Joonas Pelttari

## Semantic checks

Level 1 implemented. Variables are global and must be defined when used, same for functions and procedures. Defining var, func or proc multiple times not allowed.

I visit the nodes and add vars, procs, funcs and formal args to the symbol table. Then I visit all again and check that everything is defined and not multiple times. One difference comes with formal args because they are “undefined” naturally because they first appear in the func/proc definition without “var” keyword so I used a semdata variable to track if something is formal arg or not.

Level 2 partly implemented. Number of params for func/proc is checked but the return type check of procedure is NOT implemented.

When finding a func/proc call I check that the actual parameter number matches with formals in the func/proc definition node.

Optionally I also check the type of actual parameters to match the type of formal parameters. They can be int or date_literal and if function expects e.g. (int, date_literal, int), then caller must give same (int, date_literal, int).

## Interpretation levels

Level 1 implemented.

With ints, dates and variables I just return the value and print statement loops and prints all print items. Operators do the needed calculation using left and right value they get when evaluating the left and right expressions.

Level 2 implemented.

Variables are saved to symbol table when defined. Assignment checks if we are having “id_name” or “attr_assign” because handling them is a bit different.

Level 3 implemented.

Interpreter checks what attr (e.g “day”) we are having and returns that value or if assigning, creates a new date with the given attribute changed.

Level 4 implemented.

Unless-statement evaluates if the condition is false or true, and then evaluates the statements or otherwise statements.

Unless-expression is similar but with expressions.

Until-loop evaluates the statements once, and then checks if the condition is still true. If yes, does the same again, if not, breaks the loop.

Joonas Pelttari

Level 5 partly implemented.

I implemented function. Calling the function initializes the parameters to the value the caller gives and then the result of evaluating the function body is returned.

## Notes

I think this assignment was fun and challenging enough even with the easier parts I implemented.

For me the choice to use “id_name” for variable definitions and also when the variable is used made some little inconveniences, but I managed with it. Also the formal args were a bit difficult at first because they are not defined like “normal” variables and I had some hassle between them, the actual variable inside function and then calling with a value that needs to be given to that formal arg.

Also I had some hassle because I accidentally used both “int” and “int_literal” types.

The semantic checks were a bit hard for me and I found interpreting easier.