)})))$$
$$\forall t < 2^n \; \forall i < d \; \forall j < 2^i$$

- Link each constant 0 signal with 'false'.

$$c_{i,j} = 0 \Rightarrow \neg B_{i,j}^{(t)}$$
$$\forall t < 2^n \; \forall i \leq d \; \forall j < 2^i$$

- Link each input signal that has value 1 in the truth table, with 'true'.

$$c_{i,j} = k \Rightarrow B_{i,j}^{(t)}$$
$$\forall 1 \leq k \leq n \; \forall t < 2^n \; \forall i \leq d \; \forall j < 2^i \text{ where } \mathbf{bit}(k,t) = 1$$

- Link each input signal that has value 0 in the truth table, with 'false'.

$$c_{i,j} = k \Rightarrow \neg B_{i,j}^{(t)}$$
$$\forall 1 \leq k \leq n \; \forall t < 2^n \; \forall i \leq d \; \forall j < 2^i \text{ where } \mathbf{bit}(k,t) \; = 0$$

4 Relation between the documentation and the code

4.1 Variables

- `IntVarArray circuit =` $\{c_{0,0}, c_{1,0}, c_{1,1}, ..., c_{d,2^d-1}\}$.
- For each t, which in the code is `truth_idx`, `BoolVarArray circuit_bool =` $\{b_{0,0}^{(t)}, b_{1,0}^{(t)}, b_{1,1}^{(t)}, ..., b_{d,2^d-1}^{(t)}\}$.

4.2 Functions

- $\mathbf{left}(i,j)$ and $\mathbf{right}(v_{i,j})$ are not directly implemented as functions.
- Function `bool is_bit_up(bit_pos, number)` $= \mathbf{bit}(k,t)$.

4.3 Others

- Most of the constrains are defined within the recursive function `constraint_creation(...)`, accessing to variables like a tree.
- The algorithm tries to solve the problem from *depth* 0 to `MAX_DEPTH`, which is equal to 5, and stops in the first *depth* where it finds a solution.
- I used 4 threads when solving the instances.