New Exhaustiveness & Redundancy Checker

George Karachalias

January 13, 2015

1 Syntax

Input patterns may either be variables x, constructor patterns $K\,\vec{p}$ or guards $\mathcal{Q}$. This means that we allow patterns and guards to be interleaved, and, henceforth, pattern vectors $\vec{p}$ in a match need not have the same length. Covered/uncovered vectors v are vectors that are generated and processed by the algorithm. These all the same length in a match but carry along a contraint Δ, which represents the conditions under which the specific vector can be matched. Δ includes:

- Term-level constraints (guards) $\mathcal{Q}$.
- Type-level constraints (we ignore them for the moment for simplicity reasons)
- Strictness annotations of the form $(e = \bot)$ which represent the fact that expression e is forced.

$p ::= x \mid K\,\vec{p} \mid \mathcal{Q}$	input patterns
$u ::= x \mid K\,\vec{u}$	covered/uncovered patterns
$v ::= \Delta \vdash \vec{u}$	covered/uncovered vectors
$s ::= \overline{v}$	covered/uncovered set of vectors
$\Delta, \mathcal{Q}$	guard expressions

Figure 1: Syntax

2 Main algorithm

The main algorithm works line-by-line, instead of the previous column-based approach. We initially consider the match non-exhaustive, and we represent this by a missing pattern vector that consists of wildcards (Δ equals to *True* since it is missing unconditionally):

$$initial\_missing = \{\mathit{True} \vdash (\_ \cdots \_)\}$$

Then, the algorithm repeatedly calls function *alg* (actually it calls *process_vector* below) that takes:

- An uncovered vector v
- A clause from the match $\vec{p}$.

And returns a triple that contains:

- The set of vectors that are covered by the clause $\vec{p}$.
- The set of vectors that remain uncovered, even after trying to match $\vec{p}$.
- The set of vectors that are eliminated by clause $\vec{p}$. By eliminated we mean that if the clause $\vec{p}$ pattern matches against an argument that has not been evaluated until now, it eliminates the possibility that this argument is equal to $\bot$, i.e., if the argument is $\bot$, matching will diverge at this specific point.

$$\boxed{alg :: v \to \vec{p} \to \langle s_c, s_u, s_\bot \rangle}$$

$$
\begin{array}{lcl}
alg(\Delta \vdash (), ()) & = & \langle \{\Delta \vdash ()\}, \varnothing, \varnothing \rangle \\
alg(\Delta \vdash u\,\vec{u}, x\,\vec{p}) & = & map^{\langle\rangle}\ (u :)\ (alg(\Delta \vdash \vec{u}, [x \mapsto u]\vec{p})) \\
alg(\Delta \vdash (K_i\,\vec{u})\,\vec{u'}, (K_j\,\vec{p})\,\vec{p'}) & = & \begin{cases} map^{\langle\rangle}\ (zip\_con\ K_i)\ (alg(\Delta \vdash \vec{u}\,\vec{u'}, \vec{p}\,\vec{p'})) & \text{, if } K_i = K_j \\ \langle \varnothing, \{\Delta \vdash (K_i\,\vec{u})\,\vec{u'}\}, \varnothing \rangle & \text{, otherwise} \end{cases} \\
alg(\Delta \vdash x\,\vec{u}, (K_j\,\vec{p})\,\vec{p'}) & = & \langle \varnothing, \varnothing, \{(\Delta \wedge (x = \bot) \vdash x\,\vec{u})\} \rangle\ \cup^{\langle\rangle}\ \left(\bigcup^{\langle\rangle}_{K_j}\ alg([x \mapsto K_j\,\overline{x}](\Delta \vdash x\,\vec{u}), (K_i\,\vec{p})\,\vec{p'}) \right) \\
alg(\Delta \vdash \vec{u}, \mathcal{Q}\,\vec{p}) & = & \langle \varnothing, \{\Delta \wedge \neg\mathcal{Q} \vdash \vec{u}\}, \{\Delta \wedge (\mathcal{Q} = \bot) \vdash \vec{u}\} \rangle \cup^{\langle\rangle}\ alg(\Delta \wedge \mathcal{Q} \vdash \vec{u}, \vec{p})
\end{array}
$$

Figure 2: Main function

Auxilary functions

$$\begin{aligned} map^{\langle\rangle}\ f\ \langle s_c, s_u, s_\perp\rangle &= \langle \{\Delta \vdash f(\vec{u}) \mid (\Delta \vdash \vec{u}) \in s_c\} \\ &\quad , \{\Delta \vdash f(\vec{u}) \mid (\Delta \vdash \vec{u}) \in s_u\} \\ &\quad , \{\Delta \vdash f(\vec{u}) \mid (\Delta \vdash \vec{u}) \in s_\perp\}\rangle \end{aligned}$$

$$\langle s_{c1}, s_{u1}, s_{\perp 1}\rangle\ \cup^{\langle\rangle}\ \langle s_{c2}, s_{u2}, s_{\perp 2}\rangle = \langle s_{c1} \cup s_{c2}, s_{u1} \cup s_{u2}, s_{\perp 1} \cup s_{\perp 2}\rangle$$

Function *process_vector*

Finally, function *process_vector* processes a whole set of uncovered vectors, simply calling *alg* and combining the results (by construction all vectors are disjoint so this is a disjoint union):

$$\boxed{process\_vector :: s_u \to \vec{p} \to \langle s_c, s_u, s_\perp\rangle}$$

$$process\_vector(\overline{v}, \vec{p}) = \bigcup_{v \in \overline{v}}^{\langle\rangle} alg(v, \vec{p})$$

The algorithm is shown graphically below, in Figure 3.

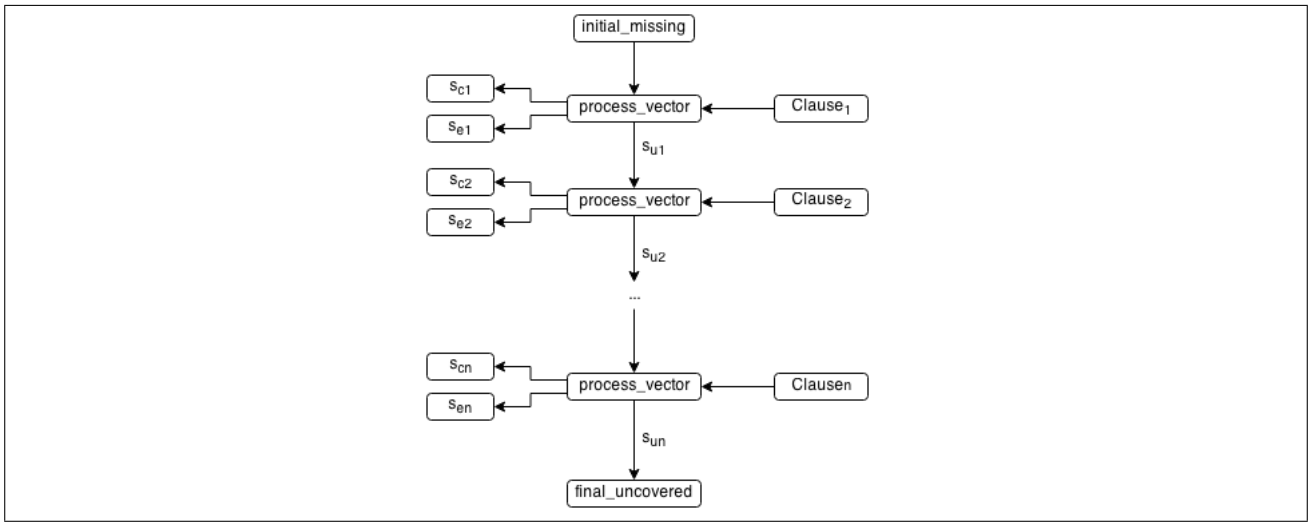

Figure 3: The algorithm