Translation Sketch

1 Expressions

E translates a Chalice expression into an equivalent SIL expression. Given an scoped identifier i, $\rho(i)$ denotes a globally unique identifier. E.g., $\rho(\texttt{someField}) = \texttt{SomeClass::someField}$

$$\begin{align}
E\,[\![e_1 \texttt{?}\, e_2 \texttt{:}\, e_3]\!]_{\text{Ch}} &= [\![\texttt{if } E\,(e_1) \texttt{ then } E\,(e_2) \texttt{ else } E\,(e_3)]\!]_{\text{SIL}} \tag{1}\\
E\,[\![e_1 \texttt{==}\, e_2]\!]_{\text{Ch}} &= [\![\texttt{==}\,(E\,(e_1)\,\texttt{,}\,E\,(e_2))]\!]_{\text{SIL}} \tag{2}\\
E\,[\![e_1 \texttt{!=}\, e_2]\!]_{\text{Ch}} &= [\![\texttt{!=}\,(E\,(e_1)\,\texttt{,}\,E\,(e_2))]\!]_{\text{SIL}} \tag{3}\\
E\,[\![e_1 \texttt{<}\, e_2]\!]_{\text{Ch}} &= [\![\texttt{<}\,(E\,(e_1)\,\texttt{,}\,E\,(e_2))]\!]_{\text{SIL}} \tag{4}\\
E\,[\![e_1 \texttt{<=}\, e_2]\!]_{\text{Ch}} &= [\![\texttt{<=}\,(E\,(e_1)\,\texttt{,}\,E\,(e_2))]\!]_{\text{SIL}} \tag{5}\\
E\,[\![e_1 \texttt{>=}\, e_2]\!]_{\text{Ch}} &= [\![\texttt{>=}\,(E\,(e_1)\,\texttt{,}\,E\,(e_2))]\!]_{\text{SIL}} \tag{6}\\
E\,[\![e_1 \texttt{>}\, e_2]\!]_{\text{Ch}} &= [\![\texttt{>}\,(E\,(e_1)\,\texttt{,}\,E\,(e_2))]\!]_{\text{SIL}} \tag{7}\\
E\,[\![e_1 \texttt{«}\, e_2]\!]_{\text{Ch}} &= \texttt{/* lock below ??? */} \tag{8}\\
E\,[\![e_1 \texttt{ in } e_2]\!]_{\text{Ch}} &= [\![\texttt{in}\,(E\,(e_1)\,\texttt{,}\,E\,(e_2))]\!]_{\text{SIL}} \tag{9}\\
E\,[\![e_1 \texttt{ !in } e_2]\!]_{\text{Ch}} &= E\,[\![\texttt{!}\,(e_1 \texttt{ in } e_2)]\!]_{\text{Ch}} \tag{10}\\
E\,[\![e_1 \texttt{+}\, e_2]\!]_{\text{Ch}} &= [\![\texttt{+}\,(E\,(e_1)\,\texttt{,}\,E\,(e_2))]\!]_{\text{SIL}} \tag{11}\\
E\,[\![e_1 \texttt{-}\, e_2]\!]_{\text{Ch}} &= [\![\texttt{-}\,(E\,(e_1)\,\texttt{,}\,E\,(e_2))]\!]_{\text{SIL}} \tag{12}\\
E\,[\![e_1 \texttt{*}\, e_2]\!]_{\text{Ch}} &= [\![\texttt{*}\,(E\,(e_1)\,\texttt{,}\,E\,(e_2))]\!]_{\text{SIL}} \tag{13}\\
E\,[\![e_1 \texttt{/}\, e_2]\!]_{\text{Ch}} &= [\![\texttt{/}\,(E\,(e_1)\,\texttt{,}\,E\,(e_2))]\!]_{\text{SIL}} \tag{14}\\
E\,[\![e_1 \texttt{\%}\, e_2]\!]_{\text{Ch}} &= [\![\texttt{\%}\,(E\,(e_1)\,\texttt{,}\,E\,(e_2))]\!]_{\text{SIL}} \tag{15}\\
E\,[\![\texttt{!}\, e]\!]_{\text{Ch}} &= [\![\texttt{!}\,(E\,(e))]\!]_{\text{SIL}} \tag{16}\\
E\,[\![\texttt{-}\, e]\!]_{\text{Ch}} &= [\![\texttt{-}\,(E\,(e))]\!]_{\text{SIL}} \tag{17}\\
E\,[\![e_1 \texttt{[}\, e_2 \texttt{]}]\!]_{\text{Ch}} &= [\![\texttt{at}\,(E\,(e_1)\,\texttt{,}\,E\,(e_2))]\!]_{\text{SIL}} \tag{18}\\
E\,[\![e_1 \texttt{[}\, e_2 \texttt{..]}]\!]_{\text{Ch}} &= [\![\texttt{drop}\,(E\,(e_1)\,\texttt{,}\,E\,(e_2))]\!]_{\text{SIL}} \tag{19}\\
E\,[\![e_1 \texttt{[..}\, e_2 \texttt{]}]\!]_{\text{Ch}} &= [\![\texttt{take}\,(E\,(e_1)\,\texttt{,}\,E\,(e_2))]\!]_{\text{SIL}} \tag{20}\\
E\,[\![e_1 \texttt{.}\, id]\!]_{\text{Ch}} &= [\![E\,(e_1)\,\texttt{.}\,id]\!]_{\text{SIL}} \quad \text{Also assert that } e_1 \text{ is not null} \tag{21}\\
E\,[\![e_1 \texttt{.}\, id(e \ldots)]\!]_{\text{Ch}} &= [\![\rho(id)(E\,(e_1)\,\texttt{,}\,E\,(e \ldots))]\!]_{\text{SIL}} \quad \text{Also assert that } e_1 \text{ is not null} \tag{22}\\
E\,[\![\texttt{true}]\!]_{\text{Ch}} &= [\![\texttt{true}]\!]_{\text{SIL}} \tag{23}\\
E\,[\![\texttt{false}]\!]_{\text{Ch}} &= [\![\texttt{false}]\!]_{\text{SIL}} \tag{24}\\
E\,[\![\texttt{null}]\!]_{\text{Ch}} &= [\![\texttt{null}]\!]_{\text{SIL}} \tag{25}\\
E\,[\![\texttt{this}]\!]_{\text{Ch}} &= \texttt{/* ??? */} \tag{26}\\
E\,[\![x]\!]_{\text{Ch}} &= [\![x]\!]_{\text{SIL}} \qquad \text{where } x \text{ is a numeric literal} \tag{27}
\end{align}$$

2 Statements

S translates Chalice statements into equivalent SIL statements/expressions. $\tau_1, \tau_2, \ldots$ are temporary variables unique to each rule instantiation. Similarly, $\eta_1, \eta_2, \ldots$ are labels unique to each rule instantiation.

$$S\,[\![\texttt{assert}\ e_1]\!]_{\text{Ch}} = [\![\texttt{assert}\ E\,(e_1)]\!]_{\text{SIL}} \tag{28}$$

$$S\,[\![\texttt{assume}\ e_1]\!]_{\text{Ch}} = [\![\texttt{assume}\ E\,(e_1)]\!]_{\text{SIL}} \tag{29}$$

$$S\,[\![\{s \ldots\}]\!]_{\text{Ch}} = [\![S\,(s \ldots)]\!]_{\text{SIL}} \tag{30}$$

Need to flatten nested stmt blocks, since SIL doesn't have local variable scoping.

$$S\,[\![\ \texttt{spec}\ id\,[e \ldots]]\!]_{\text{Ch}} = \text{What does } \texttt{spec} \text{ do?} \tag{31}$$

$$S\,[\![\ \texttt{var}\ id\ \texttt{:=}\ e_1]\!]_{\text{Ch}} = [\![id\ \texttt{:=}\ E\,(e_1)]\!]_{\text{SIL}} \tag{32}$$

$$S\,[\![\ \texttt{const}\ id\ \texttt{:=}\ e_1]\!]_{\text{Ch}} = [\![id\ \texttt{:=}\ E\,(e_1)]\!]_{\text{SIL}} \tag{33}$$

$$S\,[\![\ \texttt{ghost var}\ id\ \texttt{:=}\ e_1]\!]_{\text{Ch}} = [\![id\ \texttt{:=}\ E\,(e_1)]\!]_{\text{SIL}} \quad \text{Keep track of } \texttt{ghost}\text{?} \tag{34}$$

$$S\,[\![\ \texttt{ghost const}\ id\ \texttt{:=}\ e_1]\!]_{\text{Ch}} = [\![id\ \texttt{:=}\ E\,(e_1)]\!]_{\text{SIL}} \tag{35}$$

$$\begin{aligned} S\,[\![\ \texttt{call}\ v \ldots\ \texttt{:=}\ r\texttt{.}m_1(e \ldots)]\!]_{\text{Ch}} = [\![\ & \texttt{assert}\ E\,(r) \neq \texttt{null;} \\ & \texttt{call}\ v \ldots\ \texttt{:=}\ E\,(r)\,\texttt{.}m_1\,(E\,(e \ldots))]\!]_{\text{SIL}} \end{aligned} \tag{36}$$

If SIL doesn't support multiple return values, the method will have to return a tuple containing the actual return values.

$$\begin{aligned} S\,[\![\ \texttt{call}\ v_1, v_2, \ldots, v_n\ \texttt{:=}\ r\texttt{.}m_1(e \ldots)]\!]_{\text{Ch}} = [\![\ & \texttt{assert}\ E\,(r) \neq \texttt{null;} && (37) \\ & \texttt{call}\ \tau_1\ \texttt{:=}\ E\,(r)\,\texttt{.}m_1(E\,(e \ldots))\ \texttt{;} && (38) \\ & S\,(v_1\ \texttt{:=}\ \tau_1\texttt{.}elem_1)\ \texttt{;} \\ & S\,(v_1\ \texttt{:=}\ \tau_1\texttt{.}elem_1)\ \texttt{;} \\ & \quad\vdots \\ & S\,(v_n\ \texttt{:=}\ \tau_1\texttt{.}elem_n)\,]\!]_{\text{SIL}} \end{aligned}$$

$$\begin{aligned} S\,[\![\texttt{if}\,(e_1)\,s_1\ \texttt{else}\ s_2]\!]_{\text{Ch}} = [\![& \texttt{if}\,E\,(e_1)\ \texttt{then goto}\ \eta_1\ \texttt{else}\ \eta_2\ \texttt{;} \\ & \eta_1\texttt{:}\,S\,(s_1)\ \texttt{;} \\ & \eta_2\texttt{:}\,S\,(s_2)\,]\!]_{\text{SIL}} \end{aligned} \tag{39}$$

`lockchange` for `while` loops ignored for now.

$$\begin{aligned} S\,[\![\texttt{while}\,(c)\ \texttt{invariant}\ i \ldots\ \texttt{;}\ s]\!]_{\text{Ch}} = [\![\ & \quad \texttt{exhale}\ E\,(i \ldots)\ \texttt{;} \\ & \eta_1\texttt{:}\,\texttt{if}\ E\,(c)\ \texttt{then goto}\ \eta_2\ \texttt{else}\ \eta_3\ \texttt{;} \\ & \eta_2\texttt{:}\,\texttt{inhale}\,E\,(i \ldots) \\ & \quad S\,(s) \\ & \quad \texttt{exhale}\ E\,(i \ldots) \\ & \quad \texttt{goto}\ \eta_1\ \texttt{;} \\ & \eta_3\texttt{:}\,\texttt{inhale}\,E\,(i \ldots)]\!]_{\text{SIL}} \end{aligned} \tag{40}$$

2.1 Functions

Make sure that precondition is `assert` ed before every usage. (How does this work when functions are used in predicates, pre/postconditions/invariants? Add function precondition as another clause? "inlining" it more or less? Can't the verifier do this?)

2.2 Predicates

Are `assert` $E(e_1) \neq \texttt{null}$ necessary for folding/unfolding?

$\sigma(o.p)$ looks up the (permission) expression associated with a predicate p on object to$o$.

$$S\,[\![\texttt{fold}\ e_1.p_1]\!]_{\mathrm{Ch}} = [\![\ \texttt{exhale}\ \sigma\left(E(e_1)\texttt{.}p_1\right)]\!]_{\mathrm{SIL}} \quad \text{missing something here} \tag{41}$$

$$S\,[\![\texttt{unfold}\ e_1.p_1]\!]_{\mathrm{Ch}} = [\![\ \texttt{inhale}\ \sigma\left(E(e_1)\texttt{.}p_1\right)]\!]_{\mathrm{SIL}} \tag{42}$$