Agda is an actively developed dependently typed programming language. Its types can directly depend on values: it is, for instance, possible to define a function returning the \( n \)-th element of a list so that the typechecker itself guarantees the list to have at least \( n \) elements.

data \( \text{Vec} \, (A : \text{Set}) : \mathbb{N} \rightarrow \text{Set} \) where
\[
\text{nil} : \text{Vec} \, A \, \text{zero} \quad \text{cons} : \{n : \mathbb{N}\} \rightarrow A \rightarrow \text{Vec} \, A \, n \rightarrow \text{Vec} \, A \, (\text{succ} \, n)
\]

data \( \text{Fin} : \mathbb{N} \rightarrow \text{Set} \) where
\[
\text{fzero} : \{n : \mathbb{N}\} \rightarrow \text{Fin} \, (\text{succ} \, n) \\
\text{fsucc} : \{n : \mathbb{N}\} \rightarrow \text{Fin} \, n \rightarrow \text{Fin} \, (\text{succ} \, n)
\]

\[\_ : \{A : \text{Set}\} \quad \{n : \mathbb{N}\} \quad \text{vec} \quad \text{cons} \quad \text{fzero} \quad \text{fsucc} \quad \text{fzero} \quad \text{fsucc} \] = \text{vec} \\
\[\text{vec} \quad \text{cons} \quad \text{fzero} \quad \text{fsucc} \quad \text{fzero} \quad \text{fsucc} \] = \text{vec} \\
\[\text{vec} \quad \text{cons} \quad \text{fzero} \quad \text{fsucc} \quad \text{fzero} \quad \text{fsucc} \] = \text{vec} \\
\[\text{vec} \quad \text{cons} \quad \text{fzero} \quad \text{fsucc} \quad \text{fzero} \quad \text{fsucc} \] = \text{vec} \\

Thanks to its rich and expressive type system, Agda can also serve as an interactive theorem prover. Types correspond to logical formulae whereas values represent formal proofs of their type/formula.

**K Framework**

\( K \) is a semantic framework in which formal semantics of programming languages can be specified in terms of rewriting rules and data configurations. It provides a variety of generic, practical tools that can be used with any language defined in \( K \), such as parsers, interpreters, symbolic execution engines, semantic debuggers, test-case generators, state-space explorers and model checkers. Immediate availability of these tools makes \( K \) specifications genuinely executable.

Several real-world languages have been already defined in \( K \), including C, Java, Python and Javascript.

\[
\text{rule } \text{bind} = \Pi
\]

\[
\begin{array}{c}
\text{saveCtx} (\Gamma) \leadsto \text{loadCtx} \\
\text{ctx}
\end{array}
\]

\[\Gamma \vdash e_1 \downarrow S_1 \leadsto T_1 \\
\Gamma, x : T_1 \vdash e_2 \downarrow S_2 \leadsto T_2 \\
\text{set}_{\text{whnf}} : S_1 \leadsto \text{Set}_a \\
\text{set}_{\text{whnf}} : S_2 \leadsto \text{Set}_a \\
\Gamma \vdash (x : e_1) \downarrow e_2 \downarrow \text{set}_{\text{whnf}} \leadsto (x : T_1) \rightarrow T_2
\]