Hazelnut: A Bidirectionally Typed Structure Editor Calculus

Cyrus Omar Carnegie Mellon University
Ian Voysey Carnegie Mellon University
Michael Hilton Oregon State University
Jonathan Aldrich Carnegie Mellon University
Matthew Hammer University of Colorado Boulder

POPL 2017

Q: What is it that program editors reason about?

Q: What is it that textual program editors reason about?

```
fun summary_stats(m : matrix) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m,
    median =
```

syntactically malformed program text

Syntactic error recovery heuristics

```
fun summary_stats(m : matrix) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, ▢),
    median = ▢ }
```

syntactically malformed program text → term with holes

[Kats et al., OOPSLA 2009; Amorim et al., SLE 2016]

Syntactic structure editors

```
fun summary_stats(m : matrix) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, ▢),
    median = ▢ }
```

~~syntactically malformed program text~~ → term with holes

[Teitelbaum and Reps, Comm. ACM 1981; many others since]

Q: How to **reason statically** about terms with holes?

```
fun summary_stats(m : matrix) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, ▢),
    median = ▢ }
```

Q: How to **reason statically** about terms with holes?

What **type** is synthesized for the function as a whole?

```
fun summary_stats(m : matrix) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, ▢),
    median = ▢ }
```

Q: How to **reason statically** about terms with holes?

What **type** is synthesized for the function as a whole?

```
matrix →
  { mean   : vec,
    std    : vec,
    median : ▢ }
```

```
fun summary_stats(m : matrix) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, ▢),
    median = ▢ }
```

Q: How to **reason statically** about terms with type errors?

What **type** is synthesized for the function as a whole?

```
fun summary_stats(m : matrix) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, "oops"),
    median = ▢ }
```

Q: How to **reason statically** about terms with type errors?

What **type** is synthesized for the function as a whole?

```
fun summary_stats(m : matrix) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, "oops"),
    median = ▢ }
```

Reify type inconsistencies as <u>non-empty</u> holes!

Q: How to **reason statically** about terms with type errors?

```
matrix →
  { mean   : vec,
    std    : vec,
    median : ▢ }
```

What **type** is synthesized for the function as a whole?

```
fun summary_stats(m : matrix) =
  { mean   = stats.mean(m, ColumnWise),
    std    = stats.std(m, “oops”),
    median = ▢ }
```

Reify type inconsistencies as <u>non-empty</u> holes!

Contribution 1: A **static semantics** for lambda terms with holes

$$
\begin{array}{llll}
\mathsf{HTyp} & \dot{\tau} & ::= & (\dot{\tau} \to \dot{\tau}) \mid \texttt{num} \mid (\!|\,|\!) \\
\mathsf{HExp} & \dot{e} & ::= & x \mid (\lambda x.\dot{e}) \mid \dot{e}(\dot{e}) \mid \underline{n} \mid (\dot{e} + \dot{e}) \mid \dot{e} : \dot{\tau} \mid (\!|\,|\!) \mid (\!|\dot{e}|\!)
\end{array}
$$

Contribution 1: A **static semantics** for lambda terms with holes

$$\begin{array}{llll} \mathsf{HTyp} & \dot\tau & ::= & (\dot\tau \to \dot\tau) \mid \mathtt{num} \mid (\!|\!) \\ \mathsf{HExp} & \dot e & ::= & x \mid (\lambda x.\dot e) \mid \dot e(\dot e) \mid \underline{n} \mid (\dot e + \dot e) \mid \dot e : \dot\tau \mid (\!|\!) \mid (\!|\dot e|\!) \end{array}$$

$\boxed{\dot\Gamma \vdash \dot e \Rightarrow \dot\tau}$ $\dot e$ synthesizes $\dot\tau$

$\boxed{\dot\Gamma \vdash \dot e \Leftarrow \dot\tau}$ $\dot e$ analyzes against $\dot\tau$

Contribution 1: A **static semantics** for lambda terms with holes

$$
\begin{array}{llll}
\mathsf{HTyp} & \dot\tau & ::= & (\dot\tau \to \dot\tau) \mid \mathtt{num} \mid (\!|\,|\!) \\
\mathsf{HExp} & \dot e & ::= & x \mid (\lambda x.\dot e) \mid \dot e(\dot e) \mid \underline{n} \mid (\dot e + \dot e) \mid \dot e : \dot\tau \mid (\!|\,|\!) \mid (\!|\dot e|\!)
\end{array}
$$

$\boxed{\dot\Gamma \vdash \dot e \Rightarrow \dot\tau}$ $\dot e$ synthesizes $\dot\tau$

$\boxed{\dot\Gamma \vdash \dot e \Leftarrow \dot\tau}$ $\dot e$ analyzes against $\dot\tau$

...

$$\frac{}{\dot\Gamma \vdash (\!|\,|\!) \Rightarrow (\!|\,|\!)}$$

$$\frac{\dot\Gamma \vdash \dot e \Rightarrow \dot\tau}{\dot\Gamma \vdash (\!|\dot e|\!) \Rightarrow (\!|\,|\!)}$$

Contribution 1: A **static semantics** for lambda terms with holes

$$\begin{array}{llll} \text{HTyp} & \dot{\tau} & ::= & (\dot{\tau} \to \dot{\tau}) \mid \texttt{num} \mid (\!|\,|\!) \\ \text{HExp} & \dot{e} & ::= & x \mid (\lambda x.\dot{e}) \mid \dot{e}(\dot{e}) \mid \underline{n} \mid (\dot{e} + \dot{e}) \mid \dot{e} : \dot{\tau} \mid (\!|\,|\!) \mid (\!|\dot{e}|\!) \end{array}$$

$\boxed{\dot{\Gamma} \vdash \dot{e} \Rightarrow \dot{\tau}}$ $\dot{e}$ synthesizes $\dot{\tau}$

...

$$\frac{}{\dot{\Gamma} \vdash (\!|\,|\!) \Rightarrow (\!|\,|\!)}$$

$$\frac{\dot{\Gamma} \vdash \dot{e} \Rightarrow \dot{\tau}}{\dot{\Gamma} \vdash (\!|\dot{e}|\!) \Rightarrow (\!|\,|\!)}$$

$\boxed{\dot{\Gamma} \vdash \dot{e} \Leftarrow \dot{\tau}}$ $\dot{e}$ analyzes against $\dot{\tau}$

...

$$\frac{\dot{\Gamma} \vdash \dot{e} \Rightarrow \dot{\tau}' \qquad \dot{\tau} \sim \dot{\tau}'}{\dot{\Gamma} \vdash \dot{e} \Leftarrow \dot{\tau}}$$

Contribution 1: A **static semantics** for lambda terms with holes

$$\begin{array}{llll} \text{HTyp} & \dot\tau & ::= & (\dot\tau \to \dot\tau) \mid \texttt{num} \mid (\!|\!) \\ \text{HExp} & \dot e & ::= & x \mid (\lambda x.\dot e) \mid \dot e(\dot e) \mid \underline{n} \mid (\dot e + \dot e) \mid \dot e : \dot\tau \mid (\!|\!) \mid (\!|\dot e|\!) \end{array}$$

$\boxed{\dot\Gamma \vdash \dot e \Rightarrow \dot\tau}$ $\dot e$ synthesizes $\dot\tau$

...

$$\frac{}{\dot\Gamma \vdash (\!|\!) \Rightarrow (\!|\!)}$$

$$\frac{\dot\Gamma \vdash \dot e \Rightarrow \dot\tau}{\dot\Gamma \vdash (\!|\dot e|\!) \Rightarrow (\!|\!)}$$

$\boxed{\dot\Gamma \vdash \dot e \Leftarrow \dot\tau}$ $\dot e$ analyzes against $\dot\tau$

...

$$\frac{\dot\Gamma \vdash \dot e \Rightarrow \dot\tau' \qquad \dot\tau \sim \dot\tau'}{\dot\Gamma \vdash \dot e \Leftarrow \dot\tau}$$

$\boxed{\dot\tau \sim \dot\tau'}$ $\dot\tau$ and $\dot\tau'$ are consistent

$$\frac{}{(\!|\!) \sim \dot\tau} \qquad \frac{}{\dot\tau \sim (\!|\!)} \qquad \frac{}{\dot\tau \sim \dot\tau} \qquad \frac{\dot\tau_1 \sim \dot\tau_1' \qquad \dot\tau_2 \sim \dot\tau_2'}{(\dot\tau_1 \to \dot\tau_2) \sim (\dot\tau_1' \to \dot\tau_2')}$$

Contribution 1: A **static semantics** for lambda terms with holes

$$\begin{array}{llll} \text{HTyp} & \dot\tau & ::= & (\dot\tau \to \dot\tau) \mid \texttt{num} \mid (\!|\,|\!) \\ \text{HExp} & \dot e & ::= & x \mid (\lambda x.\dot e) \mid \dot e(\dot e) \mid \underline{n} \mid (\dot e + \dot e) \mid \dot e : \dot\tau \mid (\!|\,|\!) \mid (\!|\dot e|\!) \end{array}$$

$\boxed{\dot\Gamma \vdash \dot e \Rightarrow \dot\tau}$ $\dot e$ synthesizes $\dot\tau$

...

$$\frac{}{\dot\Gamma \vdash (\!|\,|\!) \Rightarrow (\!|\,|\!)}$$

$$\frac{\dot\Gamma \vdash \dot e \Rightarrow \dot\tau}{\dot\Gamma \vdash (\!|\dot e|\!) \Rightarrow (\!|\,|\!)}$$

$\boxed{\dot\Gamma \vdash \dot e \Leftarrow \dot\tau}$ $\dot e$ analyzes against $\dot\tau$

...

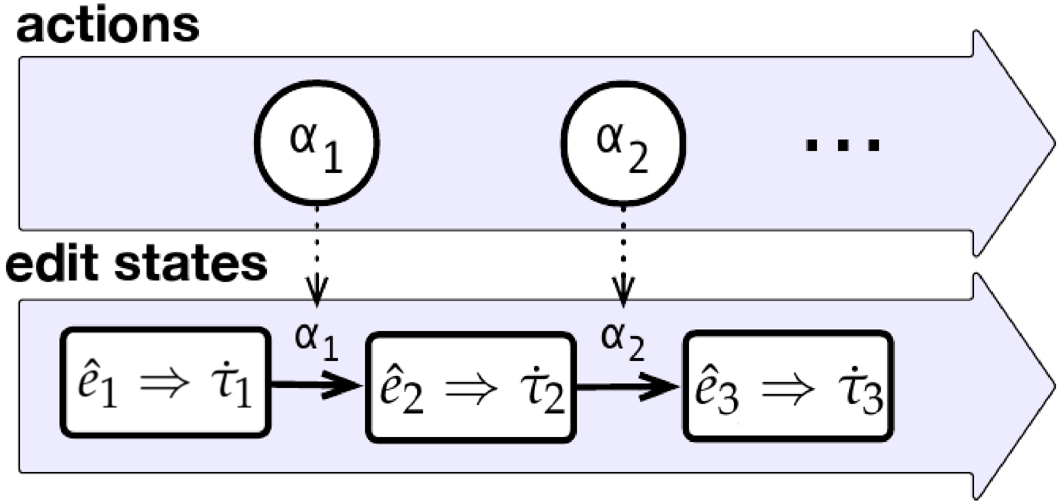
$$\frac{\dot\Gamma \vdash \dot e \Rightarrow \dot\tau' \qquad \dot\tau \sim \dot\tau'}{\dot\Gamma \vdash \dot e \Leftarrow \dot\tau}$$

$\boxed{\dot\tau \sim \dot\tau'}$ $\dot\tau$ and $\dot\tau'$ are consistent

$$\frac{}{(\!|\,|\!) \sim \dot\tau} \qquad \frac{}{\dot\tau \sim (\!|\,|\!)} \qquad \frac{}{\dot\tau \sim \dot\tau} \qquad \frac{\dot\tau_1 \sim \dot\tau_1' \qquad \dot\tau_2 \sim \dot\tau_2'}{(\dot\tau_1 \to \dot\tau_2) \sim (\dot\tau_1' \to \dot\tau_2')}$$

coincides with **gradual typing**
[Siek and Taha, 2006]

Contribution 2: A typed edit action semantics



Live Demo

See http://hazelgrove.org/

Contribution 2: A typed edit action semantics

$$
\begin{array}{llcl}
\mathsf{ZTyp} & \hat{\tau} & ::= & \triangleright\dot{\tau}\triangleleft \mid (\hat{\tau} \rightarrow \dot{\tau}) \mid (\dot{\tau} \rightarrow \hat{\tau}) \\
\mathsf{ZExp} & \hat{e} & ::= & \triangleright\dot{e}\triangleleft \mid (\lambda x.\hat{e}) \mid \hat{e}(\dot{e}) \mid \dot{e}(\hat{e}) \mid (\hat{e} + \dot{e}) \mid (\dot{e} + \hat{e}) \\
 & & \mid & \hat{e} : \dot{\tau} \mid \dot{e} : \hat{\tau} \mid (\!|\hat{e}|\!)
\end{array}
$$

Contribution 2: A typed edit action semantics

$$\begin{array}{llcl}
\mathsf{ZTyp} & \hat\tau & ::= & \triangleright\dot\tau\triangleleft \mid (\hat\tau \rightarrow \dot\tau) \mid (\dot\tau \rightarrow \hat\tau) \\
\mathsf{ZExp} & \hat e & ::= & \triangleright\dot e\triangleleft \mid (\lambda x.\hat e) \mid \hat e(\dot e) \mid \dot e(\hat e) \mid (\hat e + \dot e) \mid (\dot e + \hat e) \\
 & & \mid & \hat e : \dot\tau \mid \dot e : \hat\tau \mid (\!|\hat e|\!)
\end{array}$$

$$\begin{array}{llcl}
\mathsf{Action} & \alpha & ::= & \mathtt{move}\ \delta \mid \mathtt{construct}\ \psi \mid \mathtt{del} \mid \mathtt{finish} \\
\mathsf{Dir} & \delta & ::= & \mathtt{child}\ n \mid \mathtt{parent} \\
\mathsf{Shape} & \psi & ::= & \mathtt{arrow} \mid \mathtt{num} \\
 & & \mid & \mathtt{asc} \mid \mathtt{var}\ x \mid \mathtt{lam}\ x \mid \mathtt{ap} \mid \mathtt{lit}\ n \mid \mathtt{plus}
\end{array}$$

Contribution 2: A typed edit action semantics

$$
\begin{array}{llll}
\textsf{ZTyp} & \hat{\tau} & ::= & \triangleright\dot{\tau}\triangleleft \mid (\hat{\tau} \to \dot{\tau}) \mid (\dot{\tau} \to \hat{\tau}) \\
\textsf{ZExp} & \hat{e} & ::= & \triangleright\dot{e}\triangleleft \mid (\lambda x.\hat{e}) \mid \hat{e}(\dot{e}) \mid \dot{e}(\hat{e}) \mid (\hat{e} + \dot{e}) \mid (\dot{e} + \hat{e}) \\
 & & \mid & \hat{e} : \dot{\tau} \mid \dot{e} : \hat{\tau} \mid (\!|\hat{e}|\!)
\end{array}
$$

$$
\begin{array}{llll}
\textsf{Action} & \alpha & ::= & \texttt{move}\ \delta \mid \texttt{construct}\ \psi \mid \texttt{del} \mid \texttt{finish} \\
\textsf{Dir} & \delta & ::= & \texttt{child}\ n \mid \texttt{parent} \\
\textsf{Shape} & \psi & ::= & \texttt{arrow} \mid \texttt{num} \\
 & & \mid & \texttt{asc} \mid \texttt{var}\ x \mid \texttt{lam}\ x \mid \texttt{ap} \mid \texttt{lit}\ n \mid \texttt{plus}
\end{array}
$$

$$\boxed{\dot{\Gamma} \vdash \hat{e} \Rightarrow \dot{\tau} \xrightarrow{\alpha} \hat{e}' \Rightarrow \dot{\tau}'}$$

$$\boxed{\dot{\Gamma} \vdash \hat{e} \xrightarrow{\alpha} \hat{e}' \Leftarrow \dot{\tau}}$$

Contribution 2: A typed edit action semantics

$$\begin{array}{lrcl} \text{ZTyp} & \hat{\tau} & ::= & \triangleright\dot{\tau}\triangleleft \mid (\hat{\tau} \rightarrow \dot{\tau}) \mid (\dot{\tau} \rightarrow \hat{\tau}) \\ \text{ZExp} & \hat{e} & ::= & \triangleright\dot{e}\triangleleft \mid (\lambda x.\hat{e}) \mid \hat{e}(\dot{e}) \mid \dot{e}(\hat{e}) \mid (\hat{e} + \dot{e}) \mid (\dot{e} + \hat{e}) \\ & & \mid & \hat{e} : \dot{\tau} \mid \dot{e} : \hat{\tau} \mid (\!|\hat{e}|\!) \end{array}$$

$$\begin{array}{lrcl} \text{Action} & \alpha & ::= & \mathtt{move}\ \delta \mid \mathtt{construct}\ \psi \mid \mathtt{del} \mid \mathtt{finish} \\ \text{Dir} & \delta & ::= & \mathtt{child}\ n \mid \mathtt{parent} \\ \text{Shape} & \psi & ::= & \mathtt{arrow} \mid \mathtt{num} \\ & & \mid & \mathtt{asc} \mid \mathtt{var}\ x \mid \mathtt{lam}\ x \mid \mathtt{ap} \mid \mathtt{lit}\ n \mid \mathtt{plus} \end{array}$$

$$\boxed{\dot{\Gamma} \vdash \hat{e} \Rightarrow \dot{\tau} \xrightarrow{\alpha} \hat{e}' \Rightarrow \dot{\tau}'}$$

$$\boxed{\dot{\Gamma} \vdash \hat{e} \xrightarrow{\alpha} \hat{e}' \Leftarrow \dot{\tau}}$$

$$\frac{}{\dot{\Gamma} \vdash \triangleright(\!|\,|\!)\triangleleft \Rightarrow (\!|\,|\!) \xrightarrow{\mathtt{construct}\ \mathtt{lit}\ n} \triangleright\underline{n}\triangleleft \Rightarrow \mathtt{num}}$$

Contribution 2: A typed edit action semantics

$$
\begin{array}{lll}
\text{ZTyp } \hat{\tau} & ::= & \triangleright\dot{\tau}\triangleleft \mid (\hat{\tau} \to \dot{\tau}) \mid (\dot{\tau} \to \hat{\tau}) \\
\text{ZExp } \hat{e} & ::= & \triangleright\dot{e}\triangleleft \mid (\lambda x.\hat{e}) \mid \hat{e}(\dot{e}) \mid \dot{e}(\hat{e}) \mid (\hat{e} + \dot{e}) \mid (\dot{e} + \hat{e}) \\
 & \mid & \hat{e} : \dot{\tau} \mid \dot{e} : \hat{\tau} \mid (\!|\hat{e}|\!)
\end{array}
$$

$$
\begin{array}{llll}
\text{Action} & \alpha & ::= & \mathtt{move}\ \delta \mid \mathtt{construct}\ \psi \mid \mathtt{del} \mid \mathtt{finish} \\
\text{Dir} & \delta & ::= & \mathtt{child}\ n \mid \mathtt{parent} \\
\text{Shape} & \psi & ::= & \mathtt{arrow} \mid \mathtt{num} \\
 & & \mid & \mathtt{asc} \mid \mathtt{var}\ x \mid \mathtt{lam}\ x \mid \mathtt{ap} \mid \mathtt{lit}\ n \mid \mathtt{plus}
\end{array}
$$

$$\boxed{\dot{\Gamma} \vdash \hat{e} \Rightarrow \dot{\tau} \xrightarrow{\alpha} \hat{e}' \Rightarrow \dot{\tau}'}$$

$$\boxed{\dot{\Gamma} \vdash \hat{e} \xrightarrow{\alpha} \hat{e}' \Leftarrow \dot{\tau}}$$

$$\frac{}{\dot{\Gamma} \vdash \triangleright(\!|\,|\!)\triangleleft \Rightarrow (\!|\,|\!) \xrightarrow{\mathtt{construct\ lit}\ n} \triangleright\underline{n}\triangleleft \Rightarrow \mathtt{num}}$$

$$\frac{\dot{\Gamma} \vdash \hat{e} \xrightarrow{\alpha} \hat{e}' \Leftarrow \mathtt{num}}{\dot{\Gamma} \vdash (\hat{e} + \dot{e}) \Rightarrow \mathtt{num} \xrightarrow{\alpha} (\hat{e}' + \dot{e}) \Rightarrow \mathtt{num}}$$

Contribution 2: A typed edit action semantics

$$
\begin{array}{llll}
\text{ZTyp} & \hat{\tau} & ::= & \triangleright\dot{\tau}\triangleleft \mid (\hat{\tau} \to \dot{\tau}) \mid (\dot{\tau} \to \hat{\tau}) \\
\text{ZExp} & \hat{e} & ::= & \triangleright\dot{e}\triangleleft \mid (\lambda x.\hat{e}) \mid \hat{e}(\dot{e}) \mid \dot{e}(\hat{e}) \mid (\hat{e} + \dot{e}) \mid (\dot{e} + \hat{e}) \\
& & \mid & \hat{e} : \dot{\tau} \mid \dot{e} : \hat{\tau} \mid (\!|\hat{e}|\!)
\end{array}
$$

$$
\begin{array}{llll}
\text{Action} & \alpha & ::= & \texttt{move}\ \delta \mid \texttt{construct}\ \psi \mid \texttt{del} \mid \texttt{finish} \\
\text{Dir} & \delta & ::= & \texttt{child}\ n \mid \texttt{parent} \\
\text{Shape} & \psi & ::= & \texttt{arrow} \mid \texttt{num} \\
& & \mid & \texttt{asc} \mid \texttt{var}\ x \mid \texttt{lam}\ x \mid \texttt{ap} \mid \texttt{lit}\ n \mid \texttt{plus}
\end{array}
$$

$$\boxed{\dot{\Gamma} \vdash \hat{e} \Rightarrow \dot{\tau} \xrightarrow{\alpha} \hat{e}' \Rightarrow \dot{\tau}'}$$

$$\frac{}{\dot{\Gamma} \vdash \triangleright(\!|\,|\!)\triangleleft \Rightarrow (\!|\,|\!) \xrightarrow{\texttt{construct lit}\ n} \triangleright\underline{n}\triangleleft \Rightarrow \texttt{num}}$$

$$\frac{\dot{\Gamma} \vdash \hat{e} \xrightarrow{\alpha} \hat{e}' \Leftarrow \texttt{num}}{\dot{\Gamma} \vdash (\hat{e} + \dot{e}) \Rightarrow \texttt{num} \xrightarrow{\alpha} (\hat{e}' + \dot{e}) \Rightarrow \texttt{num}}$$

$$\boxed{\dot{\Gamma} \vdash \hat{e} \xrightarrow{\alpha} \hat{e}' \Leftarrow \dot{\tau}}$$

$$\frac{\dot{\Gamma} \vdash \hat{e}^{\diamond} \Rightarrow \dot{\tau}' \qquad \dot{\Gamma} \vdash \hat{e} \Rightarrow \dot{\tau}' \xrightarrow{\alpha} \hat{e}' \Rightarrow \dot{\tau}'' \qquad \dot{\tau} \sim \dot{\tau}''}{\dot{\Gamma} \vdash \hat{e} \xrightarrow{\alpha} \hat{e}' \Leftarrow \dot{\tau}}$$

Contribution 2: A typed edit action semantics

$$\begin{array}{llcl} \text{ZTyp} & \hat{\tau} & ::= & \triangleright\dot{\tau}\triangleleft \mid (\hat{\tau} \to \dot{\tau}) \mid (\dot{\tau} \to \hat{\tau}) \\ \text{ZExp} & \hat{e} & ::= & \triangleright\dot{e}\triangleleft \mid (\lambda x.\hat{e}) \mid \hat{e}(\dot{e}) \mid \dot{e}(\hat{e}) \mid (\hat{e} + \dot{e}) \mid (\dot{e} + \hat{e}) \\ & & \mid & \hat{e} : \dot{\tau} \mid \dot{e} : \hat{\tau} \mid (\!|\hat{e}|\!) \end{array}$$

$$\begin{array}{llcl} \text{Action} & \alpha & ::= & \texttt{move}\ \delta \mid \texttt{construct}\ \psi \mid \texttt{del} \mid \texttt{finish} \\ \text{Dir} & \delta & ::= & \texttt{child}\ n \mid \texttt{parent} \\ \text{Shape} & \psi & ::= & \texttt{arrow} \mid \texttt{num} \\ & & \mid & \texttt{asc} \mid \texttt{var}\ x \mid \texttt{lam}\ x \mid \texttt{ap} \mid \texttt{lit}\ n \mid \texttt{plus} \end{array}$$

$$\boxed{\dot{\Gamma} \vdash \hat{e} \Rightarrow \dot{\tau} \xrightarrow{\alpha} \hat{e}' \Rightarrow \dot{\tau}'}$$

$$\frac{}{\dot{\Gamma} \vdash \triangleright(\!|\,|\!)\triangleleft \Rightarrow (\!|\,|\!) \xrightarrow{\texttt{construct lit}\ n} \triangleright\underline{n}\triangleleft \Rightarrow \texttt{num}}$$

$$\frac{\dot{\Gamma} \vdash \hat{e} \xrightarrow{\alpha} \hat{e}' \Leftarrow \texttt{num}}{\dot{\Gamma} \vdash (\hat{e} + \dot{e}) \Rightarrow \texttt{num} \xrightarrow{\alpha} (\hat{e}' + \dot{e}) \Rightarrow \texttt{num}}$$

$$\boxed{\dot{\Gamma} \vdash \hat{e} \xrightarrow{\alpha} \hat{e}' \Leftarrow \dot{\tau}}$$

$$\frac{\dot{\Gamma} \vdash \hat{e}^{\diamond} \Rightarrow \dot{\tau}' \qquad \dot{\Gamma} \vdash \hat{e} \Rightarrow \dot{\tau}' \xrightarrow{\alpha} \hat{e}' \Rightarrow \dot{\tau}'' \qquad \dot{\tau} \sim \dot{\tau}''}{\dot{\Gamma} \vdash \hat{e} \xrightarrow{\alpha} \hat{e}' \Leftarrow \dot{\tau}}$$

$$\frac{\dot{\tau} \nsim \texttt{num}}{\dot{\Gamma} \vdash \triangleright(\!|\,|\!)\triangleleft \xrightarrow{\texttt{construct lit}\ n} (\!|\triangleright\underline{n}\triangleleft|\!) \Leftarrow \dot{\tau}}$$

Metatheorem: **Sensibility**

Every edit action leaves the edit state well-typed.

Theorem 1 (Action Sensibility).

1. *If* $\dot{\Gamma} \vdash \hat{e}^{\diamond} \Rightarrow \dot{\tau}$ *and* $\dot{\Gamma} \vdash \hat{e} \Rightarrow \dot{\tau} \xrightarrow{\alpha} \hat{e}' \Rightarrow \dot{\tau}'$ *then* $\dot{\Gamma} \vdash \hat{e}'^{\diamond} \Rightarrow \dot{\tau}'$.
2. *If* $\dot{\Gamma} \vdash \hat{e}^{\diamond} \Leftarrow \dot{\tau}$ *and* $\dot{\Gamma} \vdash \hat{e} \xrightarrow{\alpha} \hat{e}' \Leftarrow \dot{\tau}$ *then* $\dot{\Gamma} \vdash \hat{e}'^{\diamond} \Leftarrow \dot{\tau}$.

Metatheorem: **Reachability**

The cursor can reach any position in the program.

Theorem 3 (Reachability).

1. *If* $\hat{\tau}^\diamond = \hat{\tau}'^\diamond$ *then there exists some* $\bar{\alpha}$ *such that* $\bar{\alpha}$ movements *and* $\hat{\tau} \xrightarrow{\bar{\alpha}}{}^* \hat{\tau}'$.
2. *If* $\dot{\Gamma} \vdash \hat{e}^\diamond \Rightarrow \dot{\tau}$ *and* $\hat{e}^\diamond = \hat{e}'^\diamond$ *then there exists some* $\bar{\alpha}$ *such that* $\bar{\alpha}$ movements *and* $\dot{\Gamma} \vdash \hat{e} \Rightarrow \dot{\tau} \xrightarrow{\bar{\alpha}}{}^* \hat{e}' \Rightarrow \dot{\tau}$.
3. *If* $\dot{\Gamma} \vdash \hat{e}^\diamond \Leftarrow \dot{\tau}$ *and* $\hat{e}^\diamond = \hat{e}'^\diamond$ *then there exists some* $\bar{\alpha}$ *such that* $\bar{\alpha}$ movements *and* $\dot{\Gamma} \vdash \hat{e} \xrightarrow{\bar{\alpha}}{}^* \hat{e}' \Leftarrow \dot{\tau}$.

Metatheorem: **Constructability**

Any well-typed expression can be constructed using edit actions.

Theorem 6 (Constructability)**.**

1. *For every* $\dot{\tau}$ *there exists* $\bar{\alpha}$ *such that* $\triangleright(\!|\,|\!)\triangleleft \xrightarrow{\bar{\alpha}}{}^{*} \triangleright\dot{\tau}\triangleleft$.
2. *If* $\dot{\Gamma} \vdash \dot{e} \Rightarrow \dot{\tau}$ *then there exists* $\bar{\alpha}$ *such that:*

$$\dot{\Gamma} \vdash \triangleright(\!|\,|\!)\triangleleft \Rightarrow (\!|\,|\!) \xrightarrow{\bar{\alpha}}{}^{*} \triangleright\dot{e}\triangleleft \Rightarrow \dot{\tau}$$

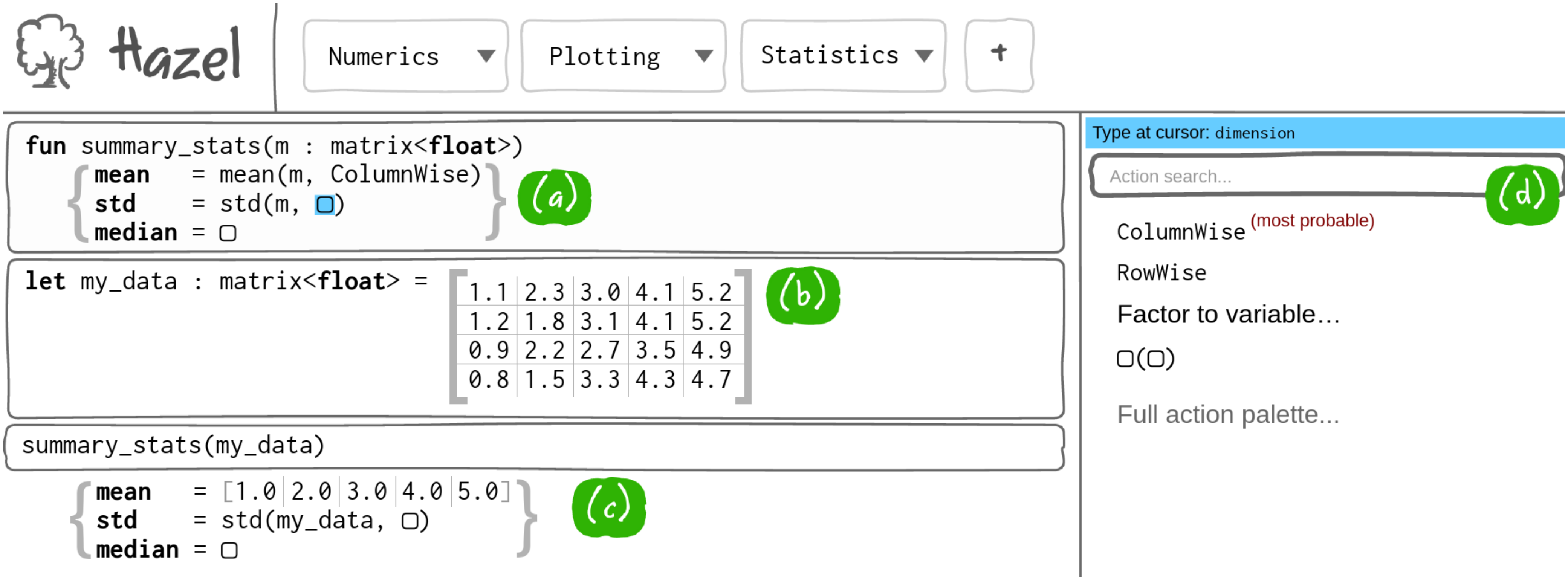
3. *If* $\dot{\Gamma} \vdash \dot{e} \Leftarrow \dot{\tau}$ *then there exists* $\bar{\alpha}$ *such that:*

$$\dot{\Gamma} \vdash \triangleright(\!|\,|\!)\triangleleft \xrightarrow{\bar{\alpha}}{}^{*} \triangleright\dot{e}\triangleleft \Leftarrow \dot{\tau}$$

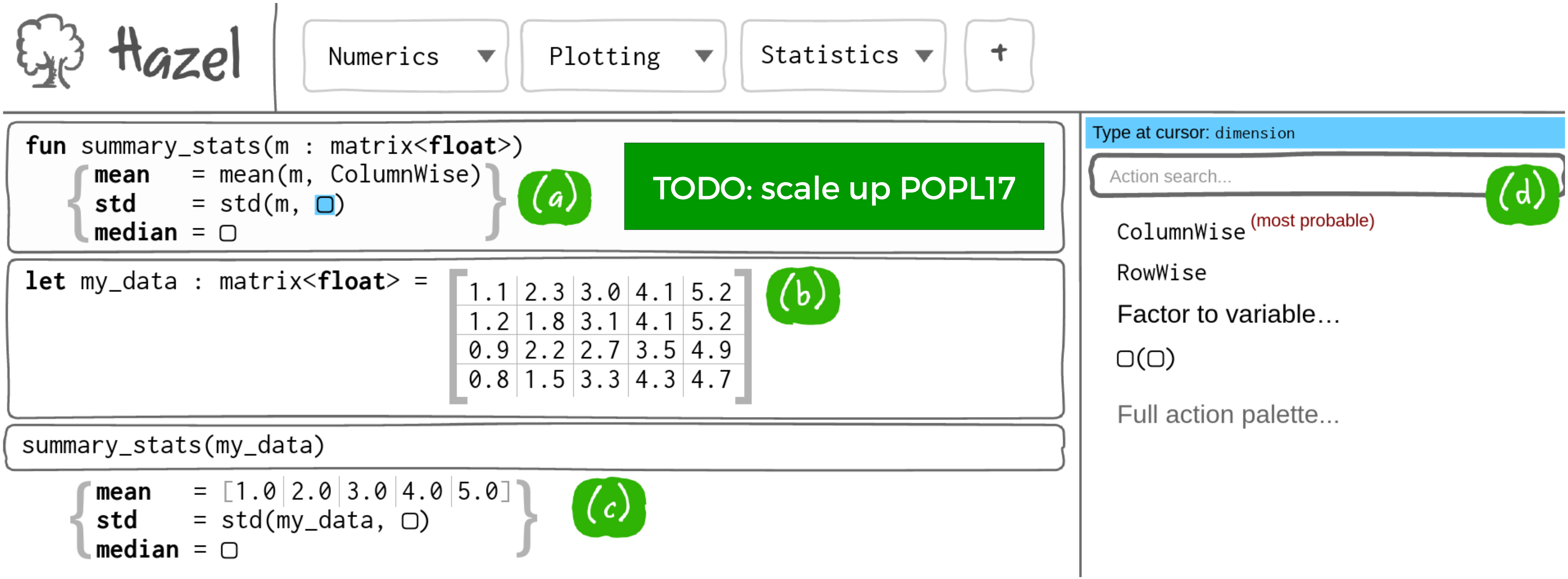
Summary: **Hazelnut**

- A **static semantics** for terms with holes and type inconsistencies.
- An **typed action semantics** that maintains sensibility invariant.
  - **HZ**: A reference implementation written in OCaml React + js_of_ocaml.
- A **rich metatheory** that establishes the correctness of Hazelnut.
  - Mechanized using the **Agda** proof assistant.
  - Guides the definition of an extension (sum types – see paper!)

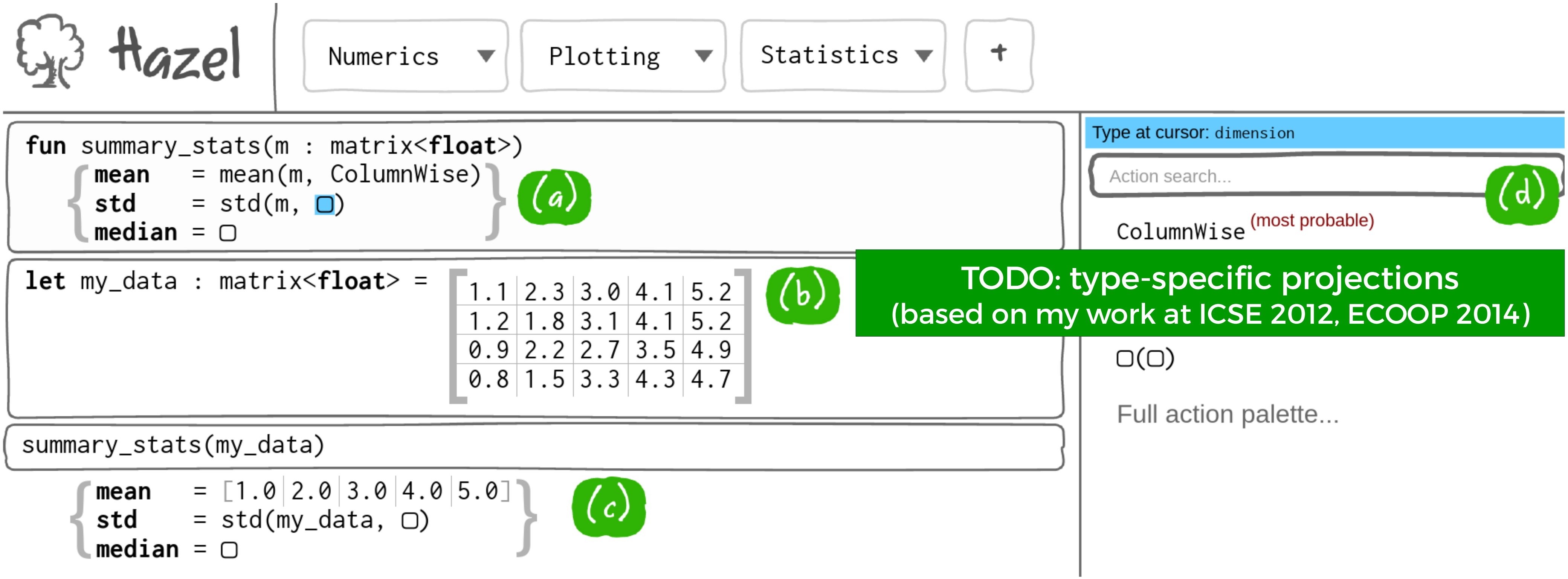
From Hazelnut to Hazel



From Hazelnut to Hazel

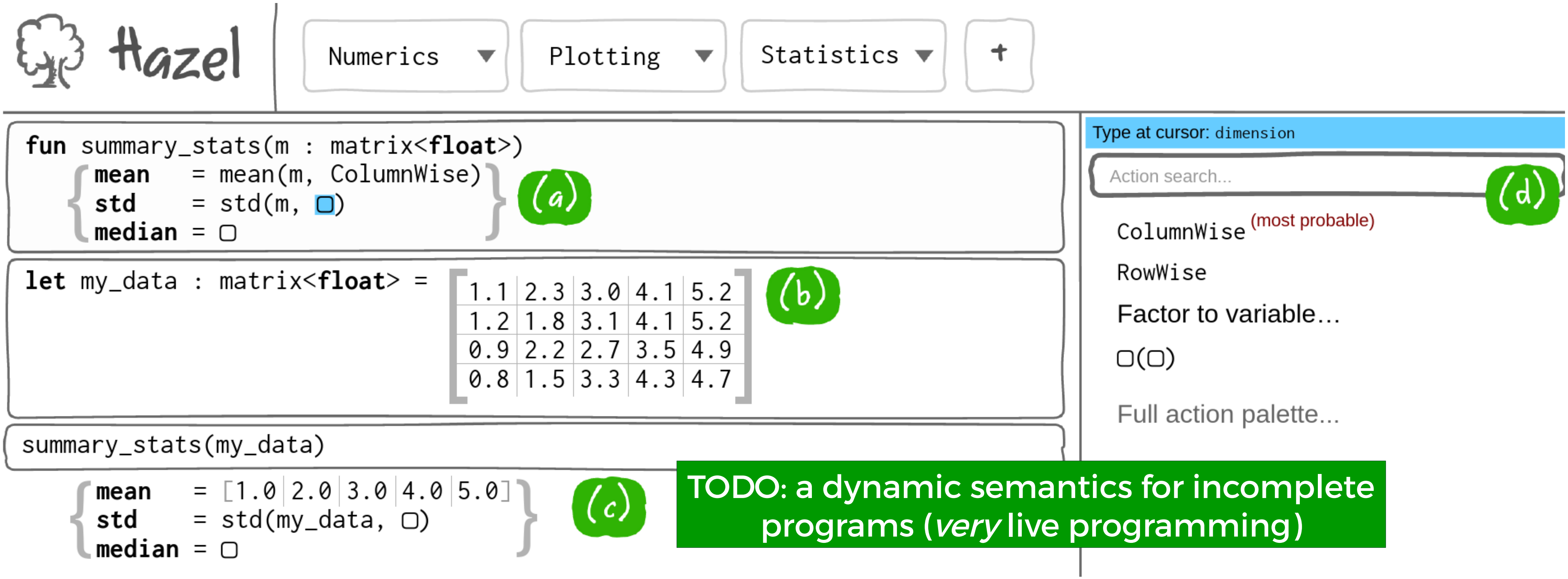


From Hazelnut to Hazel



TODO: type-specific projections
(based on my work at ICSE 2012, ECOOP 2014)

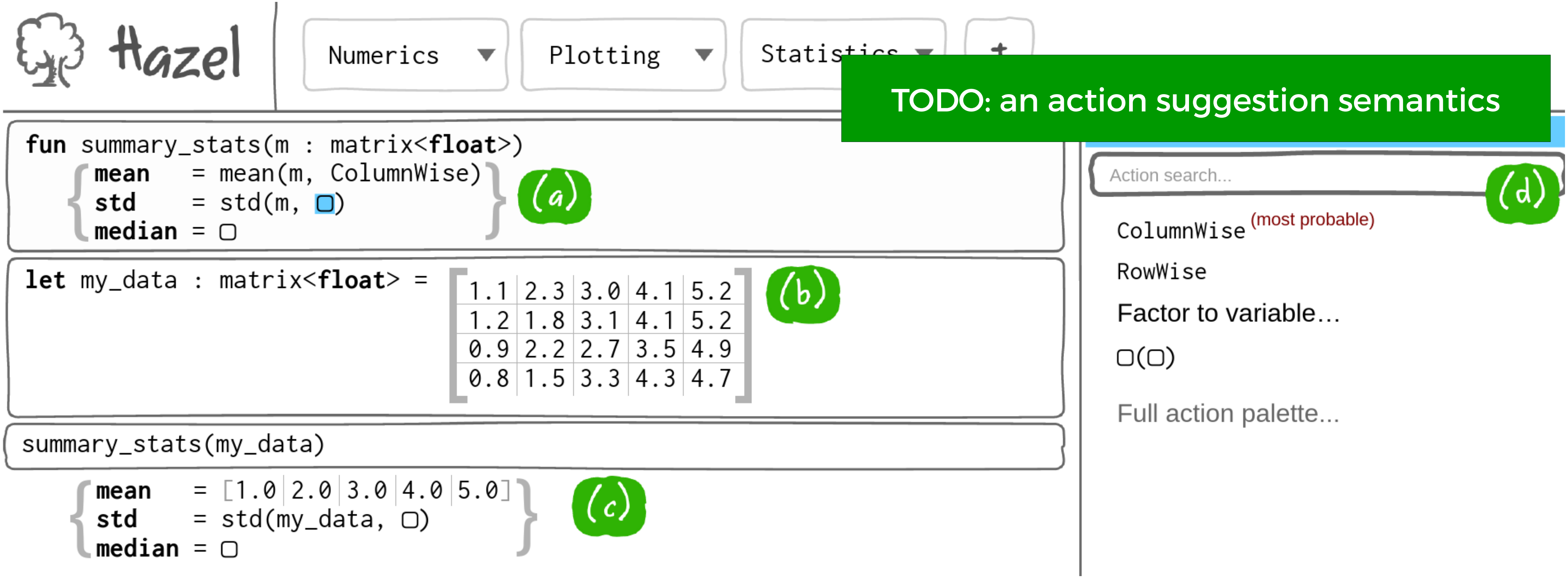
From Hazelnut to Hazel



TODO: a dynamic semantics for incomplete programs (*very* live programming)

From Hazelnut to **Hazel**

TODO: an action suggestion semantics



From Hazelnut to **Hazel**

Hazel

Numerics ▼ | Plotting ▼ | Statistics ▼ | +

```
fun summary_stats(m : matrix<float>)
  { mean   = mean(m, ColumnWise)
    std    = std(m, ▢)
    median = ▢ }
```

(a)

```
let my_data : matrix<float> =
```

1.1	2.3	3.0	4.1	5.2
1.2	1.8	3.1	4.1	5.2
0.9	2.2	2.7	3.5	4.9
0.8	1.5	3.3	4.3	4.7

```
summary_stats(my_data)
```

```
{ mean   = [1.0 2.0 3.0 4.0 5.0]
  std    = std(my_data, ▢)
  median = ▢ }
```

(c)

Type at cursor: `dimension`

Action search...

(d)

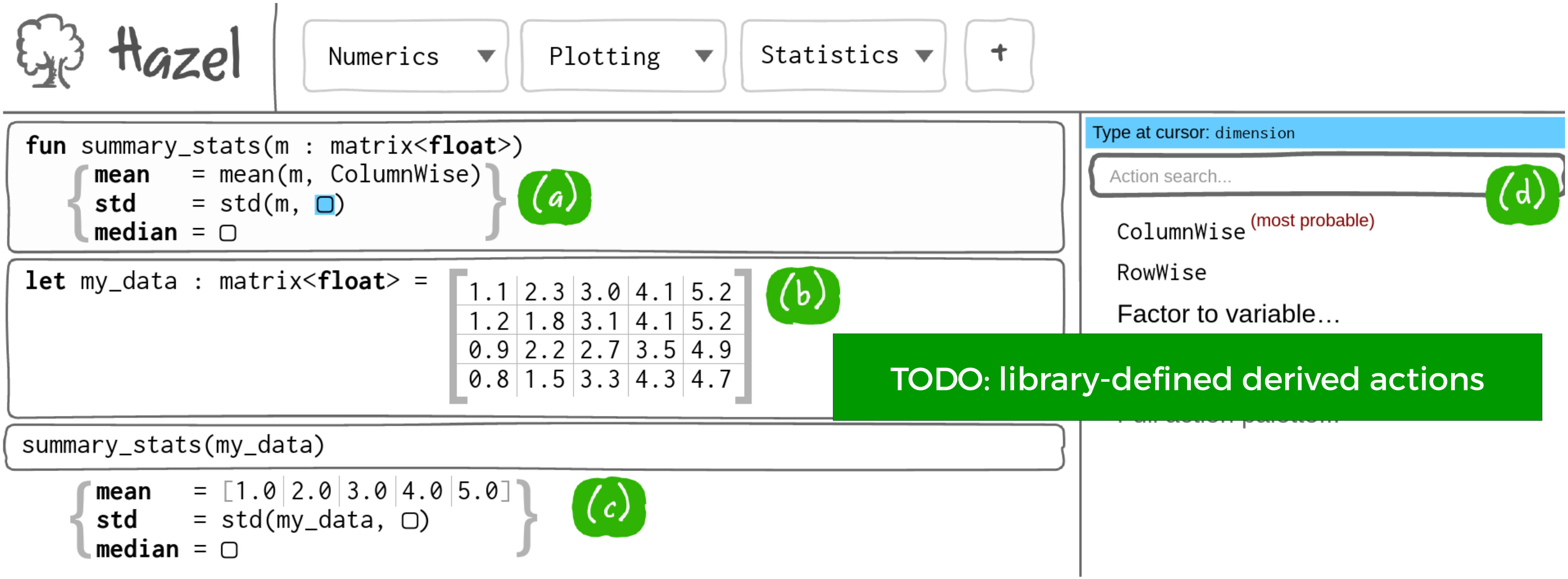
`ColumnWise` (most probable)

TODO: a statistical model of edit actions

▢(▢)

Full action palette...

From Hazelnut to Hazel



TODO: library-defined derived actions

From Hazelnut to **Hazel**

Hazel

Numerics ▼ | Plotting ▼ | Statistics ▼ | +

```
fun summary_stats(m : matrix<float>)
  { mean   = mean(m, ColumnWise)
    std    = std(m, ▢)
    median = ▢ }
```

(a)

```
let my_data : matrix<float> =
```

1.1	2.3	3.0	4.1	5.2
1.2	1.8	3.1	4.1	5.2
0.9	2.2	2.7	3.5	4.9
0.8	1.5	3.3	4.3	4.7

(b)

```
summary_stats(my_data)
```

```
{ mean   = [1.0 2.0 3.0 4.0 5.0]
  std    = std(my_data, ▢)
  median = ▢ }
```

(c)

Type at cursor: `dimension`

Action search...

(d)

ColumnWise (most probable)

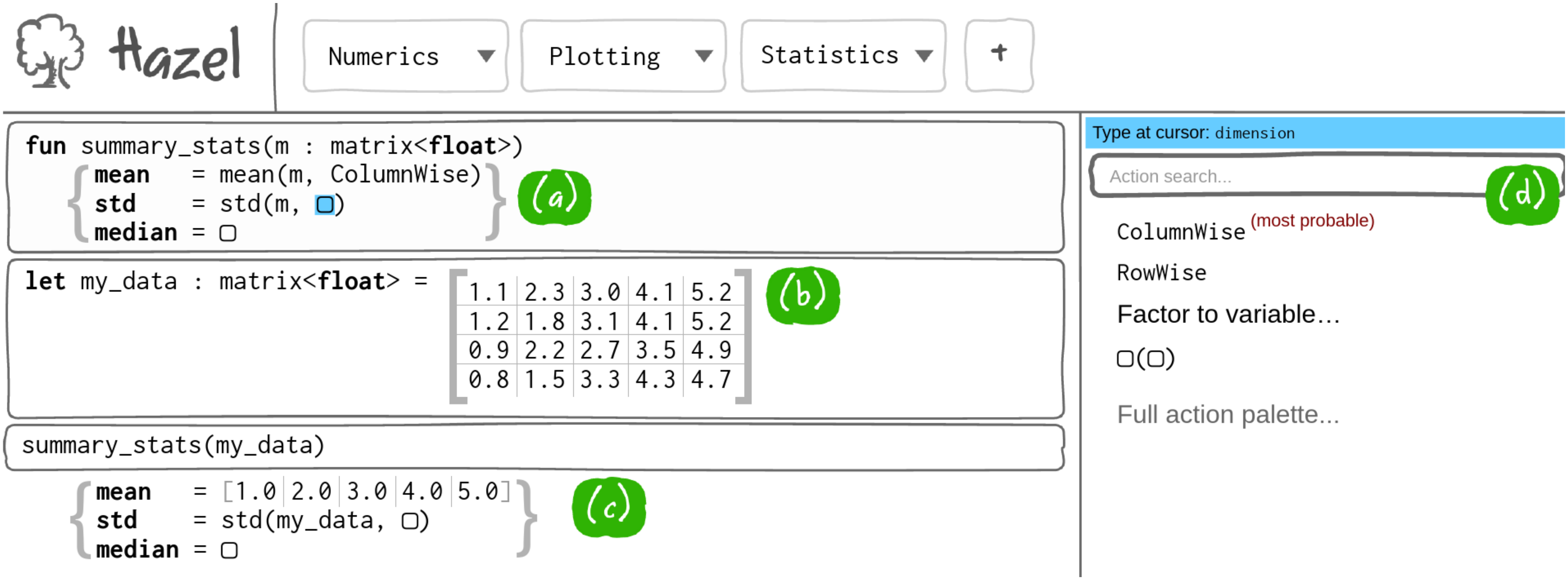
RowWise

Factor to variable…

▢(▢)

Full action palette...

From Hazelnut to **Hazel**



See http://www.hazelgrove.org/.

Thanks!