Constructing Set Constraints for ReScript

서울대학교 전기·정보공학부 이재호, 이준협

1 Definition of set expressions

se	$::=$	$\emptyset$	*empty set*
	$\mid$	$\_$	*maximum set*
	$\mid$	$()$	*unit*
	$\mid$	n	*integer*
	$\mid$	b	*boolean*
	$\mid$	$\langle \lambda x.e, \sigma \rangle$	*closure*
	$\mid$	loc	*memory location*
	$\mid$	$V(e, \sigma)$	*set variable corresponding to the possible values of e under* σ
	$\mid$	$P(e, \sigma)$	*set variable corresponding to the possible exn packets of e under* σ
	$\mid$	$\mathsf{app}_V(se, e)$	*values that se can spit out when se is applied to e*
	$\mid$	$\mathsf{app}_P(se, e)$	*exn packets that se can spit out when se is applied to e*
	$\mid$	$\mathsf{con}(\kappa, se)$	*construct*
	$\mid$	$\mathsf{exn}(\kappa, se)$	*exception*
	$\mid$	$\mathsf{fld}(se, l)$	*contents of the field l of a record se*
	$\mid$	$\mathsf{cnt}(se)$	*contents of a reference*
	$\mid$	$\mathsf{bop}(se, se)$	*binary operators, where* bop $\in \{+, -, \times, \div, =, <, >\}$
	$\mid$	$f_{(i)}^{-1}(se)$	*projection onto the i-th argument of f*
	$\mid$	$se \cup se$	*union*
	$\mid$	$se \cap se$	*intersection*
	$\mid$	$\overline{se}$	*complement*
	$\mid$	$se \Rightarrow se$	*conditional expression*

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 $1 \subseteq \mathsf{fld}(V(r, \sigma_r), x) \wedge 2 \subseteq \mathsf{fld}(V(r, \sigma_r), y)$ with $\sigma_r = [r \mapsto \{x = 1, y = 2\}]$. To pattern-match r against $\{x, y\}$, we want the fields x and y of $V(r, \sigma_r)$ to be nonempty. Thus, the value of "*match r with* $\{x, y\} \to e$" is $\mathsf{fld}(V(r, \sigma_r), x) \Rightarrow (\mathsf{fld}(V(r, \sigma_r), y) \Rightarrow V(e, \sigma_e))$.

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

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

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...

Notation: $\sigma_i := \sigma|_{e_i}$, E_x: expression variable, Σ_x: environment variable, all *Expr*s are tagged with their location

$$[\text{UNIT, INT, BOOL}] \; \frac{}{[] \rhd c : V(e, \sigma) \supseteq c} \; c = (), n, b$$

$$[\text{APP}] \; \frac{\sigma_1 \rhd e_1 : C_1 \qquad \sigma_2 \rhd e_2 : C_2}{\begin{aligned} \sigma \rhd e_1\, e_2 : \quad & V(e, \sigma) \supseteq \mathsf{app}_V(V(e_1, \sigma_1), e_2) \wedge \\ & P(e, \sigma) \supseteq \mathsf{app}_P(V(e_1, \sigma_1), e_2) \cup P(e_1, \sigma_1) \cup P(e_2, \sigma_2) \wedge C_1 \wedge C_2 \end{aligned}}$$

$$[\text{FN}]\ \frac{\sigma \cup [x \mapsto E_x] \rhd e' : C'}{\begin{array}{ll}\sigma \rhd \lambda x.e' : V(e,\sigma) \supseteq \langle \lambda x.e', \sigma\rangle \wedge & \mathsf{app}_V(\langle \lambda x.e', \sigma\rangle, E_x) \supseteq V(e', [x \mapsto E_x] \cup \sigma) \wedge \\ & \mathsf{app}_P(\langle \lambda x.e', \sigma\rangle, E_x) \supseteq P(e', [x \mapsto E_x] \cup \sigma) \wedge C'\end{array}}$$

$$[\text{VAR}]\ \frac{}{\sigma \rhd x : V(x,\sigma) \supseteq V(\sigma(x), \Sigma_x)}$$

$$[\text{LET}]\ \frac{\sigma_1 \cup \overbrace{[x \mapsto e_1]}^{\text{support rec}} \rhd e_1 : C_1 \qquad \sigma_2 \cup [x \mapsto e_1] \rhd e_2 : C_2}{\begin{array}{ll}\sigma \rhd \text{let } x = e_1 \text{ in } e_2 : & V(e,\sigma) \supseteq V(e_2, \sigma_2 \cup [x \mapsto e_1]) \wedge \\ & P(e,\sigma) \supseteq P(e_1, \sigma_1 \cup [x \mapsto e_1]) \cup P(e_2, \sigma_2 \cup [x \mapsto e_1]) \wedge C_1 \wedge C_2\end{array}}$$

$$[\text{BOP}]\ \frac{\sigma_1 \rhd e_1 : C_1 \qquad \sigma_2 \rhd e_2 : C_2}{\begin{array}{ll}\sigma \rhd e_1 \text{ bop } e_2 : & V(e,\sigma) \supseteq \mathsf{bop}(V(e_1,\sigma_1), V(e_2,\sigma_2)) \wedge \\ & P(e,\sigma) \supseteq P(e_1,\sigma_1) \cup P(e_2,\sigma_2) \wedge \underbrace{P(e,\sigma) \supseteq (V(e_2,\sigma_2) \cap 0 \Rightarrow \mathsf{exn}(\mathsf{Divide\_by\_zero}, ()))}_{\text{when bop is } \div} \wedge C_1 \wedge C_2\end{array}}$$

$$[\text{CON}]\ \frac{\sigma \rhd e' : C'}{\sigma \rhd \mathsf{con}\ \kappa\ e' : V(e,\sigma) \supseteq \mathsf{con}(\kappa, V(e',\sigma)) \wedge P(e,\sigma) \supseteq P(e',\sigma) \wedge C'}$$

$$[\text{EXN}]\ \frac{\sigma \rhd e' : C'}{\sigma \rhd \mathsf{exn}\ \kappa\ e' : V(e,\sigma) \supseteq \mathsf{exn}(\kappa, V(e',\sigma)) \wedge P(e,\sigma) \supseteq P(e',\sigma) \wedge C'}$$

$$[\text{DECON}]\ \frac{\sigma \rhd e' : C'}{\begin{array}{ll}\sigma \rhd \mathsf{decon}\ \kappa\ e' : & V(e,\sigma) \supseteq \mathsf{con}^{-1}_{(2)}(\underbrace{V(e',\sigma) \cap \mathsf{con}(\kappa,\_)}_{\text{filter}}) \cup \mathsf{exn}^{-1}_{(2)}(\underbrace{V(e',\sigma) \cap \mathsf{exn}(\kappa,\_)}_{\text{filter}}) \wedge \\ & P(e,\sigma) \supseteq P(e',\sigma) \wedge C'\end{array}}\ \text{as in SML-NJ's Lambda}$$

$$[\text{RECORD}]\ \frac{\sigma_i \rhd e_i : C_i\ (1 \le i \le n)}{\sigma \rhd \{l_1 = e_i, ..., l_n = e_n\} : \bigwedge_{i=1}^{n} (\mathsf{fld}(V(e,\sigma), l_i) \supseteq V(e_i,\sigma_i)) \wedge P(e,\sigma) \supseteq \bigcup_{i=1}^{n} P(e_i,\sigma_i) \wedge \bigwedge_{i=1}^{n} C_i}$$

$$[\text{FIELD}]\ \frac{\sigma \rhd e' : C'}{\sigma \rhd e'.l : V(e,\sigma) \supseteq \mathsf{fld}(V(e',\sigma), l) \wedge P(e,\sigma) \supseteq P(e',\sigma) \wedge C'}$$

$$[\text{REF}]\ \frac{\sigma \rhd e' : C'}{\sigma \rhd \mathsf{ref}\ e' : V(e,\sigma) \supseteq \mathsf{loc} \wedge \mathsf{cnt}(\mathsf{loc}) \supseteq V(e',\sigma) \wedge P(e,\sigma) \supseteq P(e',\sigma) \wedge C'}\ \text{new loc}$$

$$[\text{UPDATE}]\ \frac{\sigma_1 \rhd e_1 : C_1 \qquad \sigma_2 \rhd e_2 : C_2}{\sigma \rhd e_1 := e_2 : \mathsf{cnt}(V(e_1,\sigma_1)) \supseteq V(e_2,\sigma_2) \wedge P_e \supseteq P(e_1,\sigma_1) \cup P(e_2,\sigma_2) \wedge C_1 \wedge C_2}$$

$$[\text{BANG}]\ \frac{\sigma \rhd e' : C'}{\sigma \rhd !e' : V(e,\sigma) \supseteq \mathsf{cnt}(V(e',\sigma)) \wedge P(e,\sigma) \supseteq P(e',\sigma) \wedge C'}$$

$$[\text{SEQ}]\ \frac{\sigma_1 \rhd e_1 : C_1 \qquad \sigma_2 \rhd e_2 : C_2}{\sigma \rhd e_1; e_2 : V(e,\sigma) \supseteq V(e_2,\sigma_2) \wedge P(e,\sigma) \supseteq P(e_1,\sigma_1) \cup P(e_2,\sigma_2) \wedge C_1 \wedge C_2}$$

We define an auxilary function for generating constraints out of pattern matching. If we want to figure out the constraint for the expression $e =$ "match e' with $p \to e''$" under σ:

$$\mathsf{case}(e', \sigma, p, e'') := \begin{cases} V(e,\sigma) \supseteq V(e'', \sigma|_{e''} \cup [x \mapsto e']) & (p = x) \\ V(e,\sigma) \supseteq V(e'', \sigma|_{e''}) & (o.w) \end{cases}$$

$$[\text{CASE}]\ \frac{\sigma|_{e'} \rhd e' : C' \qquad \sigma_i \rhd e_i : C_i\ (1 \le i \le n)}{\sigma \rhd \mathsf{case}\ e'\ (p_i \to e_i)_{i=1}^{n} : \bigwedge_{i=1}^{n} \mathsf{case}(e', \sigma, p_i, e_i) \wedge P(e,\sigma) \supseteq \bigcup_{i=1}^{n} P(e_i,\sigma_i) \cup P(e', \sigma|_{e'}) \wedge C' \wedge \bigwedge_{i=1}^{n} C_i}$$

$$[\text{HANDLE}]\ \frac{\sigma|_{e'} \rhd e' : C' \qquad \sigma_i \rhd e_i : C_i\ (1 \le i \le n)}{\begin{array}{ll}\sigma \rhd \mathsf{handle}\ e'\ (p_i \to e_i)_{i=1}^{n} : & P(e,\sigma) \supseteq (P(e', \sigma|_{e'}) \cap \bigcap_{i=1}^{n} \overline{\mathsf{exn}(p_i,\_)}) \cup \bigcup_{i=1}^{n} P(e_i,\sigma_i) \wedge \\ & V(e,\sigma) \supseteq V(e', \sigma|_{e'}) \cup \bigcup_{i=1}^{n} V(e_i,\sigma_i) \wedge \\ & C' \wedge \bigwedge_{i=1}^{n} C_i\end{array}}$$

$$[\text{RAISE}]\ \frac{\sigma \rhd e' : C'}{\sigma \rhd \mathsf{raise}\ e' : P(e,\sigma) \supseteq V(e',\sigma) \cup P(e',\sigma) \wedge C'}$$

$$[\text{FOR}]\ \frac{\sigma_1 \rhd e_1 : C_1 \quad \sigma_2 \rhd e_2 : C_2 \quad \sigma_3 \cup [x \mapsto e_1] \rhd e_3 : C_3 \quad \sigma_3 \cup [x \mapsto x+1] \rhd e_3 : C_4}{\begin{array}{ll} \sigma \rhd \mathsf{for}\ x\ e_1\ e_2\ e_3 : & V(e,\sigma) \supseteq () \wedge V(x,[x \mapsto e_1]) \supseteq V(e_1,\sigma_1) \wedge \\ & V(x,[x \mapsto x+1]) \supseteq +(V(e_1,\sigma_1),1) \cup +(V(x,[x \mapsto x+1]),1) \wedge \\ & P(e,\sigma) \supseteq P(e_1,\sigma_1) \cup P(e_2,\sigma_2) \cup P(e_3,\sigma_3 \cup [x \mapsto e_1]) \cup P(e_3,\sigma_3 \cup [x \mapsto x+1]) \wedge \\ & C_1 \wedge C_2 \wedge C_3 \wedge C_4 \end{array}}$$

$$[\text{WHILE}]\ \frac{\sigma_1 \rhd e_1 : C_1 \quad \sigma_2 \rhd e_2 : C_2}{\sigma \rhd \mathsf{while}\ e_1\ e_2 : V(e,\sigma) \supseteq () \wedge P(e,\sigma) \supseteq P(e_1,\sigma_1) \cup P(e_2,\sigma_2) \wedge C_1 \wedge C_2}$$

3 Example

$$\underbrace{\text{let } f = \overbrace{\lambda x.\, \underbrace{\text{Error } \overbrace{(\underbrace{f}_{e''_{11}}\ \underbrace{x}_{e''_{12}})}^{e''_1}}_{e'_1}}^{e_1} \text{ in } \overbrace{\text{raise } \underbrace{(\overbrace{f}^{e'_{21}}\ \overbrace{\text{Fail}}^{e'_{22}})}_{e'_2}}^{e_2}}_{e}$$

The above program is type checked as $f : \mathsf{exn} \to \mathsf{exn}$, yet it does not terminate. The (simplified) set constraints generated for this program are:

Constraint	From	By	No.
$P(e,[]) \supseteq P(e_2,[f \mapsto e_1])$	e	[LET]	(1)
$V(e_1,[f \mapsto e_1]) \supseteq \langle \lambda x.e'_1, [f \mapsto e_1] \rangle$	e_1	[FN]	(2)
$\mathsf{app}_V(\langle \lambda x.e'_1, [f \mapsto e_1] \rangle, E_x) \supseteq V(e'_1,[f \mapsto e_1; x \mapsto E_x])$	e_1	[FN]	(3)
$V(e'_1,[f \mapsto e_1; x \mapsto E_x]) \supseteq \mathsf{exn}(\mathrm{Error}, V(e''_1,[f \mapsto e_1; x \mapsto E_x]))$	e'_1	[EXN]	(4)
$V(e''_1,[f \mapsto e_1; x \mapsto E_x]) \supseteq \mathsf{app}_V(V(e''_{11},[f \mapsto e_1]), e''_{12})$	e''_1	[APP]	(5)
$V(e''_{11},[f \mapsto e_1]) \supseteq V(e_1,\Sigma_f)$	e''_{11}	[VAR]	(6)
$V(e''_{12},[x \mapsto E_x]) \supseteq V(E_x,\Sigma_x)$	e''_{12}	[VAR]	(7)
$P(e_2,[f \mapsto e_1]) \supseteq V(e'_2,[f \mapsto e_1])$	e_2	[RAISE]	(8)
$V(e'_2,[f \mapsto e_1]) \supseteq \mathsf{app}_V(V(e'_{21},[f \mapsto e_1]), e'_{22})$	e'_2	[APP]	(9)
$V(e'_{21},[f \mapsto e_1]) \supseteq V(e_1,\Sigma_f)$	e'_{21}	[VAR]	(10)
$V(e'_{22},[]) \supseteq \mathsf{exn}(\mathrm{Fail}, ())$	e'_{22}	[EXN]	(11)

To "solve" this system of constraints, multiple reduction steps are needed.

1.

$$\begin{aligned} V(e'_2,[f \mapsto e_1]) &\supseteq \mathsf{app}_V(V(e'_{21},[f \mapsto e_1]), e'_{22}) && (\because (9)) \\ &\supseteq \mathsf{app}_V(V(e_1,\Sigma_f), e'_{22}) && (\because (10)) \\ &\supseteq \mathsf{app}_V(V(e_1,[f \mapsto e_1]), e'_{22}) && (\text{unify with } (2)) \\ &\supseteq \mathsf{app}_V(\langle \lambda x.e'_1, [f \mapsto e_1] \rangle, e'_{22}) && (\because (2)) \\ &\supseteq V(e'_1,[f \mapsto e_1; x \mapsto e'_{22}]) && (\because (3)) \end{aligned}$$

2.

$$V(e'_1,[f \mapsto e_1; x \mapsto e'_{22}]) \supseteq \mathsf{exn}(\mathrm{Error}, V(e''_1,[f \mapsto e_1; x \mapsto e'_{22}])) \qquad (\because (4))$$

3.

$$\begin{aligned} V(e''_1,[f \mapsto e_1; x \mapsto e'_{22}]) &\supseteq \mathsf{app}_V(V(e''_{11},[f \mapsto e_1]), e''_{12}) && (\because (5)) \\ &\supseteq \mathsf{app}_V(V(e_1,\Sigma_f), e''_{12}) && (\because (6)) \\ &\supseteq \mathsf{app}_V(V(e_1,[f \mapsto e_1]), e''_{12}) && (\text{unify with } (2)) \\ &\supseteq \mathsf{app}_V(\langle \lambda x.e'_1, [f \mapsto e_1] \rangle, e''_{12}) && (\because (2)) \\ &\supseteq V(e'_1,[f \mapsto e_1; x \mapsto e''_{12}]) && (\because (3)) \end{aligned}$$

4.

$$V(e'_1,[f \mapsto e_1; x \mapsto e''_{12}]) \supseteq \mathsf{exn}(\mathrm{Error}, V(e''_1,[f \mapsto e_1; x \mapsto e''_{12}])) \qquad (\because (4))$$

5.

$$\begin{aligned} V(e_1'', [f \mapsto e_1; x \mapsto e_{12}'']) &\supseteq \mathsf{app}_V(V(e_{11}'', [f \mapsto e_1]), e_{12}'') && (\because (5)) \\ &\supseteq V(e_1', [f \mapsto e_1; x \mapsto e_{12}'']) && (\text{as in 3}) \end{aligned}$$

6. Cannot reduce further as $V(e_1', [f \mapsto e_1; x \mapsto e_{12}''])$ was already reduced in 4.

Let $X_1 := V(e_1', [f \mapsto e_1; x \mapsto e_{22}'])$, $X_2 := V(e_1'', [f \mapsto e_1; x \mapsto e_{22}'])$, $X_3 := V(e_1', [f \mapsto e_1; x \mapsto e_{12}''])$, $X_4 := V(e_1'', [f \mapsto e_1; x \mapsto e_{12}''])$. Then the equations reduce to:

$$\begin{aligned} P(e, []) \supseteq P(e_2, [f \mapsto e_1]) &\supseteq V(e_2', [f \mapsto e_1]) && (\because (1), (8)) \\ V(e_2', [f \mapsto e_1]) &\supseteq X_1 && (\because 1) \\ \Rightarrow P(e, []) &\supseteq X_1 \\ X_1 &\supseteq \mathsf{exn}(\mathsf{Error}, X_2) && (\because 2) \\ X_2 &\supseteq X_3 && (\because 3) \\ X_3 &\supseteq \mathsf{exn}(\mathsf{Error}, X_4) && (\because 4) \\ X_4 &\supseteq X_3 && (\because 5) \end{aligned}$$

Note that constraint (7) was not used in the derivation of the above relations. This reflects the fact that the program does not terminate. There is absolutely no execution path evaluating what “Fail” is.

4 Why σ?

Wrapper function around raise. e is actually the location in code. Need a hash table between e and location.

TODO:

1. How to solve the set constraints(first in the case when there are no conditional expressions)：based on the work of Nevin Heintze[2]
2. Maybe, how to formulate the “parallel worlds” approach?

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.

[2] Nevin Heintze. *A Decision Procedure for a Class of Set Constraints*. Tech. rep. 1991.