

# Funcons-beta: Throwing

The P<sub>Plan</sub>CompS Project

Funcons-beta/Computations/Abnormal/Throwing/Throwing.cbs\*

## Throwing

```
[ Datatype throwing
  Funcon thrown
  Funcon finalise-throwing
  Funcon throw
  Funcon handle-thrown
  Funcon handle-recursively
  Funcon catch-else-throw ]
```

Meta-variables  $R, S, T, T', T''$  <: values

Datatype **throwing** ::= **thrown**( $\_$  : values)

**thrown**( $V$ ) is a reason for abrupt termination.

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

**finalise-throwing**( $X$ ) handles abrupt termination of  $X$  due to executing **throw**( $V$ ).

Funcon **throw**( $V : T$ ) :  $\Rightarrow$  empty-type  
 $\rightsquigarrow$  **abrupt**(**thrown**( $V$ ))

**throw**( $V$ ) abruptly terminates all enclosing computations uTil it is handled.

---

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

Funcon **handle-thrown**( $\_ : T' \Rightarrow T, \_ : T'' \Rightarrow T$ ) :  $T' \Rightarrow T$

**handle-thrown**( $X, Y$ ) first evaluates  $X$ . If  $X$  terminates normally with value  $V$ , then  $V$  is returned and  $Y$  is ignored. If  $X$  terminates abruptly with a thrown eTity having value  $V$ , then  $Y$  is executed with  $V$  as **given** value.

**handle-thrown**( $X, Y$ ) is associative, with **throw(given)** as unit. **handle-thrown**( $X, \text{else}(Y, \text{throw(given)})$ ) ensures that if  $Y$  fails, the thrown value is re-thrown.

$$\begin{array}{l}
 \text{Rule } \frac{X \xrightarrow{\text{abrupted}(\ )} X'}{\text{handle-thrown}(X, Y) \xrightarrow{\text{abrupted}(\ )} \text{handle-thrown}(X', Y)} \\
 \text{Rule } \frac{X \xrightarrow{\text{abrupted}(\text{thrown}(V'' : \text{values}))} X'}{\text{handle-thrown}(X, Y) \xrightarrow{\text{abrupted}(\ )} \text{give}(V'', Y)} \\
 \text{Rule } \frac{X \xrightarrow{\text{abrupted}(V' : \sim \text{throwing})} X'}{\text{handle-thrown}(X, Y) \xrightarrow{\text{abrupted}(V')} \text{handle-thrown}(X', Y)} \\
 \text{Rule } \text{handle-thrown}(V : T, Y) \rightsquigarrow V
 \end{array}$$

Funcon **handle-recursively**( $X : S \Rightarrow T, Y : R \Rightarrow T$ ) :  $S \Rightarrow T$   
 $\rightsquigarrow$  **handle-thrown**( $X,$   
 $\quad \text{else}(\text{handle-recursively}(Y,$   
 $\quad \quad Y,$   
 $\quad \quad \text{throw(given)}))$ )

**handle-recursively**( $X, Y$ ) behaves similarly to **handle-thrown**( $X, Y$ ), except that another copy of the handler attempts to handle any values thrown by  $Y$ . Thus, many thrown values may get handled by the same handler.

Funcon **catch-else-throw**( $P : \text{values}, Y : \Rightarrow T$ ) :  $\Rightarrow T$   
 $\rightsquigarrow$  **else**(**case-match**( $P,$   
 $\quad Y,$   
 $\quad \text{throw(given)})$ )

**handle-thrown**( $X, \text{catch-else-throw}(P, Y)$ ) handles those values thrown by  $X$  that match pattern  $P$ . Other thrown values are re-thrown.