# Exercises Lecture 7
# Intelligent Systems Programming (ISP)

## Exercise 1 (adapted from A97 4.1)

Construct the ROBDD for $\neg x_1 \wedge (x_2 \Leftrightarrow \neg x_3)$ with ordering $x_1 < x_2 < x_3$ using the algorithm BUILD in figure 9 of A97. Show the recursive call structure and the final content of the unique table.

## Exercise 2 (adapted from A97 4.3)

Suggest an improvement BUILDCONJ($t$) of Build which generates only a linear number of calls for Boolean expressions $t$ that are conjunctions of variables and negations of variables.

## Exercise 3 (adapted from A97 4.4)

Construct the ROBDDs for $x$ and $x \Rightarrow y$ for any variable ordering you want. Compute the disjunction of the two ROBDDs using APPLY.

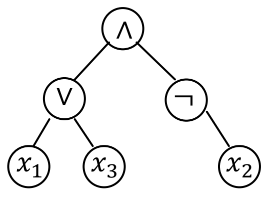
## Exercise 4 (adapted from A97 4.5)

Draw the ROBDD for $\neg(x_1 \wedge x_3)$ and $x_2 \wedge x_3$ using BUILD with the ordering the ordering $x_1 < x_2 < x_3$. Use APPLY to find the ROBDD for $\neg(x_1 \wedge x_3) \vee (x_2 \wedge x_3)$.

## Mandatory assignment

During lecture you have learned how to create an ROBDD using BUILD in $O(2^n)$ time. As stated during lecture, a more efficient way is to use APPLY and construct the ROBDD bottom-up from the expression tree. The goal of this assignment is to write the pseudo-code for this algorithm.

Consider the expression tree of $(x_1 \vee x_3) \wedge \neg x_2$ shown below



This expression tree and general Boolean expression trees can be represented by the data structure:

| Expr | | |
|---|---|---|
| type() | {VAR, NOT, AND, OR, TRUE, FALSE} | *Return the type of expression* |
| left() | Expr | *Return the left Expr node of an operation* |
| right() | Expr | *Return the right Expr node of an operation* |
| idx() | integer | *Return the ID of Expr* (*only for type VAR*) |

For negated expression (type = NOT), you can assume that right() holds the expression under negation.

1. How can you use apply to negate a ROBDD *u*?
   (hint: use one of the 16 Boolean operators and a terminal ROBDD (0 or 1) as your two other arguments to APPLY)
2. Use your result in 1. to write the pseudo-code of an algorithm that uses APPLY to create a ROBDD from a Boolean expression of type Expr.