Let $p$ be a user's binary variable, and let $m$ be an implementer's natural variable. The operations allow the user to assign a value $n$ to the implementer's variable, and to test whether the implementer's variable is a prime number.

\[
\text{assign } n \quad = \quad m := n \\
\text{check } \quad = \quad p := \text{prime } m
\]

assuming \textit{prime} is suitably defined. If \textit{prime} is an expensive function, and the \textit{check} operation is more frequent than the \textit{assign} operation, we can improve the solution by making \textit{check} less expensive even if that makes \textit{assign} more expensive. Using data transformation, make this improvement.

§

I replace the implementer's natural variable $m$ by a new implementer's binary variable $q$. The data transformer is

\[
q = \text{prime } m
\]

We have to check that this is a data transformer.

\[
\forall q. \exists m. q = \text{prime } m
\]

\[
\equiv (\exists m. \top = \text{prime } m) \land (\exists m. \bot = \text{prime } m)
\]

\[
\iff (\top = \text{prime } 2) \land (\bot = \text{prime } 3)
\]

\[
\equiv \top
\]

Using this transformer, \textit{assign} $n$ is transformed to

\[
\forall m. q = \text{prime } m \Rightarrow \exists m'. q' = \text{prime } m' \land (m := n)
\]

\[
\equiv \forall m. q = \text{prime } m \Rightarrow \exists m'. q' = \text{prime } m' \land m' = n \land p' = p
\]

\[
\equiv \forall m. q = \text{prime } m \Rightarrow q' = \text{prime } n \land p' = p
\]

\[
\equiv q' = \text{prime } n \land p' = p
\]

Using this transformer, \textit{check} is transformed to

\[
\forall m. q = \text{prime } m \Rightarrow \exists m'. q' = \text{prime } m' \land (p := \text{prime } m)
\]

\[
\equiv \forall m. q = \text{prime } m \Rightarrow \exists m'. q' = \text{prime } m' \land m' = m \land p' = \text{prime } m
\]

\[
\equiv \forall m. q = \text{prime } m \Rightarrow q' = \text{prime } m \land p' = \text{prime } m
\]

\[
\equiv q' = q \land p' = q
\]

\[
\equiv p := q
\]