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

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.

$$S\,[\![\texttt{assert}\ e_1]\!]_{\mathrm{Ch}} = [\![\texttt{assert}\ E\,(e_1)]\!]_{\mathrm{SIL}} \tag{28}$$
$$S\,[\![\texttt{assume}\ e_1]\!]_{\mathrm{Ch}} = [\![\texttt{assume}\ E\,(e_1)]\!]_{\mathrm{SIL}} \tag{29}$$
$$S\,[\![\{s\ \ldots\}]\!]_{\mathrm{Ch}} = [\![S\,(s\ \ldots)]\!]_{\mathrm{SIL}} \tag{30}$$

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

$$S\,[\![\ \texttt{spec}\ id\,[e\ \ldots]]\!]_{\mathrm{Ch}} = \text{What does } \texttt{spec} \text{ do?} \tag{31}$$
$$S\,[\![\ \texttt{var}\ id\ \texttt{:=}\ e_1]\!]_{\mathrm{Ch}} = [\![id\ \texttt{:=}\ E\,(e_1)]\!]_{\mathrm{SIL}} \tag{32}$$
$$S\,[\![\ \texttt{const}\ id\ \texttt{:=}\ e_1]\!]_{\mathrm{Ch}} = [\![id\ \texttt{:=}\ E\,(e_1)]\!]_{\mathrm{SIL}} \tag{33}$$
$$S\,[\![\ \texttt{ghost var}\ id\ \texttt{:=}\ e_1]\!]_{\mathrm{Ch}} = [\![id\ \texttt{:=}\ E\,(e_1)]\!]_{\mathrm{SIL}} \quad \text{Keep track of } \texttt{ghost}\text{?} \tag{34}$$
$$S\,[\![\ \texttt{ghost const}\ id\ \texttt{:=}\ e_1]\!]_{\mathrm{Ch}} = [\![id\ \texttt{:=}\ E\,(e_1)]\!]_{\mathrm{SIL}} \tag{35}$$
$$\begin{aligned} S\,[\![\ \texttt{call}\ v\ \ldots\ \texttt{:=}\ r\,\texttt{.}\,m_1(e\ \ldots)]\!]_{\mathrm{Ch}} = [\![\ &\texttt{assert}\ E\,(r) \neq \texttt{null;}\\ &\texttt{call}\ v\ \ldots\ \texttt{:=}\ E\,(r)\ \texttt{.}\,m_1\,(E\,(e\ \ldots))]\!]_{\mathrm{SIL}} \end{aligned} \tag{36}$$

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

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

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

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

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

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

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

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

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