Elaborating dependent (co)pattern matching

Jesper Cockx

Gothenburg University – Chalmers

31 January 2018

Jesper.

Jesper. You are talking about pattern matching . . .

Jesper. You are talking about pattern matching again???

Jesper. You are talking about pattern matching again???

Previously

From a case tree
to eliminators

Jesper. You are talking about pattern matching again???

Previously

From a case tree to eliminators

Now

From clauses to a case tree

Motivation

Dependent pattern matching is a big part of Agda, so important to get it right.

- Preserve meaning of clauses
- Be flexible in placement of dot patterns
- Combine LHS and coverage checking?

Stepping stone for an Agda Core?

Disclaimer

This is work in progress, both the theory and the implementation.

If you have any suggestions I'd love to hear them!

Checking definitions by (co)pattern matching

Agda core v0.1

From clauses to a case tree

Example: maximum

$$
\begin{array}{llll}
\texttt{max} : \mathbb{N} \to \mathbb{N} \to \mathbb{N} & & & \\
\texttt{max}\ \texttt{zero} & n & = & n \\
\texttt{max}\ m & \texttt{zero} & = & m \\
\texttt{max}\ (\texttt{suc}\ m) & \texttt{suc}\ n & = & \texttt{suc}\ (\texttt{max}\ m\ n)
\end{array}
$$

$$\vdash \mathtt{max} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$$

$$\begin{array}{lll}
\mathtt{zero} & j & \mapsto j \\
i & \mathtt{zero} & \mapsto i \\
(\mathtt{suc}\ k) & \mathtt{suc}\ l & \mapsto \mathtt{suc}\ (\mathtt{max}\ k\ l)
\end{array}$$

$$(m : \mathbb{N}) \vdash \mathtt{max}\ m : \mathbb{N} \to \mathbb{N}$$

$$\begin{array}{lll}
[\mathtt{zero} = m] & j & \mapsto j \\
[i = m] & \mathtt{zero} & \mapsto i \\
[\mathtt{suc}\ k = m] & \mathtt{suc}\ l & \mapsto \mathtt{suc}\ (\mathtt{max}\ k\ l)
\end{array}$$

$$\vdash \mathtt{max}\ \mathtt{zero} : \mathbb{N} \to \mathbb{N}$$

$$\begin{array}{lll} [\mathtt{zero} = \mathtt{zero}] & j & \mapsto j \\ [i = \mathtt{zero}] & \mathtt{zero} & \mapsto i \\ [\mathtt{suc}\ k = \mathtt{zero}] & \mathtt{suc}\ l & \mapsto \mathtt{suc}\ (\mathtt{max}\ k\ l) \end{array}$$

$$(p : \mathbb{N}) \vdash \mathtt{max}\ (\mathtt{suc}\ p) : \mathbb{N} \to \mathbb{N}$$

$$\begin{array}{lll} [\mathtt{zero} = \mathtt{suc}\ p] & j & \mapsto j \\ [i = \mathtt{suc}\ p] & \mathtt{zero} & \mapsto i \\ [\mathtt{suc}\ k = \mathtt{suc}\ p] & \mathtt{suc}\ l & \mapsto \mathtt{suc}\ (\mathtt{max}\ k\ l) \end{array}$$

$$\vdash \mathtt{max}\ \mathtt{zero} : \mathbb{N} \to \mathbb{N}$$

$$\begin{array}{rcl} & j & \mapsto j \\ [i = \mathtt{zero}] & \mathtt{zero} & \mapsto i \end{array}$$

$$(p : \mathbb{N}) \vdash \mathtt{max}\ (\mathtt{suc}\ p) : \mathbb{N} \to \mathbb{N}$$

$$\begin{array}{rcl} [i = \mathtt{suc}\ p] & \mathtt{zero} & \mapsto i \\ [k = p] & \mathtt{suc}\ l & \mapsto \mathtt{suc}\ (\mathtt{max}\ k\ l) \end{array}$$

$$(n : \mathbb{N}) \vdash \mathtt{max}\ \mathtt{zero}\ n : \mathbb{N}$$

$$\begin{array}{ll} [j = n] & \mapsto j \\ [i = \mathtt{zero}, \mathtt{zero} = n] & \mapsto i \end{array}$$

$$(p : \mathbb{N})(n : \mathbb{N}) \vdash \mathtt{max}\ (\mathtt{suc}\ p)\ n : \mathbb{N}$$

$$\begin{array}{ll} [i = \mathtt{suc}\ p, \mathtt{zero} = n] & \mapsto i \\ [k = p, \mathtt{suc}\ l = n] & \mapsto \mathtt{suc}\ (\mathtt{max}\ k\ l) \end{array}$$

$$(n : \mathbb{N}) \vdash \mathtt{max}\ \mathtt{zero}\ n \mapsto n : \mathbb{N}$$

$$(p : \mathbb{N})(n : \mathbb{N}) \vdash \mathtt{max}\ (\mathtt{suc}\ p)\ n : \mathbb{N}$$

$$\begin{array}{ll} [i = \mathtt{suc}\ p, \mathtt{zero} = n] & \mapsto i \\ [k = p, \mathtt{suc}\ l = n] & \mapsto \mathtt{suc}\ (\mathtt{max}\ k\ l) \end{array}$$

$$(n : \mathbb{N}) \vdash \mathtt{max}\ \mathtt{zero}\ n \mapsto n : \mathbb{N}$$

$$(p : \mathbb{N}) \vdash \mathtt{max}\ (\mathtt{suc}\ p)\ \mathtt{zero} : \mathbb{N}$$

$$[i = \mathtt{suc}\ p] \mapsto i$$

$$(p : \mathbb{N})(q : \mathbb{N}) \vdash \mathtt{max}\ (\mathtt{suc}\ p)\ (\mathtt{suc}\ q) : \mathbb{N}$$

$$[k = p, l = q] \mapsto \mathtt{suc}\ (\mathtt{max}\ k\ l)$$

$$(n : \mathbb{N}) \vdash \mathtt{max}\ \mathtt{zero}\ n \mapsto n : \mathbb{N}$$

$$(p : \mathbb{N}) \vdash \mathtt{max}\ (\mathtt{suc}\ p)\ \mathtt{zero} \mapsto \mathtt{suc}\ p : \mathbb{N}$$

$$(p : \mathbb{N})(q : \mathbb{N}) \vdash \mathtt{max}\ (\mathtt{suc}\ p)\ (\mathtt{suc}\ q) : \mathbb{N}$$

$$[k = p, l = q] \mapsto \mathtt{suc}\ (\mathtt{max}\ k\ l)$$

$$(n : \mathbb{N}) \vdash \mathtt{max}\ \mathtt{zero}\ n \mapsto n : \mathbb{N}$$

$$(p : \mathbb{N}) \vdash \mathtt{max}\ (\mathtt{suc}\ p)\ \mathtt{zero} \mapsto \mathtt{suc}\ p : \mathbb{N}$$

$$(p : \mathbb{N})(q : \mathbb{N}) \vdash \mathtt{max}\ (\mathtt{suc}\ p)\ (\mathtt{suc}\ q)$$
$$\mapsto \mathtt{suc}\ (\mathtt{max}\ p\ q) : \mathbb{N}$$

Example with copatterns

```
record UpFrom (n : ℕ) : Type where
  hd : ℕ
  pf : n < hd
  tl : UpFrom hd

from : (n : ℕ) → UpFrom n
from n .hd = suc n
from n .pf = <-suc n
from n .tl = from (suc n)
```

$$\vdash \texttt{from} : (n : \mathbb{N}) \to \texttt{UpFrom}\ n$$

$$n\ .\text{hd} \mapsto \texttt{suc}\ n$$

$$n\ .\text{pf} \mapsto \texttt{<-suc}\ n$$

$$n\ .\text{tl} \mapsto \texttt{from}\ (\texttt{suc}\ n)$$

$$(n : \mathbb{N}) \vdash \texttt{from}\ n : \texttt{UpFrom}\ n$$

$$[n = n]\ \texttt{.hd} \mapsto \texttt{suc}\ n$$
$$[n = n]\ \texttt{.pf} \mapsto <\texttt{-suc}\ n$$
$$[n = n]\ \texttt{.tl} \mapsto \texttt{from}\ (\texttt{suc}\ n)$$

$$(n : \mathbb{N}) \vdash \texttt{from}\ n\ .\texttt{hd} : \mathbb{N}$$

$$[n = n] \mapsto \texttt{suc}\ n$$

$$(n : \mathbb{N}) \vdash \texttt{from}\ n\ .\texttt{pf} : n < (\texttt{from}\ n\ .\texttt{hd})$$

$$[n = n] \mapsto <\texttt{-suc}\ n$$

$$(n : \mathbb{N}) \vdash \texttt{from}\ n\ .\texttt{tl} : \texttt{UpFrom}\ (\texttt{from}\ n\ .\texttt{hd})$$

$$[n = n] \mapsto \texttt{from}\ (\texttt{suc}\ n)$$

$$(n : \mathbb{N}) \vdash \texttt{from}\ n\ .\texttt{hd} \mapsto \texttt{suc}\ n : \mathbb{N}$$

$$(n : \mathbb{N}) \vdash \texttt{from}\ n\ .\texttt{pf} : n < \texttt{suc}\ n$$

$$[n = n] \mapsto <\texttt{-suc}\ n$$

$$(n : \mathbb{N}) \vdash \texttt{from}\ n\ .\texttt{tl} : \texttt{UpFrom}\,(\texttt{suc}\ n)$$

$$[n = n] \mapsto \texttt{from}\,(\texttt{suc}\ n)$$

$$(n : \mathbb{N}) \vdash \texttt{from}\ n\ \texttt{.hd} \mapsto \texttt{suc}\ n : \mathbb{N}$$

$$(n : \mathbb{N}) \vdash \texttt{from}\ n\ \texttt{.pf}$$
$$\mapsto {<}\texttt{-suc}\ n : n < \texttt{suc}\ n$$

$$(n : \mathbb{N}) \vdash \texttt{from}\ n\ \texttt{.tl}$$
$$\mapsto \texttt{from}\ (\texttt{suc}\ n) : \texttt{UpFrom}\ (\texttt{suc}\ n)$$

Example based on #2896

```
data D : ℕ → Type where
  c : (n : ℕ) → D n

foo : (m : ℕ) → D (suc m) → ℕ
foo m (c (suc n)) = m + n
```

$$\texttt{data}\ \texttt{D} : \mathbb{N} \to \texttt{Type}\ \texttt{where}$$
$$\quad \texttt{c} : (n : \mathbb{N}) \to \texttt{D}\ n$$

$$\vdash \texttt{foo} : (m : \mathbb{N}) \to \texttt{D}\ (\texttt{suc}\ m) \to \mathbb{N}$$

$$m\ (\texttt{c}\ (\texttt{suc}\ n)) \mapsto m + n$$

$$\texttt{data}\ \texttt{D} : \mathbb{N} \to \texttt{Type}\ \texttt{where}$$
$$\quad \texttt{c} : (n : \mathbb{N}) \to \texttt{D}\ n$$

$$(m : \mathbb{N})(x : \texttt{D}\ (\texttt{suc}\ m)) \vdash \texttt{foo}\ m\ x : \mathbb{N}$$

$$[m = m, \texttt{c}\ (\texttt{suc}\ n) = x] \mapsto m + n$$

$$\mathtt{data}\ \mathtt{D} : \mathbb{N} \to \mathtt{Type}\ \mathtt{where}$$
$$\quad \mathtt{c} : (n : \mathbb{N}) \to \mathtt{D}\ n$$

$$(m : \mathbb{N}) \vdash \mathtt{foo}\ m\ (\mathtt{c}\ (\mathtt{suc}\ m)) : \mathbb{N}$$

$$[m = m, \mathtt{c}\ (\mathtt{suc}\ n) = \mathtt{c}\ (\mathtt{suc}\ m)] \mapsto m + n$$

$$\texttt{data}\ \texttt{D} : \mathbb{N} \to \texttt{Type}\ \texttt{where}$$
$$\quad \texttt{c} : (n : \mathbb{N}) \to \texttt{D}\ n$$

$$(m : \mathbb{N}) \vdash \texttt{foo}\ m\ (\texttt{c}\ (\texttt{suc}\ m)) : \mathbb{N}$$

$$[m = m, n = m] \mapsto m + n$$

$$\texttt{data}\ \texttt{D} : \mathbb{N} \to \texttt{Type}\ \texttt{where}$$
$$\quad \texttt{c} : (n : \mathbb{N}) \to \texttt{D}\ n$$

$$(m : \mathbb{N}) \vdash \texttt{foo}\ m\ (\texttt{c}\ (\texttt{suc}\ m))$$
$$\quad \mapsto m + m : \mathbb{N}$$

General form of a LHS problem

$$
\begin{array}{ll}
\begin{bmatrix} p_{11} = v_{11} : A_{11} \\ \vdots \\ p_{1n_1} = v_{1n_1} : A_{1n_1} \end{bmatrix} & q_{11} \; \ldots \; \mapsto \; rhs_1 \\
\vdots & \\
\begin{bmatrix} p_{m1} = v_{m1} : A_{m1} \\ \vdots \\ p_{mn_m} = v_{mn_m} : A_{mn_m} \end{bmatrix} & q_{m1} \; \ldots \; \mapsto \; rhs_m
\end{array}
$$

Operations on LHS problems

$$\textsc{NextSplit}(\mathit{problem})$$
$$\Rightarrow \textsc{SplitOn}\ x$$
$$\mid\ \ \textsc{SplitOnResult}$$
$$\mid\ \ \textsc{SplittingDone}\ \mathit{rhs}$$

$$\textsc{UpdatePatterns}(\sigma, \mathit{problem})$$
$$\Rightarrow \mathit{problem}'$$

$$\textsc{UpdateTarget}(e, \mathit{problem})$$
$$\Rightarrow \mathit{problem}'$$

Checking definitions by (co)pattern matching

Agda core v0.1

From clauses to a case tree

Introducing: Agda Core v0.1

Included:

- Parametrized datatypes
- Coinductive records
- Equality type
- (Co)pattern matching

Introducing: Agda Core v0.1

Included:

- Parametrized datatypes
- Coinductive records
- Equality type
- (Co)pattern matching

Not included:

- Lambda expressions
- Mutual definitions
- Termination
- Metavariables

Introducing: Agda Core v0.1

Included:

- Parametrized datatypes
- Coinductive records
- Equality type
- (Co)pattern matching

Not included:

- Lambda expressions
- Mutual definitions
- Termination
- Metavariables , . . .

Introducing: Agda Core v0.1

Included:

- Parametrized datatypes
- Coinductive records
- Equality type
- (Co)pattern matching

Not included:

- Lambda expressions
- Mutual definitions
- Termination
- Metavariables , . . . , . . .

Term syntax

$$
\begin{aligned}
A, B, u, v \;&::=\; (x : A) \to B \mid \mathtt{Type}_\ell \\
&\mid\; \mathtt{D}\ \bar{u} \mid \mathtt{R}\ \bar{u} \mid u \equiv_A v \\
&\mid\; x\ \bar{e} \mid \mathtt{c}\ \bar{u} \mid \mathtt{refl} \mid \mathtt{f}\ \bar{e} \\
e \;&::=\; u \mid .\pi \\
\Delta \;&::=\; \epsilon \mid (x : A)\Delta
\end{aligned}
$$

Declarations

$$
\begin{array}{lcl}
\mathit{decl} & ::= & \texttt{data}\ \mathtt{D}\ \Delta : \mathtt{Type}_\ell\ \texttt{where}\ \overline{\mathtt{c}\ \Delta} \\
& | & \texttt{record}\ \mathtt{R}\ \Delta : \mathtt{Type}_\ell\ \texttt{where}\ \overline{\mathtt{f} : A} \\
& | & \texttt{definition}\ \mathtt{f} : A\ \texttt{where}\ \overline{\mathit{cls}} \\
\mathit{cls} & ::= & \bar{q} \mapsto u \mid \bar{q} \mapsto \texttt{impossible}\ u \\
q & ::= & p \mid .\pi \\
p & ::= & x \mid \mathtt{c}\ \bar{p} \mid \lfloor u \rfloor \mid \lfloor \mathtt{c} \rfloor\ \bar{p}
\end{array}
$$

Signatures

$$
\begin{array}{rcl}
\Sigma & ::= & \epsilon \\
& | & \Sigma, \mathtt{data}\ \mathtt{D}\ \Delta : \mathtt{Type}_\ell \\
& | & \Sigma, \mathtt{record}\ \mathtt{R}\ \Delta : \mathtt{Type}_\ell \\
& | & \Sigma, \mathtt{constructor}\ \mathtt{c}\ \Delta_{\mathtt{c}} : \mathtt{D}\ \Delta \\
& | & \Sigma, \mathtt{projection}\ x : \mathtt{R}\ \Delta \vdash\ .\pi : A \\
& | & \Sigma, \mathtt{definition}\ \mathtt{f} : A \\
& | & \Sigma, \mathtt{clause}\ \Delta \vdash\ u \mapsto v : B
\end{array}
$$

Typing judgements

- $\Sigma \vdash \Gamma$
- $\Sigma; \Gamma \vdash u : A$
- $\Sigma; \Gamma \mid u : A \vdash \bar{e} : B$
- $\Sigma; \Gamma \vdash \Delta$
- $\Sigma; \Gamma \vdash \bar{u} : \Delta$
- $\Sigma; \Gamma \vdash u = v : A$
- $\Sigma; \Gamma \mid u : A \vdash \bar{e} = \bar{e}' : B$
- $\Sigma; \Gamma \vdash \bar{u} = \bar{v} : \Delta$

Case trees

$$
\begin{aligned}
Q ::= {} & \mathtt{done}\ u \\
\mid {} & \mathtt{intro}_x\ Q \\
\mid {} & \mathtt{split}_x\ \{\mathtt{c_1}\ \Delta_i \mapsto Q_1; \ldots; \mathtt{c_n}\ \Delta_n \mapsto Q_n\} \\
\mid {} & \mathtt{cosplit}\ \{\pi_{\mathtt{1}} \mapsto Q_1; \ldots; \pi_{\mathtt{n}} \mapsto Q_n\} \\
\mid {} & \mathtt{eqsplit}_e\ \tau\ Q \\
\mid {} & \mathtt{absurdsplit}_e
\end{aligned}
$$

Case tree typing

$$\Gamma \mid \mathtt{f} : A \vdash Q$$

"The case tree Q gives a well-typed implementation of a function $\mathtt{f}$ of type A"

Case tree typing: done

$$\frac{\Gamma \;\vdash\; v : C}{\Gamma \;\mid\; u : C \;\vdash\; \texttt{done}\; v}$$

Case tree typing: done

$$\frac{\Gamma \vdash v : C}{\Gamma \mid u : C \vdash \texttt{done}\ v}$$

Note: also update Σ with $\texttt{clause}\ u \mapsto v$!

Case tree typing: intro_x

$$\frac{\Gamma(x : A) \mid u\,x : B \vdash Q}{\Gamma \mid u : (x : A) \to B \vdash \texttt{intro}_x\ Q}$$

Case tree typing: split_x

$$\frac{\begin{array}{c}(\texttt{constructor}\ \texttt{c}_\texttt{i}\ \Delta_i : \texttt{D}\ \Delta \in \Sigma)_{i=1\ldots n} \\ \Delta'_i = \Delta_i[\Delta \mapsto \bar{v}] \qquad \sigma = [x \mapsto \texttt{c}_\texttt{i}\ \Delta'_i] \\ (\Gamma_1 \Delta'_i \Gamma_2 \sigma \mid u\sigma : C\sigma \vdash Q_i)_{i=1\ldots n}\end{array}}{\begin{array}{l}\Gamma_1 (x : \texttt{D}\ \bar{v}) \Gamma_2 \mid u : C \\ \quad \vdash \texttt{split}_x \{\texttt{c}_1\ \Delta_1 \mapsto Q_1; \ldots; \texttt{c}_\texttt{n}\ \Delta_n \mapsto Q_n\}\end{array}}$$

Case tree typing: cosplit

$$
\frac{(\texttt{projection}\; x : \mathtt{R}\;\Delta \vdash\; .\pi_{\mathtt{i}} : A_i \in \Sigma)_{i=1\ldots n} \qquad (\Gamma \mid u\; .\pi_{\mathtt{i}} : A_i[\Delta \mapsto \bar{v}, x \mapsto u] \vdash\; Q_i)_{i=1\ldots n}}{\Gamma \mid u : \mathtt{R}\;\bar{v} \;\vdash \texttt{cosplit}\,\{\pi_1 \mapsto Q_1; \ldots; \pi_{\mathtt{n}} \mapsto Q_n\}}
$$

Unification

If $\textsc{unify}(\Gamma, A, u, v) \Rightarrow (\Gamma', \sigma, \tau)$, then

- $\Gamma' \vdash \sigma : \Gamma$
- $\Gamma' \vdash u\sigma = v\sigma : A\sigma$
- $\Gamma(e : u \equiv_A v) \vdash \tau : \Gamma'$
- $\Gamma' \vdash \tau; \sigma = [] : \Gamma'$
- If $\Gamma'' \vdash \sigma' : \Gamma$ and $\Gamma'' \vdash u\sigma' = v\sigma' : A\sigma'$, then $\sigma' = \sigma; \tau; (\sigma'; [e \mapsto \texttt{refl}])$

Case tree typing: eqsplit$_e$

$$\frac{\begin{array}{c}\textsc{unify}(\Gamma_1, A, v, w) \Rightarrow (\Gamma_1', \sigma, \tau) \\ \sigma' = \sigma; [e \mapsto \mathtt{refl}] \\ \Gamma_1'\Gamma_2\sigma' \mid u\sigma' : C\sigma' \vdash Q\end{array}}{\Gamma_1(e : v \equiv_a w)\Gamma_2 \mid u : C \vdash \mathtt{eqsplit}_e\ \tau\ Q}$$

Case tree typing: absurdsplit$_e$

$$\frac{\textsc{unify}(\Gamma_1, A, v, w) \Rightarrow \bot}{\Gamma_1(e : v \equiv_a w)\Gamma_2 \mid u : C \vdash \texttt{absurdsplit}_e}$$

Operational semantics

$$\textsc{Eval}\big(Q, \sigma, \bar{e}\big) \Rightarrow v$$

$$\frac{}{\textsc{Eval}\big(\texttt{done}\ v, \sigma, \bar{e}\big) \Rightarrow v\sigma\ \bar{e}}$$

$$\frac{\textsc{Eval}\big(Q, (\sigma; [x \mapsto u]), \bar{e}\big) \Rightarrow v}{\textsc{Eval}\big(\texttt{intro}_x\ Q, \sigma, u\ \bar{e}\big) \Rightarrow v}$$

$$\vdots$$

Checking definitions by (co)pattern matching

Agda core v0.1

From clauses to a case tree

From clauses to a case tree

The clauses guide us in the construction of a well-typed case tree:

From clauses to a case tree

The clauses guide us in the construction of a well-typed case tree: as we construct the case tree, we deconstruct the clauses.

From clauses to a case tree

The clauses guide us in the construction of a well-typed case tree: as we construct the case tree, we deconstruct the clauses.

$$\textsc{CheckLHS}(\Gamma \vdash u : A, \mathit{problem}) \Rightarrow Q$$

From clauses to a case tree

The clauses guide us in the construction of a well-typed case tree: as we construct the case tree, we deconstruct the clauses.

$$\textsc{CheckLHS}(\Gamma \vdash u : A, \mathit{problem}) \Rightarrow Q$$

$$\text{satisfying } \Gamma \mid u : A \vdash Q$$

CheckLHS: nothing to split

$$\frac{\textsc{NextSplit}(\mathit{problem}) \Rightarrow \textsc{SplittingDone}\ v}{\textsc{CheckLHS}(\Gamma \vdash u : A, \mathit{problem}) \Rightarrow \texttt{done}\ v}$$

$\textsc{CheckLHS}$: nothing to split

$$\frac{\textsc{NextSplit}(\mathit{problem}) \Rightarrow \textsc{SplittingDone}\ v}{\textsc{CheckLHS}(\Gamma \vdash u : A, \mathit{problem}) \Rightarrow \texttt{done}\ v}$$

Note: also update Σ with `clause` $u \mapsto v$!

CheckLHS: introduce new variable

$$\frac{\begin{array}{c}\textsc{NextSplit}(\mathit{problem}) \Rightarrow \textsc{SplitOnResult} \\ C \longrightarrow^{whnf} (x : A) \to B \\ \textsc{UpdateTarget}(x, \mathit{problem}) \Rightarrow \mathit{problem}' \\ \textsc{CheckLHS}(\Gamma(x : A) \vdash u\ x : B, \mathit{problem}') \Rightarrow Q\end{array}}{\textsc{CheckLHS}(\Gamma \vdash u : C, \mathit{problem}) \Rightarrow \mathtt{intro}_x\ Q}$$

CheckLHS: split on variable

$$
\frac{
\begin{array}{c}
\textsc{NextSplit}(\mathit{problem}) \Rightarrow \textsc{SplitOn}\ x \\
\Gamma = \Gamma_1(x : B)\Gamma_2 \qquad B \longrightarrow^{\mathsf{whnf}} \mathtt{D}\ \bar{v} \\
\left\{
\begin{array}{c}
\mathtt{constructor}\ \mathtt{c_i}\ \Delta_i : \mathtt{D}\ \Delta \in \Sigma \\
\Delta_i' = \Delta_i[\Delta \mapsto \bar{v}] \quad \sigma_i = [x \mapsto \mathtt{c_i}\ \Delta_i'] \\
\Gamma_i = \Gamma_1 \Delta_i \Gamma_2 \sigma \\
\textsc{UpdatePatterns}(\sigma_i, \mathit{problem}) \Rightarrow \mathit{problem}_i \\
\textsc{CheckLHS}(\Gamma_i \vdash u\sigma_i : A\sigma_i, \mathit{problem}_i) \Rightarrow Q_i
\end{array}
\right\}_{i=1\ldots n}
\end{array}
}{
\begin{array}{l}
\textsc{CheckLHS}(\Gamma \vdash u : A, \mathit{problem}) \\
\quad \Rightarrow \mathtt{split}_x \{\mathtt{c_1}\ \Delta_1 \mapsto Q_1; \ldots; \mathtt{c_n}\ \Delta_n \mapsto Q_n\}
\end{array}
}
$$

CheckLHS: split on result

$$\dfrac{\begin{array}{c}\textsc{NextSplit}(\mathit{problem}) \Rightarrow \textsc{SplitOnResult} \\ A \longrightarrow^{\mathsf{whnf}} \mathtt{R}\ \bar{v} \\ \left\{\begin{array}{c} \mathtt{projection}\ x : \mathtt{R}\ \Delta \vdash .\pi_{\mathtt{i}} : A_i \in \Sigma \\ \sigma_i = [\Delta \mapsto \bar{v}, x \mapsto u] \\ \textsc{UpdateTarget}(.\pi_{\mathtt{i}}, \mathit{problem}) \Rightarrow \mathit{problem}_i \\ \textsc{CheckLHS}(\Gamma \vdash u\ .\pi_{\mathtt{i}} : A_i\sigma, \mathit{problem}_i) \Rightarrow Q_i \end{array}\right\}_{i=1\ldots n}\end{array}}{\begin{array}{c}\textsc{CheckLHS}(\Gamma \vdash u : A, \mathit{problem}) \\ \Rightarrow \mathtt{cosplit}\ \{\pi_{\mathtt{1}} \mapsto Q_1; \ldots; \pi_{\mathtt{n}} \mapsto Q_n\}\end{array}}$$

CheckLHS: split on equation

$$\frac{\begin{array}{c}\textsc{NextSplit}(\mathit{problem}) \Rightarrow \textsc{SplitOn}\ e \\ \Gamma = \Gamma_1(x : B)\Gamma_2 \qquad B \longrightarrow^{\mathsf{whnf}} v \equiv_C w \\ \textsc{unify}(\Gamma_1, B, v, w) \Rightarrow (\Gamma_1', \sigma, \tau) \\ \sigma' = \sigma; [e \mapsto \texttt{refl}] \qquad \Gamma' = \Gamma_1'\Gamma_2\sigma \\ \textsc{UpdatePatterns}(\sigma', \mathit{problem}) \Rightarrow \mathit{problem}' \\ \textsc{CheckLHS}(\Gamma' \vdash u\sigma : A\sigma, \mathit{problem}') \Rightarrow Q\end{array}}{\textsc{CheckLHS}(\Gamma \vdash u : A, \mathit{problem}) \Rightarrow \texttt{eqsplit}_e\ \tau\ Q}$$

CheckLHS: split on absurd equation

$$\frac{\textsc{NextSplit}(\mathit{problem}) \Rightarrow \textsc{SplitOn}\ e \qquad \Gamma = \Gamma_1(x : B)\Gamma_2 \qquad B \longrightarrow^{\mathsf{whnf}} v \equiv_C w \qquad \textsc{unify}(\Gamma_1, B, v, w) \Rightarrow \bot}{\textsc{CheckLHS}(\Gamma \vdash u : A, \mathit{problem}) \Rightarrow \texttt{absurdsplit}_e}$$

Correctness of LHS checking

Theorem. Let $\mathit{problem} = \{\bar{q}_i \mapsto v_i\}_{i=1\dots n}$. If

- $\textsc{CheckLHS}(\Gamma \vdash \texttt{f} : A, \mathit{problem}) \Rightarrow Q$
- $\textsc{Match}(\bar{q}_j, \bar{e}) \Rightarrow \bot$ for $j = 1 \dots i-1$
- $\textsc{Match}(\bar{q}_i, \bar{e}) \Rightarrow \sigma$

then $\textsc{Eval}(Q, [], \bar{e}) \Rightarrow v_i\sigma$.