

# Constructing Set Constraints for ReScript

서울대학교 전기·정보공학부 2018-12602 이준협

## 1 Definition of set expressions

$se$	$::=$	$\emptyset$	<i>empty set</i>
		$-$	<i>maximum set</i>
		$()$	<i>unit</i>
		$n$	<i>integer</i>
		$b$	<i>boolean</i>
		$\lambda x.e$	<i>function</i>
		$V_e$	<i>set variable corresponding to the possible values of <math>e</math></i>
		$P_e$	<i>set variable corresponding to the possible exn packets of <math>e</math></i>
		$\text{body}_V(se)$	<i>values that <math>se</math> can spit out when <math>se</math> is applied to something</i>
		$\text{body}_P(se)$	<i>exn packets that <math>se</math> can spit out when <math>se</math> is applied to something</i>
		$\text{par}(se)$	<i>values that can be a parameter for <math>se</math></i>
		$\kappa$	<i>constructor</i>
		$l$	<i>field of a record</i>
		$\text{con}(\kappa, se)$	<i>construct</i>
		$\text{exn}(\kappa, se)$	<i>exception</i>
		$\text{fld}(se, l)$	<i>contents of the field <math>l</math> of a record <math>se</math></i>
		$\text{cnt}(se)$	<i>contents of a reference</i>
		$\text{bop}(se, se)$	<i>binary operators, where <math>\text{bop} \in \{+, -, \times, \div, =, &lt;, &gt;\}</math></i>
		$f_{(i)}^{-1}(se)$	<i>projection onto the <math>i</math>-th argument of <math>f</math></i>
		$se \cup se$	<i>union</i>
		$se \cap se$	<i>intersection</i>
		$\overline{se}$	<i>complement</i>

The definition of the conditional set expression needs clarification.

$$se_1 \Rightarrow se_2 := \begin{cases} \emptyset & (se_1 = \emptyset) \\ se_2 & (o.w) \end{cases}$$

The conditional set expression is a naive approximation for pattern matching. Consider the case when we want to match an expression against the record pattern with fields  $x$  and  $y$ . The constraints describing the record  $r = \{x = 1, y = 2\}$  are  $1 \subseteq \text{fld}(V_r, x) \wedge 2 \subseteq \text{fld}(V_r, y)$ . To pattern-match  $r$  against  $\{x, y\}$ , we want the fields  $x$  and  $y$  of  $V_r$  to be nonempty. Thus, the value of “match  $r$  with  $\{x, y\} \rightarrow e$ ” is  $\text{fld}(V_r, x) \Rightarrow (\text{fld}(V_r, y) \Rightarrow V_e)$ .

The conditional set expression can also be used to define conditional set constraints [1].

$$se \Rightarrow \bigwedge_{i=1}^n X_i \subseteq Y_i := \bigwedge_{i=1}^n (se \Rightarrow X_i) \subseteq Y_i$$

## 2 Constructing set constraints

Now we are in a position to define constraint construction rules for our ReScript-like language. Hopefully this would be reasonably fast when implemented and be accurate enough...

$$\begin{aligned}
 & [\text{UNIT, INT, BOOL}] \frac{}{\triangleright c : V_e \supseteq c} \quad c = (), n, b \\
 & [\text{APP}] \frac{\triangleright e_1 : C_1 \quad \triangleright e_2 : C_2}{\triangleright e_1 e_2 : (V_e \supseteq \text{body}_V(V_{e_1})) \wedge (P_e \supseteq (\text{body}_P(P_{e_1}) \cup P_{e_2})) \wedge (\text{par}(V_{e_1}) \supseteq V_{e_2}) \wedge C_1 \wedge C_2} \\
 & [\text{FN}] \frac{\triangleright e' : C'}{\triangleright \lambda x.e' : (V_e \supseteq \lambda x.e') \wedge (\text{body}_V(V_e) \supseteq V_{e'}) \wedge (\text{body}_P(V_e) \supseteq P_{e'}) \wedge (\text{par}(V_e) \subseteq V_x) \wedge C'}
 \end{aligned}$$

$$\begin{array}{c}
\text{[LET]} \frac{\triangleright e_1 : C_1 \quad \triangleright e_2 : C_2}{\triangleright \text{let } x = e_1 \text{ in } e_2 : (V_x \supseteq V_{e_1}) \wedge (V_e \supseteq V_{e_2}) \wedge (P_e \supseteq P_{e_1} \cup P_{e_2}) \wedge C_1 \wedge C_2} \\
\text{[BOP]} \frac{\triangleright e_1 : C_1 \quad \triangleright e_2 : C_2}{\triangleright e_1 \text{ bop } e_2 : (V_e \supseteq \text{bop}(V_{e_1}, V_{e_2})) \wedge (P_e \supseteq P_{e_1} \cup P_{e_2}) \wedge \underbrace{(P_e \supseteq (V_{e_2} \cap 0 \Rightarrow \text{exn}(\text{Divide\_by\_zero}, ())))}_{\text{when bop is } \div} \wedge C_1 \wedge C_2} \\
\text{[CON]} \frac{\triangleright e' : C'}{\triangleright \text{con } \kappa \ e' : (V_e \supseteq \text{con}(\kappa, V_{e'})) \wedge (P_e \supseteq P_{e'}) \wedge C'} \\
\text{[EXN]} \frac{\triangleright e' : C'}{\triangleright \text{exn } \kappa \ e' : (V_e \supseteq \text{exn}(\kappa, V_{e'})) \wedge (P_e \supseteq P_{e'}) \wedge C'} \\
\text{[DECON]} \frac{\triangleright e' : C'}{\triangleright \text{decon}(\kappa, e') : (V_e \supseteq \text{con}_{(2)}^{-1}(V_{e'} \cap \text{con}(\kappa, \_)) \cup \text{exn}_{(2)}^{-1}(V_{e'})) \wedge (P_e \supseteq P_{e'}) \wedge C'} \\
\text{[RECORD]} \frac{\triangleright e_i : C_i \ (1 \leq i \leq n)}{\triangleright \{l_1 = e_i, \dots, l_n = e_n\} : \bigwedge_{i=1}^n (\text{fld}(V_e, l_i) \supseteq V_{e_i}) \wedge (P_e \supseteq \bigcup_{i=1}^n P_{e_i}) \wedge \bigwedge_{i=1}^n C_i} \\
\text{[FIELD]} \frac{\triangleright e' : C'}{\triangleright e'.l : (V_e \supseteq \text{fld}(V_{e'}, l)) \wedge (P_e \supseteq P_{e'}) \wedge C'} \\
\text{[REF]} \frac{\triangleright e' : C'}{\triangleright \text{ref } e' : (V_e \supseteq \text{loc}) \wedge (\text{cnt}(\text{loc}) \supseteq V_{e'}) \wedge (P_e \supseteq P_{e'}) \wedge C'} \text{ new loc} \\
\text{[UPDATE]} \frac{\triangleright e_1 : C_1 \quad \triangleright e_2 : C_2}{\triangleright e_1 := e_2 : (\text{cnt}(V_{e_1}) \supseteq V_{e_2}) \wedge (P_e \supseteq P_{e_1} \cup P_{e_2}) \wedge C_1 \wedge C_2} \\
\text{[BANG]} \frac{\triangleright e' : C'}{\triangleright !e' : (V_e \supseteq \text{cnt}(V_{e'})) \wedge (P_e \supseteq P_{e'}) \wedge C'}
\end{array}$$

Function for generating constraints out of pattern matching.

If we want to figure out the constraint for the value  $X$  of “match  $Y$  with  $p \rightarrow e$ ”:

$$\text{case}(X, Y, p, e) := \begin{cases} (Y \subseteq V_x) \wedge (V_e \subseteq X) & (p = x) \\ (\text{fld}(Y, l_1) \Rightarrow \dots \Rightarrow \text{fld}(Y, l_n) \Rightarrow V_e) \subseteq X & (p = \{l_i\}_{i=1}^n) \\ (\kappa \cap (\text{con}_{(1)}^{-1}(Y) \cup \text{exn}_{(1)}^{-1}(Y))) \Rightarrow V_e \subseteq X & (p = \kappa) \\ (Y \cap c) \Rightarrow V_e \subseteq X & (p = \text{constant}) \\ V_e \subseteq X & (p = \_) \end{cases}$$

$$\begin{array}{c}
\text{[CASE]} \frac{\triangleright e' : C' \quad \triangleright e_i : C_i \ (1 \leq i \leq n)}{\triangleright \text{case } e' \ (p_i \rightarrow e_i)_{i=1}^n : \bigwedge_{i=1}^n \text{case}(V_e, V_{e'}, p_i, e_i) \wedge (P_e \supseteq \bigcup_{i=1}^n P_{e_i} \cup P_{e'}) \wedge C' \wedge \bigwedge_{i=1}^n C_i} \\
\text{[HANDLE]} \frac{\triangleright e' : C' \quad \triangleright e_i : C_i \ (1 \leq i \leq n)}{\triangleright \text{handle } e' \ (p_i \rightarrow e_i)_{i=1}^n : (P_e \supseteq (P_{e'} \cap \bigcap_{i=1}^n \overline{\text{exn}(p_i, \_)})) \cup \bigcup_{i=1}^n P_{e_i} \wedge (V_e \supseteq V_{e'} \cup \bigcup_{i=1}^n V_{e_i})}
\end{array}$$

## References

- [1] Alexander Aiken. “Introduction to set constraint-based program analysis”. In: *Science of Computer Programming* 35.2 (1999), pp. 79–111. issn: 0167-6423. doi: [https://doi.org/10.1016/S0167-6423\(99\)00007-6](https://doi.org/10.1016/S0167-6423(99)00007-6).