Transition System (a Coalgebra)

type $\mathit{TS}\ f\ s\quad = s \to f\ s$

Labelled Transition System

type $\mathit{LTS}\ a\ f\ s = \mathit{Map}\ a\ (\mathit{TS}\ f\ s)$ $\boxed{\text{map is a finite function}}$

$\mathit{alphabet} = M.\mathit{keys}$

infixl $9\ \backslash\$$

$(\backslash\$)\quad :: \mathit{Map}\ a\ b \to (a \to b)$

$f\ \backslash\$\ k = \mathit{guardFromJust}$ `"action not defined"` $(M.\mathit{lookup}\ k\ f)$

$\mathit{guardFromJust}\ \_\quad (\mathit{Just}\ x) = x$

$\mathit{guardFromJust}\ \mathit{err}\ \mathit{Nothing} = \mathit{error}\ \mathit{err}$

$$\begin{aligned}
&\mathit{runForgetful} :: (..., \mathit{Crush}\ f, \mathit{Functor}\ f) \Rightarrow \mathit{LTS}\ \ a\ f\ \ s \rightarrow [\,a\,] \rightarrow s \rightarrow \mathit{Set}\ \ s\\
&\mathit{runForgetful}\ \delta\ [\,]\qquad\quad\ = \mathit{singleton}\\
&\mathit{runForgetful}\ \delta\ (a : \mathit{as}) = \mathit{setjoin} \cdot \mathit{toSet} \cdot \mathit{fmap}\ (\mathit{runForgetful}\ \delta\ \ \mathit{as}) \cdot (\delta\ \backslash\$\ a)
\end{aligned}$$

$$\begin{aligned}
&\mathit{toSet} :: (..., \mathit{Crush}\ f) \Rightarrow f\ \ a \rightarrow \mathit{Set}\ \ a\\
&\mathit{toSet} = \mathit{crush}\ \ S.\mathit{insert}\ \emptyset
\end{aligned}$$

$$\mathit{setjoin} :: \mathit{Set}\ \ (\mathit{Set}\ \ a) \rightarrow \mathit{Set}\ \ a$$

$$\begin{aligned}
&\mathit{runForgetful} :: (..., \mathit{Crush}\ f, \mathit{Functor}\ f) \Rightarrow \mathit{LTS}\ \ a\ f\ \ s \to [\,a\,] \to s \to \mathit{Set}\ \ s \\
&\mathit{runForgetful}\ \delta\ [\,] \qquad\quad = \mathit{singleton} \\
&\mathit{runForgetful}\ \delta\ (a : \mathit{as}) = \mathit{setjoin} \cdot \mathit{toSet} \cdot \mathit{fmap}\ (\mathit{runForgetful}\ \delta\ \mathit{as}) \cdot (\delta \backslash\$\ a)
\end{aligned}$$

$$\boxed{\begin{aligned}
&\mathit{runPreserving} :: (..., \mathit{Functor}\ f) \Rightarrow \mathit{LTS}\ \ a\ f\ \ s \to [\,a\,] \to s \to \mathit{Star}\ f\ \ s \\
&\mathit{runPreserving}\ \delta\ [\,] \qquad\quad = \mathit{End} \\
&\mathit{runPreserving}\ \delta\ (a : \mathit{as}) = \mathit{Step} \cdot \mathit{fmap}\ (\mathit{runPreserving}\ \delta\ \mathit{as}) \cdot (\delta \backslash\$\ a)
\end{aligned}}$$

$$\mathbf{data}\ \mathit{Star}\ f\ \ s = \mathit{End}\ \ s \mid \mathit{Step}\ (f\ \ (\mathit{Star}\ f\ \ s))$$

$$\begin{aligned}
&\mathit{runForgetful} :: (\ldots, \mathit{Crush}\ f, \mathit{Functor}\ f) \Rightarrow \mathit{LTS}\ \ a\ f\ \ s \to [\,a\,] \to s \to \mathit{Set}\ s\\
&\mathit{runForgetful}\ \delta\ [\,] &&= \mathit{singleton}\\
&\mathit{runForgetful}\ \delta\ (a : as) &&= \mathit{setjoin} \cdot \mathit{toSet} \cdot \mathit{fmap}\ (\mathit{runForgetful}\ \delta\ \ as) \cdot (\delta\ \backslash\$\ a)
\end{aligned}$$

$$\begin{aligned}
&\mathit{runPreserving} :: (\ldots, \mathit{Functor}\ f) \Rightarrow \mathit{LTS}\ \ a\ f\ \ s \to [\,a\,] \to s \to \mathit{Star}\ f\ \ s\\
&\mathit{runPreserving}\ \delta\ [\,] &&= \mathit{End}\\
&\mathit{runPreserving}\ \delta\ (a : as) &&= \mathit{Step} \cdot \mathit{fmap}\ (\mathit{runPreserving}\ \delta\ \ as) \cdot (\delta\ \backslash\$\ a)
\end{aligned}$$

$$\boxed{\begin{aligned}
&\mathit{runInMonad} :: (\ldots, \mathit{Functor}\ f, \mathit{Monad}\ f) \Rightarrow \mathit{LTS}\ \ a\ f\ \ s \to [\,a\,] \to s \to f\ \ s\\
&\mathit{runInMonad}\ \delta\ [\,] &&= \mathit{return}\\
&\mathit{runInMonad}\ \delta\ (a : as) &&= \mathit{runInMonad}\ \delta\ \ as \bullet (\delta\ \backslash\$\ a)\\
& &&\mathit{or}\\
&\mathit{runInMonad}\ \delta\ (a : as) &&= \mathit{join} \cdot \mathit{runInMonad}\ \delta\ \ as \cdot (\delta\ \backslash\$\ a)
\end{aligned}}$$

```
bisimilar :: (Eq s, Ord a) ⇒ LTS a f s → s → s → Bool
bisimilar δ p q = runReader (bisim δ p q) [ ]


bisim :: (Eq s, Ord a) ⇒ LTS a f s → s → s → Reader [(s, s)] Bool
bisim δ p q = do stack ← ask
                 if p ≡ q ∨ (p, q) ∈ stack ∨ (q, p) ∈ stack
                   then return True
                   else liftM and $ mapM (bisimBy δ p q) (alphabet δ)
bisimBy :: (Eq s, Ord a) ⇒ LTS a f s → s → s → a → Reader [(s, s)] Bool
bisimBy δ p q a = let p′ = (δ \$ a) p
                      q′ = (δ \$ a) q
                  in local ((p, q):) $
                     liftM (maybe False and) $ fSafeZipWithM (bisim δ) p′ q′


fSafeZipWithM :: (a → b → m c) → f a → f b → m (Maybe (f c))
```