```
(*****************************************************************************************************************************)
(* HaskWeakVars: types and variables for HaskWeak                                                                            *)
(*****************************************************************************************************************************)

Generalizable All Variables.
Require Import Preamble.
Require Import Coq.Strings.String.
Require Import Coq.Lists.List.
Require Import General.
Require Import HaskKinds.
Require Import HaskLiteralsAndTyCons.
Require Import HaskCoreVars.
Require Import HaskCoreTypes.
Require Import HaskWeakTypes.

(* a WeakExprVar just wraps a CoreVar and tags it with the type of its value *)
Inductive WeakExprVar := weakExprVar : CoreVar -> WeakType -> WeakExprVar.

(* a WeakVar is one of the three sorts *)
Inductive WeakVar : Type :=
| WExprVar : WeakExprVar -> WeakVar
| WTypeVar : WeakTypeVar -> WeakVar
| WCoerVar : WeakCoerVar -> WeakVar.
  Coercion WExprVar : WeakExprVar >-> WeakVar.
  Coercion WTypeVar : WeakTypeVar >-> WeakVar.
  Coercion WCoerVar : WeakCoerVar >-> WeakVar.

Definition weakTypeVarToKind (tv:WeakTypeVar) : Kind :=
  match tv with weakTypeVar _ k => k end.
  Coercion weakTypeVarToKind : WeakTypeVar >-> Kind.

Definition weakVarToCoreVar (wv:WeakVar) : CoreVar :=
  match wv with
  | WExprVar (weakExprVar v _    ) => v
  | WTypeVar (weakTypeVar v _    ) => v
  | WCoerVar (weakCoerVar v _ _ _) => v
 end.
 Coercion weakVarToCoreVar : WeakVar >-> CoreVar.

Definition haskLiteralToWeakType lit : WeakType :=
  WTyCon (haskLiteralToTyCon lit).
```

```
  Coercion haskLiteralToWeakType : HaskLiteral >-> WeakType.

Variable coreVarToWeakVar  : CoreVar  -> WeakVar.   Extract Inlined Constant coreVarToWeakVar    => "coreVarToWeakVar".
Variable getTyConTyVars_   : CoreTyCon   -> list CoreVar.  Extract Inlined Constant getTyConTyVars_   => "getTyConTyVars".
Definition tyConTyVars (tc:CoreTyCon) :=
  filter (map (fun x => match coreVarToWeakVar x with WTypeVar v => Some v | _ => None end) (getTyConTyVars_ tc)).
  Opaque tyConTyVars.
Definition tyConKind (tc:TyCon) : list Kind := map (fun (x:WeakTypeVar) => x:Kind) (tyConTyVars tc).

Variable rawTyFunKind : CoreTyCon -> Kind. Extract Inlined Constant rawTyFunKind => "(coreKindToKind . TyCon.tyConKind)".

Definition tyFunKind (tc:TyFun) : ((list Kind) * Kind) :=
  splitKind (rawTyFunKind tc).

Instance WeakVarToString : ToString WeakVar :=
  { toString := fun x => toString (weakVarToCoreVar x) }.
```