Miguel Alcón Doganoc
Combinatorial Problem Solving
June 24, 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, the size s, 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
  - $0 \leq i \leq d$
  - $0 \leq j < 2^i$
  - $1 \leq k \leq n$
- $\mathbf{B^{(t)}_{i,j}}$:= "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 (variable assignation) is shown in figure 2.

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

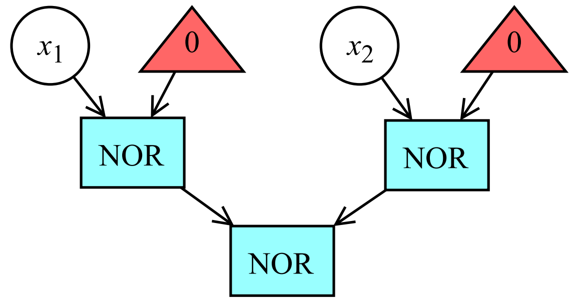
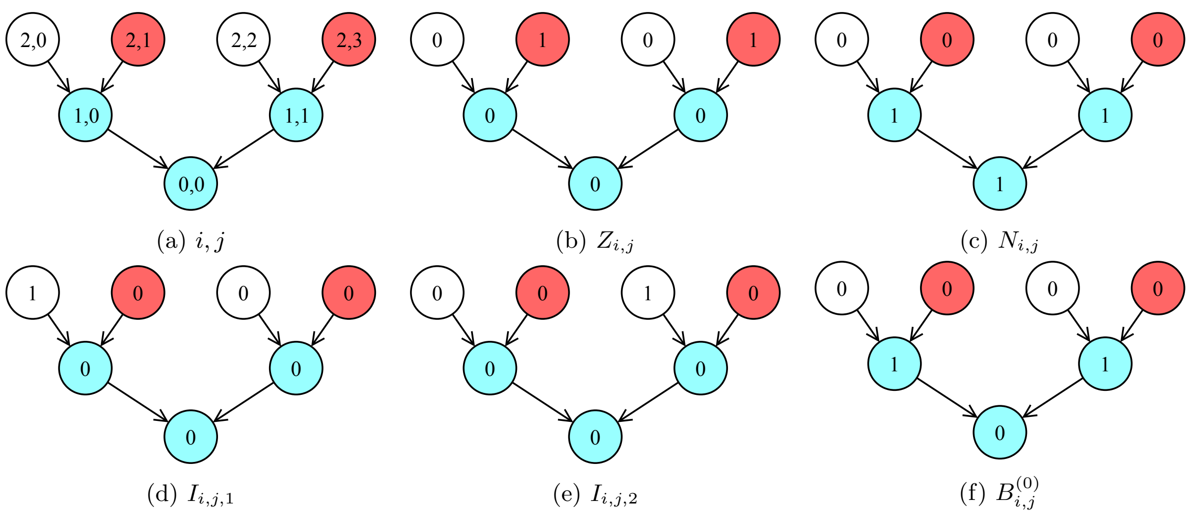


Figure 1: Truth table of $y = \text{AND}(x_1, x_2)$ and NOR-circuit implementing it.



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

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

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

3 Constraints

In order to simplify the definition of the constraints, I define the following 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 $v_{i,j}$" $\equiv v_{i+1,2\times j}$.
- **right**$(v_{i,j})$:= "Variable corresponding to the one on the right of $v_{i,j}$" $\equiv v_{i+1,2\times j+1}$.
- **bit**(k,t) := "Boolean value of x_k in the t-th row of the truth table" $\equiv$ "Value of the position k of the binary representation of t (i.e. $t_k \in \{t_1 t_2 ... t_n\}$)".
- **AMO**(l) := "Given the list of variables l, at most one of the variables in it can be true."
- **AMK**$(k; l)$:= "Given the list of variables l, at most k of the variables in it can be true."

I defined **AMO** and **AMK** as it is explained in the SAT encoding slides. The constraints that define the problem are the following:

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

$$\begin{aligned} \text{if } y_t \text{ then } B_{0,0}^{(t)} \\ \text{otherwise } \neg B_{0,0}^{(t)} \\ \forall t < 2^n \end{aligned}$$

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

$$\begin{aligned} \neg N_{d,j} \\ \forall j < 2^d \end{aligned}$$

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

$$\begin{aligned} N_{i,j} \vee \textbf{left}(Z_{i,j}) \\ N_{i,j} \vee \textbf{right}(Z_{i,j}) \\ \forall i < d \ \forall j < 2^i \end{aligned}$$

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

$$\begin{aligned} \neg N_{i,j} \vee \neg\textbf{left}(B_{i,j}^{(t)}) \vee \neg B_{i,j}^{(t)} \\ \neg N_{i,j} \vee \neg\textbf{right}(B_{i,j}^{(t)}) \vee \neg B_{i,j}^{(t)} \\ \neg N_{i,j} \vee \textbf{left}(B_{i,j}^{(t)}) \vee \textbf{right}(B_{i,j}^{(t)}) \vee B_{i,j}^{(t)} \\ \forall t < 2^n \ \forall i < d \ \forall j < 2^i \end{aligned}$$

- Link each constant 0 signal with 'false'.

$$\begin{aligned} \neg Z_{i,j} \vee \neg B_{i,j}^{(t)} \\ \forall t < 2^n \ \forall i \leq d \ \forall j < 2^i \end{aligned}$$

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

$$\begin{aligned} \text{if } \textbf{bit}(k,t) \text{ then } \neg I_{i,j,k} \vee B_{i,j}^{(t)} \\ \text{otherwise } \neg I_{i,j,k} \vee \neg B_{i,j}^{(t)} \\ \forall 1 \leq k \leq n \ \forall t < 2^n \ \forall i \leq d \ \forall j < 2^i \end{aligned}$$

- Force each node to be only of one type.

$$\begin{aligned}\mathbf{AMO}(\{Z_{i,j}, N_{i,j}, I_{i,j,1}, I_{i,j,2}, ..., I_{i,j,n}\})\\ Z_{i,j} \vee N_{i,j} \vee I_{i,j,1} \vee I_{i,j,2} \vee ... \vee I_{i,j,n}\\ \forall i < d\ \forall j < 2^i\end{aligned}$$

- Limit the number of NOR gates to be less than size.

$$\begin{aligned}\mathbf{AMK}(s; \{Z_{i,j}, N_{i,j}, I_{i,j,1}, I_{i,j,2}, ..., I_{i,j,n}\})\\ Z_{i,j} \vee N_{i,j} \vee I_{i,j,1} \vee I_{i,j,2} \vee ... \vee I_{i,j,n}\\ \forall i < d\ \forall j < 2^i\end{aligned}$$

3.1 Worsen performance

I tried to use some constraints that at the end affected negatively to the performance of the program.

- Force non-symmetry of NOR gates' children.
  - Do not allow the same input on both sides.

$$\begin{aligned}\mathbf{AMO}(\{\mathbf{left}(I_{i,j,k}), \mathbf{right}(I_{i,j,k})\})\\ \forall i < d\ \forall j < 2^i\ \forall k \leq n\end{aligned}$$

4 Extra comments

I tried two implementations of **AMO**, the quadratic and logarithmic encodings. At the final version of the program, I used the quadratic one because I had better performance with it. Both encodings are implemented in the program, but the logarithmic one is not used.
I also used the `frozenset` of *Python* to avoid repeating clauses and variables inside the clauses.
The program is able to solve all the problems in 511s. With 1 min of `timeout`, it never has to stop the program because it finished its execution before. Inside the 'out/' directory you can find the solutions for to problems.