Funcons-beta: Failing [*]

The PLanCompS Project

Failing.cbs | PLAIN | PRETTY

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($X :\Rightarrow T$) $:\Rightarrow T$ | null-type
$\rightsquigarrow$ finalise-abrupting(X)

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

Funcon fail : $\Rightarrow$ empty-type
$\rightsquigarrow$ abrupt(failed)

fail abruptly terminates all enclosing computations until it is handled.

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

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. else(X, Y) is associative, with unit fail.

[*] Suggestions for improvement: plancomps@gmail.com.
Reports of issues: https://github.com/plancomps/CBS-beta/issues.

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

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

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

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

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

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

else-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. else(X, Y) is associative and commutative, with unit fail.

```
Rule  else-choice(W*, X, Y, Z*) ⇝
        choice(
          else(
            X,
            else-choice(W*, Y, Z*),
            else(Y, else-choice(W*, X, Z*))))
Rule  else-choice(X) ⇝ X
```

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

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

check-true(X) terminates normally if the value computed by X is true, and fails if it is false.

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

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

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

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 $\text{checked}(V : T) \rightsquigarrow V$

Rule $\text{checked}(\ ) \rightsquigarrow \text{fail}$