JEYHUN YAGUBLU : S260229 . APA 01.28.2020 EXAM REPORT

I stored the given formula in a data structure which is very close to its file format. I have struct **formula_u** (which I type defined as **formula_t**) which consists of 2 integers and one struct: **int n_clause** (number of clauses in formula) and **int n_variables** (number of variables that we have in formula) and pointer to the array of **struct clause_u** in which I stored clauses of formula.

Struct clause_u in itself consists of **int n_var** (number of variables in that clause) and **int* vars** which is pointer to an array of integers (every field of array tells us if we need to disjunct variable itself (>0) or its **NOT** (<0)).

It is obvious that we need to check different mappings of variables (consisting of 0 and 1 s). For that purpose I used Powerset of n bits. I could do it with recursion choosing first variable to be 1 and then 0 but I decided to do it with the help of binary computations (because problem is about Boolean so I thought it would be in place). I created array of integers size of (n+1) and then **from i=0 till ((2 to the power of n+1) -1**) I converted my array to the one corresponding to binary representation of **i**. I treated every box of array as one digit.

Function named **turn_int_tobitarray** does that for me. Now that I have array which represents if the variable x is 1 or 0 (**arr[x]= 0 or 1**) I can check if this mapping gives true for given formula.

Function named **check_sol** does that. It reads my formula clause by clause and puts values of variables there and then it conjuncts clauses. Function **eval** decides what number should we put (variable itself or its NOT) .

If **check_sol** finds that this mapping return true for that formula it puts our **flag to 1**(which is initially set to 0) And then if our flag is 1 **print_sol** will print our solution in a formatted way **(x$ is $** etc) .If flag is 0 then our function will print error saying that this formula is **UNSAT** .

Differences in my program from my pape :

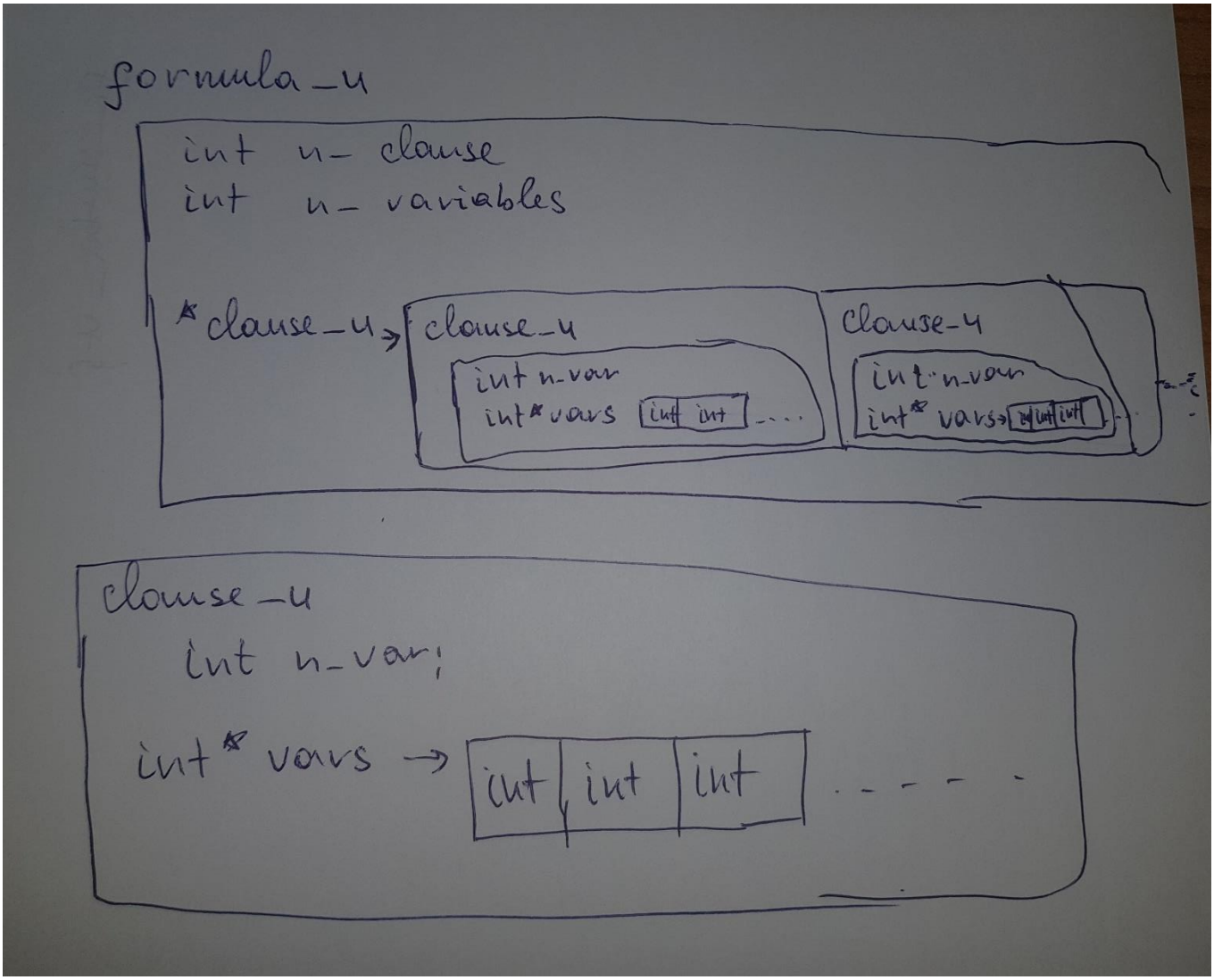
1)in function **load_file** I confused arguments of fgets (I wrote FILE* in first and then char*s).

2)in function **turn_int_tobitarray**: on paper I confused and wrote **(2<<i)&n** which must be **(1<<i)&a**

3)in function **eva**l I forgot to put else before second if

4)in function **solve,** in for I confused and again wrote **(2<<i)** which must be **(1<<i)**

I am aware that my program checks same solution twice because I wanted to start from the 1^{st} place of array of bits so 0^{th} digit is neglected and still changes to 0 and 1. So program checks twice the same solution .I could simply avoid it by using arr[x-1] and not arr[x] when I want to find value of variable but as program still works correctly I decided to not change it as it would make my program to be a bit different from what is on paper.


formula_4
int n_clause
int n_variables
*clause_4
clause_4
int n_var
int* vars
clause_4
int n_var
int* vars
clause_4
int n_var;
int* vars
int
int
int