1 Intro

I need to prove that haskell types and terms that I expose wouldn't break the system. It means two things:

1. Types preserve the same set of invariants
2. Terms have the same interface: any combination of `APPLY` that can be used(ignoring types) to original term must be usable with generated one; and primitives(numbers, strings, ... and their ops) are the same.

2 Preserving type invariants

Conversion for kinds:

$$KT[\![Kind]\!] = HaskellKind$$

$$\begin{aligned}
KT[\![Set]\!] &= * \\
KT[\![Set_0]\!] &= * \\
KT[\![Kind_1 \to Kind_2]\!] &= KT[\![Kind_1]\!] \to KT[\![Kind_2]\!] \\
KT[\![_]\!] &= \bot
\end{aligned}$$

Conversion for types:

$$TT[\![AgdaType]\!](Context) = HaskellType$$

$$\begin{aligned}
TT[\![A\ args\ldots]\!](\Gamma) &= a\ TT[\![args\ldots]\!](\Gamma), \quad (A \mapsto a) \in \Gamma \\
TT[\![CT\ args\ldots]\!](\Gamma) &= CT\ TT[\![args\ldots]\!](\Gamma), \quad CT \text{ is a } \texttt{COMPILED_TYPE}, \texttt{EXPORT} \text{ or a primitive postulate} \\
TT[\![(A : Kind) \to T]\!](\Gamma) &= \forall (a :: KT[\![Kind]\!]).\ TT[\![T]\!](\Gamma \cup (A \mapsto a)) \\
TT[\![(x : T_1) \to T_2]\!](\Gamma) &= TT[\![T_1]\!](\Gamma) \to TT[\![T_2]\!](\Gamma), \quad x \notin freevars(T_2) \\
TT[\![(x : T_1,\ T_2)]\!](\Gamma) &= (TT[\![T_1]\!](\Gamma),\ TT[\![T_2]\!](\Gamma)), \quad x \notin freevars(T_2) \\
TT[\![_]\!](\Gamma) &= \bot
\end{aligned}$$

Two things to watch for:

- `newtype` wrappers in the first case
- The third case

Every other case is exactly the same.

TODO:

3 Preserving term interface

Conversion for terms:

$$\begin{aligned}
Wrap[\![AgdaType]\!](MAlonzoTerm) &= MyTerm \\
Unwrap[\![AgdaType]\!](MyTerm) &= MAlonzoTerm
\end{aligned}$$

Both are only valid when $TT[\![AgdaType]\!](\varnothing) \neq \bot$

$$\begin{aligned}
Wrap[\![A\ args\ldots]\!](term) &= \texttt{unsafeCoerce}\ term \\
Wrap[\![(A : Kind) \to T]\!](term) &= Wrap[\![T]\!](term\ ()) \\
Wrap[\![(x : T_1) \to T_2]\!](term) &= \lambda x.\ Wrap[\![T_2]\!](term\ Unwrap[\![T_1]\!](x)) \\
Wrap[\![(x : T_1,\ T_2)]\!]((term_1,\ term_2)) &= (Wrap[\![T_1]\!](term_1),\ Wrap[\![T_2]\!](term_2)) \\
Wrap[\![_]\!](term) &= \bot
\end{aligned}$$

$$\begin{aligned}
Unwrap[\![A\ args\ldots]\!](term) &= \texttt{unsafeCoerce}\ term \\
Unwrap[\![(A : Kind) \to T]\!](term) &= Unwrap[\![T]\!](\lambda_.\ term) \\
Unwrap[\![(x : T_1) \to T_2]\!](term) &= \lambda x.\ Unwrap[\![T_2]\!](term\ Wrap[\![T_1]\!](x)) \\
Unwrap[\![(x : T_1,\ T_2)]\!]((term_1,\ term_2)) &= (Unwrap[\![T_1]\!](term_1),\ Unwrap[\![T_2]\!](term_2)) \\
Unwrap[\![_]\!](term) &= \bot
\end{aligned}$$

`unsafeCoerce` is legal because it's either:

- The same term(when its type is a type variable)
- A newtype around MAlonzo generated type
- A primitive