A desugaring of Scheme `cond` to V:

$$\begin{aligned} \texttt{(cond)} &\triangleq \texttt{wrong} \\ \texttt{(cond [}e_g\texttt{,}\ e_a\texttt{] ...)} &\triangleq \texttt{if}\ e_g\ \texttt{then}\ e_a\ \texttt{else (cond ...)} \end{aligned}$$

where e_g is a *guarded-exp* and e_a is an *exp* on the right-hand side.
This translation desugars `cond` into the Verse `if-then-else` form, which itself is syntactic sugar for **one**.

Here is a translation from `cond` to **one**.

$$\texttt{(cond)} \triangleq \texttt{wrong}$$

$$\texttt{(cond [}e_{g1}\ e_{a1}\texttt{] [}e_{g2}\ e_{a2}\texttt{] }\ldots\texttt{[}e_{gn}\ e_{an}\texttt{])} \triangleq$$

$$(\textbf{one}\ \{(e_{g1};\ \lambda\langle\rangle.\ e_{a1}) \mid (e_{g2};\ \lambda\langle\rangle.\ e_{a2}) \mid \ldots \mid (e_{gn};\ \lambda\langle\rangle.\ e_{an}) \mid \textbf{wrong}\})\langle\rangle$$

Question: is it the above, or is the **wrong** in a lambda, like

$$(\textbf{one}\ \{(e_{g1};\ \lambda\langle\rangle.\ e_{a1}) \mid (e_{g2};\ \lambda\langle\rangle.\ e_{a2}) \mid \ldots \mid (e_{gn};\ \lambda\langle\rangle.\ e_{an}) \mid \lambda\langle\rangle.\ \textbf{wrong}\})\langle\rangle$$

?
Either way, getting to the last branch produces wrong- if reaching **wrong** is a run-time error, it needen't be in the lambda, but if it's any sort of value, maybe evaluating it with he same semantics as the rest of the branches, as the result of a applying the returned lambda to $\langle\rangle$, could be best.

Also, I suspect there's a better symbol to use than $\triangleq$ for translations. What have you used?