Language processing

This activity gives you a chance to extend the language processing case study in a number of different directions.

Adding operations

Can you add more operations to the expression type and all the functions working over it. Examples to add include

- Subtraction (binary operation).
- Unary minus (how will this interact with subtraction?)
- Absolute value (unary operation).
- Division: this operation can raise an error when division by zero: you will need to handle this.

Adding "variables"

Can you add definitions to the system, with the form `let v=e1 in e2` which gives `v` the (value of) `e1` in the expression `e2`.

- You should think about how to *scope* this definition: what happens if there is already a definition of `v` in scope? Is it possible to use that definition of `v` within the expression `e1`?
- What about extending this to functions: you could allow `e1` to be something like an Erlang `fun` expression: could you use `v` recursively within `e1`?

Adding other types

It is possible to add *conditional* expressions if you allow boolean valued expressions such as the `b` in `if b then e1 else e2`.

- How would you have to extend all the functions that we have written so far?
- What boolean operations could you add to this?
- How would adding these affect compilation and the virtual machine for expressions?

Simplification

We have looked at simplifications which work with numbers `0` and `1` but it is also possible to simplify all completely numeric expressions like `(2+3)*4` to values (`20` in this case). How would you add simplification like this, and how could you ensure that it is done most efficiently?

Are there any other simplifications that you could imagine doing? How about changing the order of arguments? What do you have to be careful about with this?

Another transformation – it might not strictly be a simplification – is to *re-associate* expressions so that, for example, all additions are left-associated: this would transform `(1+(2+3))` to `((1+2)+3)`, for example.

Changing the syntax

We‘ve written expressions in fully bracketed form so that parsing is easy. How could you transform the parser to accept expressions which are less bracketed: as an example, so that `2+3*4` could read as representing `(2+(3*4))` in its fully parenthesised form.