UNIVERSITÉ TOULOUSE III PAUL SABATIER

Labwork 5 – Extensions to CellLang

These labworks are automatically assessed based on an archive you have to deliver on time on the Moodle webpage. To make the archive, you have to type the command:

```
> make archive
```

This produces a file named `archive.tgz` that you have to deposit. As the compilation labworks are held each week, deposit deadline is set one day before your next session (to perform the assessment). This deadline is not strict but delay will have a negative impact on your mark and you will not benefit from the comments of your teacher.

This exercise consists in implementing an extension to *CellLang*. The following process has to be applied:

1. Understand the proposed syntax and extension.
2. If needed, add required tokens to `parser.mly`.
3. If needed, add the token scanning to `lexer.mll`.
4. Add the required rules to `parser.mly`.
5. Build the AST in `parser.mly`.
6. Provide the translation to quads in `comp.ml`.

Very important! replace the `Makefile` and `ast.ml` with the ones provided in Moodle.

There is an extension to assignments that uses a special condition form as below:

```
x :=
        y + 1  when [0,1] = 1,
        y      when [0, 0] = 1,
        y - 1  otherwise
```

Semantics is clear:

- If the first condition following keyword `when` is true, the first expression is assigned.
- Else if the second is true, the second expression is assigned.

- Otherwise the last expression is assigned.

This is a shortcut to `if` selection when the selection aims to assign the same variable in all of its alternatives.

To implement this *CellLang* , the following constraint must hold:

- The number of `clauses` is not bounded.
- `otherwise` is required and always at the end of `when` clauses.
- An `otherwise` with 0 `when` is also accepted.
- The assigned memory may be a variable but also a cell.
- `when` and `otherwise` clauses are separated by commas '`,`'.
- The `when` and `otherwise` are not part of an expression but of the assignment.

Test Sources :

- `test53/simple.auto`
- `test53/multi.auto`
- `test53/cell.auto`
- `test53/fail.auto` (*that must fail !*).

Hint: x `:=` e_1 `when` c_1, e_2 `when`, ..., e_n `otherwise`
is really equivalent to:

```
if c1 then
        x := e1
elif c2 then
        x := e2
elif
        ...
else
        x := en
end
```