Ülessuunatud jätkud

- Allasuunatud (ingl. *downward*) jätkud
  - jätku kasutatakse alamavaldises konteksti taastamiseks
  - kõik seni vaadeldud jätkude kasutused
- Ülessuunatud (ingl. *upward*) jätkud
  - jätku kasutatakse välises kontekstis
  - kaasprotseduurid (ingl. *coroutines*)

Ülessuunatud jätkud

"Katkestuspunktide" määramine:

```
define toplevel = letcont cont in cont
define resume = proc (x) error()
define break =
      proc (x)
        letcont cont
        in begin
             resume := cont;
             write(breakmsg);
             toplevel(x)
           end
```

Ülessuunatud jätkud

Näide:

```
--> "+(break(1), break(2))"
break:
1
--> "resume(5)"
break:
2
--> "resume(3)"
8
```

Kaasprotsessid

```
define example =
  proc ()
    letcont returncont
    in let co1 = ignored;
          co2 = ignored
       in begin
            co1 := makecoroutine(...);
            co2 := makecoroutine(...);
            co1(33)
          end
```

Kaasprotsessid

```
co1 := makecoroutine(proc (initval1)
                begin
                  write(1);
                  write(initval1);
                  initval1 := resume(co2, add1(initval1));
                  write(1);
                  write(initval1);
                  initval1 := resume(co2, add1(initval1));
                  returncont(initval1)
                end);
```

Kaasprotsessid

```
co2 := makecoroutine(proc (initval2)
                begin
                  write(2);
                  write(initval2);
                  initval2 := resume(co1, add1(initval2));
                  write(2);
                  write(initval2);
                  resume(co1, add1(initval2))
                end);
```

Kaasprotsessid

```
--> "example ()"
1
33
2
34
1
35
2
36
37
```

Kaasprotsessid

- Kirjutada protseduur, mis kontrollib kas etteantud kahes puus on lehtedes (samas järjekorras) ühesugused väärtused

```
--> "samefringe (list (list (1,2), 3),
                 list (list (1,2), 3))"
1
--> "samefringe (list (list (1,2), 3),
                 list (1, list (2,3)))"
1
--> "samefringe (list (list (1,2), 3),
                 list (1, list (2, list (2,3))))"
0
```

Kaasprotsessid

```
define makesfcoroutine = proc (driver, tree)
  makecoroutine(proc (initvalue)
      letrecproc traverse(tree) =
                    if pair(tree)
                    then begin traverse(car(tree));
                               if pair(cdr(tree))
                               then traverse(cdr(tree))
                               else noop
                         end
                    else resume(driver, tree)
          in begin traverse(tree);
                   resume(driver, 0)
             end)
```

Kaasprotsessid

```
define samefringe =
  proc (tree1, tree2)
    letcont returncont
    in let co1 = ignored;
           co2 = ignored;
           driver = ignored
       in begin
            driver := makecoroutine(...);
            co1 := makesfcoroutine(driver, tree1);
            co2 := makesfcoroutine(driver, tree2);
            driver(ignored)
          end
```

Kaasprotsessid

```
driver :=
  makecoroutine(proc (initvalue)
                    letrecproc
                       loop() =
                         let leaf1 = resume(co1, ignored);
                             leaf2 = resume(co2, ignored)
                         in if equal(leaf1, leaf2)
                            then if zero(leaf1)
                                 then returncont(1)
                                 else loop()
                            else returncont(0)
                    in loop());
```

Kaasprotsessid

Kaasprotsesside realisatsioon protseduurina “1se-klassi” jätkude abil:

```
define makecoroutine =
  proc (body)
    let lcs = ignored
    in letrecproc
         newcoroutine(value) = lcs(value);
         localresume(cont, value) = ...
       in letcont exit
          in begin
               body(localresume(exit, newcoroutine));
               error()
             end
```

Kaasprotsessid

Kaasprotsesside realisatsioon protseduurina “1se-klassi” jätkude abil (järg):

```
localresume(cont, value) =
        let val = letcont localcont
                  in begin
                       lcs := localcont;
                       cont(value)
                     end
        in begin
             resume := localresume;
             val
           end
```

Järgmiseks korraks

- Lugeda läbi EOPL ptk. 9.4
- "Mängida" interpretaatoriga `loeng22.ss`
  - laiendatud versioon eelmise loengu protseduursete jätkudega interpretaatorist `loeng21-1.ss`
  - lisatud lokaalsed (rekursiivsed) definitsioonid, omistamine, järjestikustamine ja mõned primitiivid