Tüübituletus

Avaldised

$$
\begin{aligned}
\textbf{type}\ \mathit{Var} &= \mathit{String} \\
\textbf{data}\ \mathit{Term} &= \mathit{Var}\ \ \mathit{Var} \\
&\mid \mathit{App}\ \ \mathit{Term}\ \ \mathit{Term} \\
&\mid \mathit{Lam}\ \ \mathit{Var}\ \ \ \mathit{Term} \\
&\mid \mathit{Let}\ \ \ \mathit{Var}\ \ \ \mathit{Term}\ \ \mathit{Term} \\
&\mid \mathit{Con}\ \ \mathit{Int} \\
&\mid \mathit{Pair}\ \ \mathit{Term}\ \ \mathit{Term} \\
&\mid \mathit{If}\ \ \ \ \mathit{Term}\ \ \mathit{Term}\ \ \mathit{Term}
\end{aligned}
$$

Tüübituletus

Tüübid

```
type TVar    = String
type TCon    = String
data Type    = TVar TVar
             | TCon TCon [Type]
tArrow t1 t2 = TCon "->" [t1,t2]
tPair t1 t2  = TCon "*" [t1,t2]
tBool        = TCon "Bool" []
tInt         = TCon "Int" []
```

Tüübituletus

Substitutsioonid

$$\begin{aligned}
&\textbf{type } TSubst = [(TVar, Type)] \\
&nullTS \;\; :: \; TSubst \\
&nullTS \;\; = [\,] \\
&(+\!\!\rightarrow) \;\; :: \; TVar \to Type \to TSubst \\
&x +\!\!\rightarrow t \;\; = [(x, t)] \\
&(@@) \;\; :: \; TSubst \to TSubst \to TSubst \\
&s1 \; @@ \; s2 = [(x, applyTS \; s1 \; t) \mid (x, t) \leftarrow s2] +\!\!+ s1
\end{aligned}$$

Substitutsiooni rakendamine

$$\begin{aligned}
&\textbf{class } Types \; t \; \textbf{where} \\
&\quad applyTS :: TSubst \to t \to t \\
&\quad tVars \;\; :: t \to [TVar]
\end{aligned}$$

Tüübituletus

Substitutsiooni rakendamine

```
instance Types Type where
  applyTS s (TVar x)     = case lookup x s of
                             Just t → t
                             Nothing → TVar x
  applyTS s (TCon c ts) = TCon c (map (applyTS s) ts)
  tVars (TVar x)     = [x]
  tVars (TCon c ts) = foldr union [] ∘ map tVars $ ts
instance Types a ⇒ Types [a] where
  applyTS s = map (applyTS s)
  tVars       = nub ∘ concat ∘ map tVars
```

Tüübituletus

Unifitseerimine

```
mgu :: Monad m => Type -> Type -> m TSubst
mgu (TVar x) t = varBind x t
mgu t (TVar x) = varBind x t
mgu (TCon c1 ts1) (TCon c2 ts2)
  | c1 == c2     = mguList nullTS ts1 ts2
  | otherwise    = fail "Types do not unify"

mguList :: Monad m =>
           TSubst -> [Type] -> [Type] -> m TSubst
mguList s0 (t1 : ts1) (t2 : ts2)
                  = do s1 <- mgu (applyTS s0 t1)
                                 (applyTS s0 t2)
                       mguList (s1 @@ s0) ts1 ts2
mguList s0 [] [] = return s0
```

Tüübituletus

Unifitseerimine

$$
\begin{aligned}
&varBind :: Monad\ m \Rightarrow TVar \rightarrow Type \rightarrow m\ TSubst \\
&varBind\ x\ (TVar\ y) \mid x \equiv y = return\ nullTS \\
&varBind\ x\ t \mid x \in tVars\ t \quad = fail\ \texttt{"Occurs check fails"} \\
&\qquad\qquad \mid otherwise \quad = return\ (x \mathbin{+\!\!>} t)
\end{aligned}
$$

Tüübituletus

Tüübiskeemid

$$\textbf{data}\ TScheme = ForAll\ [TVar]\ Type$$

$$\begin{aligned}&\textbf{instance}\ Types\ TScheme\ \textbf{where}\\ &\quad applyTS\ s\ (ForAll\ xs\ t) = ForAll\ xs\ (applyTS\ s\ t)\\ &\quad tVars\ (ForAll\ xs\ t) \quad\ = tVars\ t \setminus\setminus xs\end{aligned}$$

$$\begin{aligned}&toScheme :: Type \to TScheme\\ &toScheme\ t = ForAll\ [\,]\ t\end{aligned}$$

NB!

applyTS definitsioon eeldab, et skemaatilised muutujad oleksid kõigist teistest muutujatest erinevate nimedega!?

Tüübituletus

Eeldused

$$\textbf{data}\ Assumption = Var :>: TScheme$$

$$\begin{aligned}&\textbf{instance}\ Types\ Assumption\ \textbf{where}\\ &\quad applyTS\ s\ (x :>: sc) = x :>: applyTS\ s\ sc\\ &\quad tVars\ (x :>: sc) \quad\ = tVars\ sc\end{aligned}$$

Keskonnad

$$\textbf{type}\ TEnv = [Assumption]$$

$$\begin{aligned}&find :: Monad\ m \Rightarrow Var \to TEnv \to m\ TScheme\\ &find\ x\ [\,] = fail\ (\texttt{"Unbound identifier: "} +\!\!+ x)\\ &find\ x\ ((y :>: sc) : env)\ |\ x \equiv y \quad\ = return\ sc\\ &\qquad\qquad\qquad\qquad\ \ |\ otherwise = find\ x\ env\end{aligned}$$

$$\begin{aligned}&extEnv :: TEnv \to (Var, TScheme) \to TEnv\\ &extEnv\ env\ (x, sc) = (x :>: sc) : env\end{aligned}$$

Tüübituletus

Tüübituletusmonaad

$$
\begin{aligned}
&\textbf{newtype}\ TI\ a = TI\ (TSubst \to Int \to (a, TSubst, Int))\\
&\textbf{instance}\ Monad\ TI\ \textbf{where}\\
&\quad return\ x \quad\ = TI\ (\lambda s\ n \to (x, s, n))\\
&\quad (TI\ f) \gg\!= g = TI\ (\lambda s\ n \to \textbf{case}\ f\ s\ n\ \textbf{of}\\
&\qquad\qquad\qquad\qquad (x, s', n') \to \textbf{let}\ TI\ gx = g\ x\\
&\qquad\qquad\qquad\qquad\qquad\qquad \textbf{in}\ gx\ s'\ n')\\
&runTI :: TI\ a \to a\\
&runTI\ (TI\ f) = x\\
&\quad \textbf{where}\ (x, \_, \_) = f\ nullTS\ 0
\end{aligned}
$$

Tüübituletus

Tüübituletusmonaad

$$
\begin{aligned}
newTVar &:: TI\ TVar \\
newTVar &= TI\ (\lambda s\ n \rightarrow (\text{"a"} +\!\!+ show\ n, s, n+1)) \\
getSubst &:: TI\ TSubst \\
getSubst &= TI\ (\lambda s\ n \rightarrow (s, s, n)) \\
extSubst &:: TSubst \rightarrow TI\ () \\
extSubst\ u &= TI\ (\lambda s\ n \rightarrow ((), u\ @@\ s, n))
\end{aligned}
$$

Unifitseerimine

$$
\begin{aligned}
&unify :: Type \rightarrow Type \rightarrow TI\ () \\
&unify\ t1\ t2 = \mathbf{do}\ s \leftarrow getSubst \\
&\qquad\qquad\qquad u \leftarrow mgu\ (applyTS\ s\ t1)\ (applyTS\ s\ t2) \\
&\qquad\qquad\qquad extSubst\ u
\end{aligned}
$$

Tüübituletus

Instantsieerimine

$$
\begin{aligned}
&\mathit{freshInst} :: \mathit{TScheme} \to \mathit{TI}\ \mathit{Type} \\
&\mathit{freshInst}\ (\mathit{ForAll}\ \mathit{xs}\ t) \\
&\quad = \mathbf{do}\ s \leftarrow \mathit{mapM}\ (\lambda x \to \mathit{newTVar} \gg\!= \lambda y \to \\
&\qquad\qquad\qquad\qquad\qquad\qquad \mathit{return}\ (x, \mathit{TVar}\ y))\ \mathit{xs} \\
&\qquad\quad \mathit{return}\ (\mathit{applyTS}\ s\ t)
\end{aligned}
$$

Tüübituletus

Kvantifitseerimine

```
mkScheme :: Type → TEnv → TI TScheme
mkScheme t env
  = do s ← getSubst
       let t'    = applyTS s t
       let scvs = tVars t' \\ tVars env
       xs ← mapM (λx → newTVar) scvs
       let s'    = zipWith (λv x → (v, TVar x)) scvs xs
       return (ForAll xs (applyTS s' t'))
```

Tüübituletus

Tuletusreeglid

```
inferT :: Term → TEnv → TI Type
inferT (Con n) env       = return tInt
inferT (Pair e1 e2) env  = do t1 ← inferT e1 env
                              t2 ← inferT e2 env
                              return (tPair t1 t2)
inferT (If e0 e1 e2) env = do t0 ← inferT e0 env
                              unify t0 tBool
                              t1 ← inferT e1 env
                              t2 ← inferT e2 env
                              unify t1 t2
                              return t1
inferT (Var x) env       = do sc ← find x env
                              freshInst sc
```

Tüübituletus

Tuletusreeglid

```
inferT (App e1 e2) env
  = do t1 ← inferT e1 env
       t2 ← inferT e2 env
       t ← newTVar >>= λx → return (TVar x)
       unify (t2 `tArrow` t) t1
       return t
inferT (Lam x e1) env
  = do t0 ← newTVar >>= λx → return (TVar x)
       t1 ← inferT e1 (extEnv env (x, toScheme t0))
       return (t0 `tArrow` t1)
inferT (Let x e1 e0) env
  = do t1 ← inferT e1 env
       sc ← mkScheme t1 env
       inferT e0 (extEnv env (x, sc))
```

Tüübituletus

Algkeskond ja kogu avaldise tüübi leidmine

```
env0 = [ "true"  :>: toScheme tBool
       , "false" :>: toScheme tBool
       , "eq"    :>: toScheme (tArrow tInt (tArrow tInt tBool))
       , "add"   :>: toScheme (tArrow tInt (tArrow tInt tInt))
       ]
typeInf0 :: Term → TI Type
typeInf0 e = do t ← inferT e env0
                s ← getSubst
                return (applyTS s t)
```