```
(*********************************************************************************************************************)
(* HaskCoreVars: basically GHC's Var.Var imported into Coqland                                                        *)
(*********************************************************************************************************************)

Generalizable All Variables.
Require Import Preamble.
Require Import General.
Require Import Coq.Strings.String.
Require Import HaskLiteralsAndTyCons.

(* GHC uses a single type for expression variables, type variables, and coercion variables; this is that type *)
Variable CoreVar            : Type.                                               Extract Inlined Constant CoreVar    => "Var.Var".
Variable coreVar_eq         : forall (a b:CoreVar), sumbool (a=b) (not (a=b)).    Extract Inlined Constant coreVar_eq => "(==)".
Variable coreVarToString    : CoreVar      -> string.             Extract Inlined Constant coreVarToString => "outputableToString".
Instance CoreVarEqDecidable : EqDecidable CoreVar := { eqd_dec  := coreVar_eq      }.
Instance CoreVarToString    : ToString CoreVar    := { toString := coreVarToString }.

Variable CoreTyCon       : Type.                      Extract Inlined Constant CoreTyCon            => "TyCon.TyCon".

(* because Haskell's 3-tuples (triples) are distinct from both ((x,y),z) and (x,(y,z)), we need a new type: *)
Inductive triple {A B C:Type} :=
| mkTriple : A -> B -> C -> triple.
Notation "a ** b ** c" := (mkTriple a b c) (at level 20).
Extract Inductive triple => "(,,)" [ "(,,)" ].

Inductive CoreAltCon :=
| DataAlt : CoreDataCon -> CoreAltCon
| LitAlt  : HaskLiteral -> CoreAltCon
| DEFAULT :                CoreAltCon.
Extract Inductive CoreAltCon =>
  "CoreSyn.AltCon" [ "CoreSyn.DataAlt" "CoreSyn.LitAlt" "CoreSyn.DEFAULT" ].
```