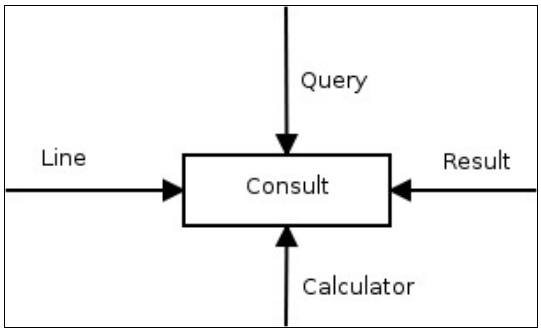
Logical operation (kinda natural language)

Description of the system

We need to analyze data from data base in order to get a result. We need to get or not the first line that match (Null for nothin). Each time that data base is consulted with the same query it get the next result till the end of the data base is reached. Data base structure will be described in another paper. A calculator will take care of the query and try to make it match with the line as an input. The output must be true or false as a result. Now watch the SADT diagram that illustrate the principle:



Consulting a data from DB[1] (SADT diagram)

The query description

We define a query s.a[2] *a AND b OR c*
We define a, b and, c as logical operations on the columns from the current line from DB. We admit that a, b and, c can take the value true(1), false(0). We can develop *a* as the test '*is the column called RANK is equal to 1*' which can be written as *'RANK=1'* hence the result is true if it matches or false otherwise. Same thing for *b* and *c*.
Then to tie up all these results, logical operands between each column tests will do the nasty stuff. This nasty stuff can be called query and modelised on a certain point of view as a calculator.

Works on the query *a AND b OR c* by itself

For the time being logical operators AND, OR will be used to simplyfy modelisation. Logical operators can be defined by logical tables. Now we an extend our previous exemple.
Lets say that a is true, b false and, c true.

We say that we don't have any operator precedence for the time being. In the future we will add parenthesis (which increase operation of precedency). Let's focus for now on, on a basic operations:

We have AND operator:

AND	**0**	**1**

1 DB means Data Base
2 s.a means such as

0	0	0
1	0	1

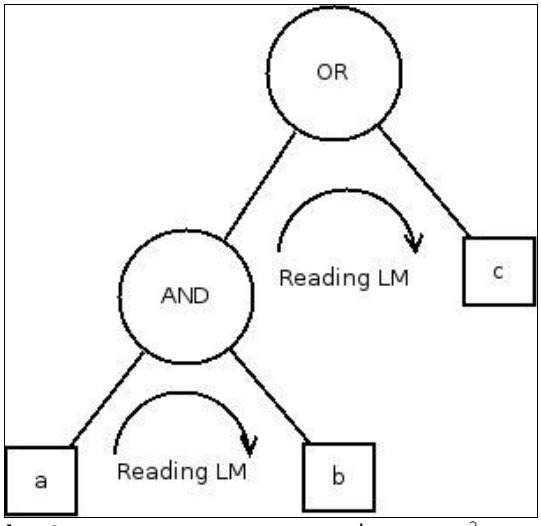
Table for AND logical operator

We have OR operator:

OR	0	1
0	0	1
1	1	1

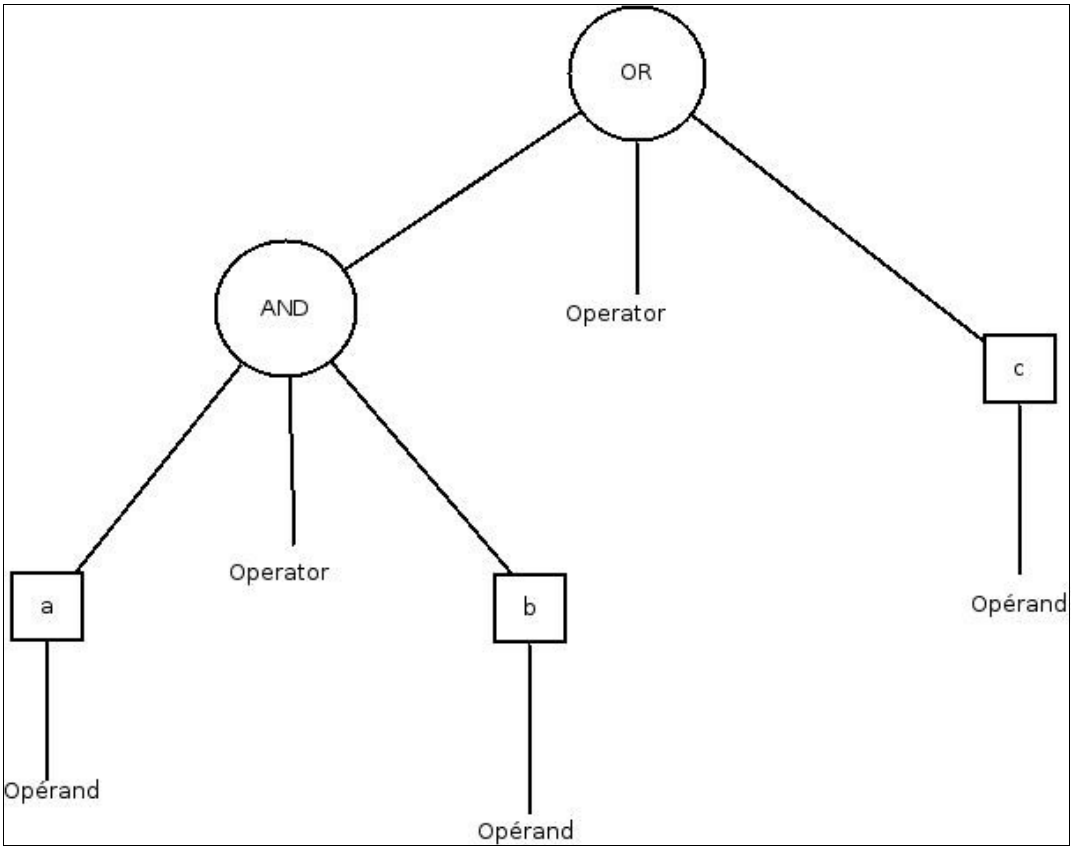
Table for OR logical operator

Lets represent with a tree (left most reading) the query (no precedency for the time being) on the a AND b OR c:



a AND b OR c representation LM[3] as a tree

3 LM stand for Left Most reads from left to right



Hierarchy and relationship between operand and operator

To solve this problem many algorithms exist. The one choose is the one from the stack[4]. This is a well known problem n the theory of formal languages. To define priorities we will use the algorithm to solve operations. This ad-hoc method can be easily adapted to our problem. Problem of arity of logical operand will be solved.

The Theory first part

Operator precedence parsing is based on bottom-up shift-reduce parsing. As an expression is parsed tokens are shifted to a stack. At the appropriate time the stack is reduced by applying the operator (opr) to the top of the stack. This is best illustrated by example.

opr	val	input	action
$	$	(1+2)*3$	shift
$(	$	1+2)*3$	shift
$(	$1_	+2)*3$	shift
$(+	$1_	2)*3$	shift
$(+	$1 2	)*3$	reduce
$(	$3_	*3$	shift
$	$3_	*3$	shift
$*	$3_	3$_	shift
$*	$3 3	$_	reduce
$	$9_	$_	accept

Equation sovled: (1+2)*3$

4 THOMAS NIEMANN Portland, Oregon web site: epaperpress.com

$ is a delimitor

From the above example we deduce the table below:

V2		input				
		+	*	(	)	$
opr	+	reduce	shift	shift	reduce	reduce
	*	reduce	reduce	shift	reduce	reduce
	(	shift	shift	shift	shift	error
	)	reduce	reduce	error	reduce	reduce
	$	shift	shift	shift	error	accept

Relation ship between opr and input

Theory part 2

Now we can work with logical operator *AND* and *OR*. We define the arity of the two operator as the same value. To sum up *AND* and *OR* have the same priorities. In a basic logical expression to define priorities we use parentheses.