The files (`iccp2020-cps-LSYN^omega.hs` and `iccp2020-cps-LSYN.hs`) included in this directory contain Haskell (`www.haskell.org`) implementations of the denotational semantics presented in the IEEE ICCP 2020 submission entitled:

"Metric Semantics for Concurrent Languages Designed in Continuation-Passing Style"

File `iccp2020-cps-LSYN^omega.hs` implements the denotational semantics of language $\mathcal{L}_{SYN^\omega}$. File `iccp2020-cps-LSYN.hs` implements the denotational semantics of language $\mathcal{L}_{SYN}$. The semantic interpreters can be tested using function `run` as in the following example:

```
Main> run t1

...
```

(`Main>` is the GHC prompt.) $\mathcal{L}_{SYN^\omega}$ example programs t_1, t_2 and t_3 presented in the paper (Example 1) can be executed using function `main`.

```
Main> main

...
```

Various other $\mathcal{L}_{SYN^\omega}$ example programs are also available, and can be executed using functions `main2` and `main3`.

Note that $\mathcal{L}_{SYN}$ is a sublanguage of $\mathcal{L}_{SYN^\omega}$ (because $\mathcal{N} \subseteq J$; see Definition 11 and Section V-A). Due to space limitations, the paper presents no $\mathcal{L}_{SYN}$ example programs. However, the file `iccp2020-cps-LSYN.hs` contains a couple of $\mathcal{L}_{SYN}$ example programs, which can be tested using function `main`, as illustrated above.

Further explanations are provided as comments in the Haskell files `iccp2020-cps-LSYN^omega.hs` and `iccp2020-cps-LSYN.hs`.