University of Sussex
Informatics

Spring 2020

Limits of Computation
Feedback to Exercises 2
(covers Lectures 3–4)
Dr Bernhard Reus

`WHILE`-programs: Syntax & Semantics

1. Which of the following are legal expressions or commands in *core* `WHILE` as presented in Lecture 3?

   Answers:

   (a) `tl nil` is a legal expression (core `WHILE`)

   (b) `tl rl` is a legal expression (core `WHILE`) as `rl` must be a variable.

   (c) `cons hd hd x x tl x` is an illegal expression as there are too many arguments, either for `hd` or `cons`.

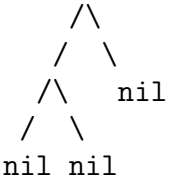
   (d) `while a { cons hd a }` is an illegal command, as there is no proper statement in the loop body. Moreover, also the expression in use is ill-formed as `cons` needs two arguments.

   (e) `if tl tl X { } else { X:= Y }` is a legal command in core `WHILE`.

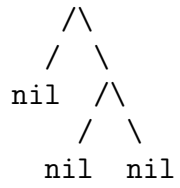
2. Given are the following elements of $\mathbb{D}$:

$$s = \langle\, \langle\, \text{nil.nil} \,\rangle.\text{nil} \,\rangle \qquad t = \langle\, \text{nil}.\langle\, \text{nil.nil} \,\rangle \,\rangle$$

   (a) Draw s and t as (two-dimensional) trees:
   Answer: This is s:



   This is t:



(b) Is $s = t$?

Answer: No these are different trees clearly.

(c) State whether s and t, respectively, can be read as numbers, and if so, which number?

Answer: s is not a number, but t represents 2.

(d) State whether s and t, respectively, can be read as lists, and if so, which lists?

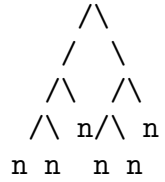
Answer: Yes, s represents `[1]` or `[[0]]` or `[[nil]]` or `[[[]]]`, while t encodes `[0,0]` or `[[],[]]` or `[nil,nil]`. Multiple answers are possible as a tree can encode several values of different "type". Types are just in the eye of the beholder.

3. Consider the binary tree t in linear notation:

$$\langle\,\langle\,\langle\,\mathrm{nil.nil}\,\rangle.\mathrm{nil}\,\rangle.\langle\,\langle\,\mathrm{nil.nil}\,\rangle.\mathrm{nil}\,\rangle\,\rangle$$

(a) Draw t in the usual two-dimensional form.

Answer (we abbreviate nil by `n`):



(b) Does t encode a natural number, and if so, which number?

Answer: NO, the tree does not have the required format (consider that a number corresponds to a list of nils).

(c) Does t encode a list of natural numbers, and if so, which list of numbers?

Answer: NO, the second element would work – $\langle\,\mathrm{nil.nil}\,\rangle$ – but not the first element – $\langle\,\langle\,\mathrm{nil.nil}\,\rangle.\mathrm{nil}\,\rangle$ – as it does not encode a number.

(d) Does t encode a list of lists of natural numbers, and if so, which list of list of numbers?

Answer: YES, `[[1],[0]]` .

Comment: In principle, other readings of the tree are also possible (but not asked for) like: `[[1],1]` , `[`$\langle\langle$ nil.nil $\rangle$.nil $\rangle$, `1]` or `[[[0]],[0]]`
In these cases the elements of the list all have different "types" (note that "types" are in the "eye of the beholder" in `WHILE`).

4. Let a `WHILE`-program `p` be given as follows:

```
p read L {
 X:= hd L;
 Y:= hd tl L;
  while X {
      Y:= cons nil Y;
      X:= tl X
      }
}
write Y
```

(a) What does the program `p` compute for an input list `[M,N]` where `M` and `N` are both lists encoding natural numbers?

Answer: Addition of the two natural numbers encoded by `M` and `N`.

(b) What happens if we provide more lists than two in the input, i.e. input is of the form `[M,N,O,P]` where `M` , `N`, `O` , `P` are all lists encoding natural numbers?

Answer: Addition of the two natural numbers encoded by `M` and `N`, the other lists are ignored.

(c) What happens if we only provide one list in the input, i.e. input is of the form `[M]` where `M` is a list encoding a natural number?

Answer: Result is the number encoded by `M`.

(d) What does the program `p` compute for an input list `[M,N]` where `M` and `N` are not both lists encoding natural numbers?
Answer: The result is the concatenation of the list encoding

length(`M`) and list `N`. Note that every tree can be seen as list, so it is ok to apply *length*.

(e) Let `p2` be the program we get from `p` by changing the line `Y:= cons nil Y;` into `Y:= cons (hd X) Y;`
What does the program `p2` compute for an input list `[M,N]` if `M` and `N` are both lists, but of an unspecified type?

Answer: The program will return *reverse*(`M`) concatenated with `N`. It will ignore any extra elements in the input list. Any missing list will be replaced by the empty list. Note that every tree can be seen as list.

5. *(To be completed at home)* Write a *core* `WHILE`-program `isnumber` that returns ⌜true⌝ if the input tree encodes a natural number and ⌜false⌝ otherwise. *Core* means that you must not use any extensions (from Lecture 5). Test your program with `hwhile`.
The interpreter `hwhile` is available from the labs' Software Hub and binaries can be downloaded also from our Canvas page entitled "`WHILE` (Programs)". Read the short manual I put on canvas and familiarise yourself with `hwhile`.

Answer: Try this at home first. The answer will be published as a `.while` program in the programs section of our Canvas site in a few days.