Funcons-beta: Failing

The PLanCompS Project

`Funcons-beta/Computations/Abnormal/Failing/Failing.cbs`*

Failing

[*Datatype* failing
  Funcon failed
  Funcon finalise-failing
  Funcon fail
  Funcon else
  Funcon else-choice
  Funcon checked
  Funcon check-true]

Meta-variables T <: values

Datatype failing ::= failed

failed is a reason for abrupt termination.

Funcon finalise-failing(X :⇒ T) :⇒ T | null-type
  ⇝ finalise-abrupting(X)

finalise-failing(X) handles abrupt termination of X due to executing fail.

Funcon fail :⇒ empty-type
  ⇝ abrupt(failed)

fail abruptly terminates all enclosing computations until it is handled.

*Suggestions for improvement: `plancomps@gmail.com`.
Issues: `https://github.com/plancomps/CBS-beta/issues`.

Funcon $\mathsf{else}(\_ : \Rightarrow T, \_ : (\Rightarrow T)^+) : \Rightarrow T$

$\mathsf{else}(X_1, X_2, \cdots)$ executes the arguments in turn until either some Xi does *not* fail, or all arguments Xi have been executed. The last argument executed determines the result. $\mathsf{else}(X, Y)$ is associative, with unit fail.

Rule $$\frac{X \xrightarrow{\mathsf{abrupted}(\ )} X'}{\mathsf{else}(X, Y) \xrightarrow{\mathsf{abrupted}(\ )} \mathsf{else}(X', Y)}$$

Rule $$\frac{X \xrightarrow{\mathsf{abrupted}(\mathsf{failed})} \_}{\mathsf{else}(X, Y) \xrightarrow{\mathsf{abrupted}(\ )} Y}$$

Rule $$\frac{X \xrightarrow{\mathsf{abrupted}(V : \sim \mathsf{failing})} X'}{\mathsf{else}(X, Y) \xrightarrow{\mathsf{abrupted}(V)} \mathsf{else}(X', Y)}$$

Rule $\mathsf{else}(V : T, Y) \rightsquigarrow V$

Rule $\mathsf{else}(X, Y, Z^+) \rightsquigarrow \mathsf{else}(X, \mathsf{else}(Y, Z^+))$

Funcon $\mathsf{else\text{-}choice}(\_ : (\Rightarrow T)^+) : \Rightarrow T$

$\mathsf{else\text{-}choice}(X, \cdots)$ executes the arguments in any order until either some Xi does *not* fail, or all arguments Xi have been executed. The last argument executed determines the result. $\mathsf{else}(X, Y)$ is associative and commutative, with unit fail.

Rule $\mathsf{else\text{-}choice}(W^*, X, Y, Z^*) \rightsquigarrow \mathsf{choice}(\mathsf{else}(X, \mathsf{else\text{-}choice}(W^*, Y, Z^*), \mathsf{else}(Y, \mathsf{else\text{-}choice}(W^*, X, Z^*))))$

Rule $\mathsf{else\text{-}choice}(X) \rightsquigarrow X$

Funcon $\mathsf{check\text{-}true}(\_ : \mathsf{booleans}) : \Rightarrow \mathsf{null\text{-}type}$

Alias $\mathsf{check} = \mathsf{check\text{-}true}$

$\mathsf{check\text{-}true}(X)$ terminates normally if the value computed by X is true, and fails if it is false.

Rule $\mathsf{check\text{-}true}(\mathsf{true}) \rightsquigarrow \mathsf{null\text{-}value}$

Rule $\mathsf{check\text{-}true}(\mathsf{false}) \rightsquigarrow \mathsf{fail}$

Funcon checked(_ : $(T)^?$) : $\Rightarrow T$

checked(X) fails when X gives the empty sequence of values ( ), representing that an optional value has not been computed. It otherwise computes the same as X.

Rule checked($V : T$) $\rightsquigarrow V$

Rule checked( ) $\rightsquigarrow$ fail