

# Funcons-beta: Failing \*

The P<sub>L</sub>anCompS 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*  $T <: \text{values}$

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

---

\*Suggestions for improvement: [plancomps@gmail.com](mailto:plancomps@gmail.com).  
Reports of issues: <https://github.com/plancomps/CBS-beta/issues>.

$$\begin{array}{l}
\text{Rule} \quad \frac{X \xrightarrow{\text{abrupted}(\ )} X'}{\text{else}(X, Y) \xrightarrow{\text{abrupted}(\ )} \text{else}(X', Y)} \\
\text{Rule} \quad \frac{X \xrightarrow{\text{abrupted}(\text{failed})} -}{\text{else}(X, Y) \xrightarrow{\text{abrupted}(\ )} Y} \\
\text{Rule} \quad \frac{X \xrightarrow{\text{abrupted}(V:\sim \text{failing})} X'}{\text{else}(X, Y) \xrightarrow{\text{abrupted}(V)} \text{else}(X', Y)} \\
\text{Rule} \quad \text{else}(V : T, Y) \rightsquigarrow V \\
\text{Rule} \quad \text{else}(X, Y, Z^+) \rightsquigarrow \text{else}(X, \text{else}(Y, Z^+))
\end{array}$$

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

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

$$\begin{array}{l}
\text{Rule} \quad \text{else-choice}(W^*, X, Y, Z^*) \rightsquigarrow \\
\quad \text{choice}(\text{else}(X, \text{else-choice}(W^*, Y, Z^*), \text{else}(Y, \text{else-choice}(W^*, X, Z^*)))) \\
\text{Rule} \quad \text{else-choice}(X) \rightsquigarrow X
\end{array}$$

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

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

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

$$\begin{array}{l}
\text{Rule} \quad \text{check-true}(\text{true}) \rightsquigarrow \text{null-value} \\
\text{Rule} \quad \text{check-true}(\text{false}) \rightsquigarrow \text{fail}
\end{array}$$

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

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

$$\begin{array}{l}
\text{Rule} \quad \text{checked}(V : T) \rightsquigarrow V \\
\text{Rule} \quad \text{checked}(\ ) \rightsquigarrow \text{fail}
\end{array}$$