Constructing Set Constraints for ReScript

서울대학교 전기·정보공학부 2018-12602 이준협

1 Definition of set expressions

se	$::=$	$\emptyset$	*empty set*
	$\mid$	$\_$	*maximum set*
	$\mid$	$()$	*unit*
	$\mid$	n	*integer*
	$\mid$	b	*boolean*
	$\mid$	$\lambda x.e$	*function*
	$\mid$	V_e	*set variable corresponding to the possible values of e*
	$\mid$	P_e	*set variable corresponding to the possible exn packets of e*
	$\mid$	$\mathsf{body}_V(se)$	*values that se can spit out when se is applied to something*
	$\mid$	$\mathsf{body}_P(se)$	*exn packets that se can spit out when se is applied to something*
	$\mid$	$\mathsf{par}(se)$	*values that can be a parameter for se*
	$\mid$	κ	*constructor*
	$\mid$	l	*field of a record*
	$\mid$	$\mathsf{con}(\kappa, se)$	*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* $\mathsf{bop} \in \{+, -, \times, \div, =, <, >\}$
	$\mid$	$f^{-1}_{(i)}(se)$	*projection onto the i-th argument of f*
	$\mid$	$se \cup se$	*union*
	$\mid$	$se \cap se$	*intersection*
	$\mid$	$\overline{se}$	*complement*

Case expressions need special concern when building set constraints. For example, take a look at the for loop.

```
let for x = match x > e2 with
            | true -> ()
            | false -> e3; for (x + 1)
in
for e1
```

The above program is equivalent to `for x = e1 to e2 do e3 done`. Translating this is difficult as case statements partition the program states. That is, "`e3; for (x + 1)`" is evaluated under the constraint that $>(V_x, V_{e_2}) \subseteq \texttt{false}$ and "`()`" is evaluated under the constraint that $>(V_x, V_{e_2}) \subseteq \texttt{true}$.

These constraints obviously cannot be and-ed together, as the result is trivially `false`. Since the constraints above partition the program states, it is straightforward that each set expression must have different "versions" of itself in each partition.

So each case statement creates a parallel world where some set constraint becomes true.

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...

$$\text{UNIT, INT, BOOL}\ \frac{}{\rhd c : V_e \supseteq c}\ c = (), n, b$$

$$\text{APP}\ \frac{\rhd e_1 : C_1 \qquad \rhd e_2 : C_2}{\rhd e_1\ e_2 : (V_e \supseteq \mathsf{body}_V(V_{e_1})) \wedge (P_e \supseteq (\mathsf{body}_P(P_{e_1}) \cup P_{e_1} \cup P_{e_2})) \wedge (\mathsf{par}(V_{e_1}) \supseteq V_{e_2}) \wedge C_1 \wedge C_2}$$

$$\text{FN}\ \frac{\rhd e' : C'}{\rhd \lambda x.e' : (V_e \supseteq \lambda x.e') \wedge (\mathsf{body}_V(V_e) \supseteq V_{e'}) \wedge (\mathsf{body}_P(V_e) \supseteq P_{e'}) \wedge (\mathsf{par}(V_e) \subseteq V_x) \wedge C'}$$

$$\text{LET}\ \frac{\triangleright e_1 : C_1 \qquad \triangleright e_2 : C_2}{\triangleright \mathit{let}\ x = e_1\ \mathit{in}\ e_2 : (V_x \supseteq V_{e_1}) \wedge (P_x \supseteq P_{e_2}) \wedge}$$

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.