Assignment 1

Rajbir Malik

2017CS10416

Theorem

$$\forall \texttt{ exptree e}$$
$$\texttt{mk\_big } (\texttt{eval(e)}) = \texttt{stackmc li (compile e)}$$

Proof

We shall be proving the theorem by applying induction on the `height` of `exptree`.

Base Case

The base case `height` for `exptree` is 0. In this case the tree `e`, is just `N(x)` for some `int`. In this case,

$$\texttt{eval(e)} = \texttt{x} \text{ and,}$$
$$\texttt{compile(e)} = \texttt{[CONST(mk\_big x)]}$$

which after evaluation with stackmc would give

$$\texttt{stackmc li (compile e)} = (\text{mk\_big x})$$

Thus, with this we can conclude

$$\texttt{mk\_big } (\texttt{eval(e)}) = \texttt{stackmc li (compile e)}$$

Inductive Hypothesis

Now, we assume that $\forall$ `exptree e` with `height(e)` $\leq k$ satisfy the above theorem, i.e. `mk_big` (`eval(e)`) = `stackmc li (compile e)`, where $k \geq 0$.

Inductive Step

Now, let `e` be an `exptree` with `height` $= k + 1$.

Now, since $k \geq 0$, `height` of `e` ≥ 1.

This ensures that `e` is of the form

(BIN of exptree*exptree)
or
(UN of exptree)

, where `BIN and UN` are binary and unary operations respectively.

Case 1 (Binary Operation)

Now, `e` is of the form `BIN(el, er)`. Since, `el and er` are subtrees of `e` their `height` is going to be $\leq k$. Thus, our induction hypothesis holds for these trees. Therefore,

`mk_big (eval(er))` = `stackmc li (compile er)` = `mk_big xr` (say)
`mk_big (eval(el))` = `stackmc li (compile el)` = `mk_big xl` (say)

Also,

`compile(e)` = `compile(el) @ compile(er) @ [BIN]`

Now, for `LHS`,

`eval(e)` = `eval(el) ** eval(er)`, and so
`eval(e)` = `xl ** xr`

where ** is the syntactic representation of `BIN`.

Now, for `RHS`

`stackmc l1 (compile e)` = `stackmc l1 (compile el)@(compile er)@[BIN]`

And, by the definition of stackmc, since `compile el` represents a complete tree, its values `mk_big xl` will be prepended to stack. The same goes for `compile er`. And, therefore,

`stackmc l1 (compile e)` = `stackmc ((mk_big xr)::(mk_big xl)::l1) [BIN]`

Now, following the definition of `stackmc`, the above expression evaluates to,

`stackmc l1 (compile e)` = `BIN (mk_big xl) (mk_big xr)`, which is same as

`BIN (mk_big xl) (mk_big xr)` = `mk_big (xl ** xr)` = `mk_big (eval(e)`

and, hence induction holds.

Case 2 (Unary Operation)

With most properties same as above and `e` as `UN(e')`, by induction hypothesis,

`mk_big eval(e') = stackmc li (compile e') = mk_big x'` (say)

Also,

`compile(e) = compile(e') @ [UN]`

Now, for `LHS`,

`eval(e) = #eval(e'),`
`eval(e) = #x`

where # is the syntactic representation of `UN`.

Now, for `RHS`

`stackmc l1 (compile e) = stackmc l1 (compile e')@[BIN]`

And, by the definition of stackmc, since `compile e'` represents a complete tree, its values `mk_big x'` will be prepended to stack. And, therefore,

`stackmc l1 (compile e) = stackmc ((mk_big x')::l1) [BIN]`

Now, following the definition of `stackmc`, the above expression evaluates to,

`stackmc l1 (compile e) = UN (mk_big x')`, which is same as

`UN (mk_big x') = mk_big (#xr) = mk_big (eval(e)`

and, hence induction holds.

Thus, induction holds, and thus the theorem is correct.

Note

We have used some properties which are cleared here

- `UN (mk_big x') = mk_big (#xr)`, from definition of `UN` in `bigint`
- `BIN (mk_big xl) (mk_big xr) = mk_big (xl ** xr)`, from definition of `BIN` in `bigint`