| | |
|---|---|
| *ident* | Core identifier |
| *tag* | struct/union tag |
| $n$, $i$ | |
| <*impl-const*> | |
| $x$, $y$, *ident* | |
| *intval* | integer value |
| $floatval$ | floating value |
| *memval* | |
| *member* | C struct/union member name |
| $\tau$ | |
| *bty* | |
| *annots* | |
| *Mem_mem_iv_constraint* | |
| *ub-name* | |
| *string* | |

| | |
|---|---|
| $n$ | |
| *bool* | |
| *Loc_t* | |

| | |
|---|---|
| *memory-order* | |
| *linux-memory-order* | |
| *thread-id* | |

| | | | |
|---|---|---|---|
| $oTy$ | ::= | | types for C objects |
| | \| | `integer` | |
| | \| | `floating` | |
| | \| | `pointer` | |
| | \| | `array` $(oTy)$ | |
| | \| | `struct` $tag$ | |
| | \| | `union` $tag$ | |
| $bTy$ | ::= | | Core base types |
| | \| | `unit` | unit |
| | \| | $bool$ | boolean |
| | \| | `ctype` | Core type of C type exprs |
| | \| | $[bTy]$ | list |
| | \| | $(\overline{bTy_i}^{\,i})$ | tuple |
| | \| | $oTy$ | C object value |
| | \| | `loaded` $oTy$ | $oTy$ or unspecified |
| | \| | `storable` | top type for integer/float/pointer/structs (maybe union?). This is only |
| $coreTy$ | ::= | | Core types |
| | \| | $bTy$ | pure base type |
| | \| | `eff` $bTy$ | effectful base type |
| $binop$ | ::= | | binary operators |
| | \| | `+` | |
| | \| | `-` | |
| | \| | `*` | |
| | \| | `/` | |
| | \| | `rem_t` | |
| | \| | `rem_f` | |
| | \| | `^` | |
| | \| | `=` | |
| | \| | `>` | |
| | \| | `<` | |
| | \| | `>=` | |
| | \| | `<=` | |
| | \| | `/\` | |
| | \| | `\/` | |
| $polarity$ | ::= | | memory action polarities |
| | \| | `Pos` | sequenced by `let weak` and `let strong` |
| | \| | `Neg` | only sequenced by `let strong` |
| $name$ | ::= | | |
| | \| | $ident$ | Core identifier |
| | \| | <$impl\text{-}const$> | implementation-defined constant |
| $ptrval$ | ::= | | |

| | | | |
|---|---|---|---|
| | \| | $\texttt{nullptr}\,(\tau)$ | |
| *object_value* | ::= | | C object values |
| | \| | *intval* | integer value |
| | \| | *floatval* | floating-point |
| | \| | *ptrval* | pointer value |
| | \| | $\texttt{array}\,(\overline{loaded\_value_i}^{\,i})$ | C array value |
| | \| | $(\texttt{struct}\ tag)\{\,\overline{.member_i : \tau_i = memval_i}^{\,i}\,\}$ | C struct value |
| | \| | $(\texttt{union}\ tag)\{.member = memval\}$ | C union value |
| *loaded_value* | ::= | | potentially unsp |
| | \| | $\texttt{Specified}\,(object\_value)$ | non-unspecifie |
| | \| | $\texttt{Unspecified}\,(\tau)$ | unspecified loa |
| *value* | ::= | | Core values |
| | \| | *object_value* | C object value |
| | \| | *loaded_value* | loaded C obje |
| | \| | Unit | |
| | \| | True | |
| | \| | False | |
| | \| | $'\tau'$ | C type as valu |
| | \| | $[value_1, .., value_i]$ | |
| | \| | $(value_1, .., value_i)$ | tuple |
| *ctor* | ::= | | data constructor |
| | \| | Nil *bty* | empty list |
| | \| | Cons | list cons |
| | \| | Tuple | tuple |
| | \| | Array | C array |
| | \| | Ivmax | max integer va |
| | \| | Ivmin | min integer va |
| | \| | Ivsizeof | sizeof value |
| | \| | Ivalignof | alignof value |
| | \| | IvCOMPL | bitwise compl |
| | \| | IvAND | bitwise AND |
| | \| | IvOR | bitwise OR |
| | \| | IvXOR | bitwise XOR |
| | \| | Specified | non-unspecifie |
| | \| | Unspecified | unspecified loa |
| | \| | Fvfromint | cast integer to |
| | \| | Ivfromfloat | cast floating t |
| *maybesym_base_type* | ::= | | |
| | \| | $\_ : bTy$ | binders = {} |
| | \| | $ident : bTy$ | binders = *ident* |
| *mu_pattern_aux* | ::= | | |

| | | |
|---|---|---|
| | \| | $maybesym\_base\_type$ |
| | \| | $ctor(\overline{mu\_pattern_i}^{\,i})$ |
| $mu\_pattern$ | ::= | |
| | \| | $annots\ mu\_pattern\_aux$ |
| $mu\_pexpr\_aux$ | ::= | Core p |
| | \| | $ident$ |
| | \| | <*impl-const*> imp |
| | \| | $value$ |
| | \| | $\texttt{constrained}(\overline{Mem\_mem\_iv\_constraint_i, ident_i}^{\,i})$ cons |
| | \| | $\texttt{undef}\ Loc\_t(ub\text{-}name)$ und |
| | \| | $\texttt{error}(string, ident)$ imp |
| | \| | $ctor(\overline{ident_i}^{\,i})$ data |
| | \| | $\texttt{array\_shift}(ident_1, \tau, ident_2)$ poi |
| | \| | $\texttt{member\_shift}(ident_1, ident_2, member)$ poi |
| | \| | $\texttt{not}(ident)$ boo |
| | \| | $ident_1\ binop\ ident_2$ |
| | \| | $(\texttt{struct}\ ident)\{\overline{.member_i = ident_i}^{\,i}\}$ C st |
| | \| | $(\texttt{union}\ ident_1)\{.member = ident_2\}$ C u |
| | \| | $\texttt{memberof}(ident_1, member, ident_2)$ C st |
| | \| | $name(ident_1, .., ident_n)$ pur |
| | \| | $\texttt{assert\_undef}(ident, ub\text{-}name)$ |
| | \| | $\texttt{bool\_to\_integer}(ident)$ |
| | \| | $\texttt{conv\_int}(\tau, ident)$ |
| | \| | $\texttt{wrapI}(\tau, ident)$ |
| $e$ | ::= | |
| | \| | $annots\ bty\ mu\_pexpr\_aux$ |
| $mu\_tpexpr$ | ::= | Core t |
| | \| | $\texttt{case}\ ident\ \texttt{of}\ \overline{\vert mu\_pattern_i => mu\_tpexpr_i}^{\,i}\ \texttt{end}$ patt |
| | \| | $\texttt{let}\ mu\_pattern = mu\_tpexpr_1\ \in\ mu\_tpexpr_2$ pur |
| | \| | $\texttt{if}\ ident\ \texttt{then}\ mu\_tpexpr_1\ \texttt{else}\ mu\_tpexpr_2$ pur |
| | \| | $\texttt{done}\ ident$ pur |
| $mu\_action\_aux$ | ::= | memo |
| | \| | $\texttt{create}(e_1, e_2)$ |
| | \| | $\texttt{create\_readonly}(e_1, e_2, e_3)$ |
| | \| | $\texttt{alloc}(e_1, e_2)$ |
| | \| | $\texttt{kill}(bool, e)$ the |
| | \| | $\texttt{store}(bool, e_1, e_2, e_3, memory\text{-}order)$ the |
| | \| | $\texttt{load}(e_1, e_2, memory\text{-}order)$ |
| | \| | $\texttt{rmw}(e_1, e_2, e_3, e_4, memory\text{-}order_1, memory\text{-}order_2)$ |
| | \| | $\texttt{fence}(memory\text{-}order)$ |
| | \| | $\texttt{compare\_exchange\_strong}(e_1, e_2, e_3, e_4, memory\text{-}order_1, memory\text{-}order_2)$ |
| | \| | $\texttt{compare\_exchange\_weak}(e_1, e_2, e_3, e_4, memory\text{-}order_1, memory\text{-}order_2)$ |

| | | | |
|---|---|---|---|
| | $\mid$ | $\texttt{linux\_fence}\,(\textit{linux-memory-order})$ | |
| | $\mid$ | $\texttt{linux\_load}\,(e_1, e_2, \textit{linux-memory-order})$ | |
| | $\mid$ | $\texttt{linux\_store}\,(e_1, e_2, e_3, \textit{linux-memory-order})$ | |
| | $\mid$ | $\texttt{linux\_rmw}\,(e_1, e_2, e_3, \textit{linux-memory-order})$ | |
| *mu_action* | ::= | | |
| | $\mid$ | *Loc_t mu_action_aux* | |
| *mu_paction* | ::= | | memory actions with po |
| | $\mid$ | *polarity mu_action* | |
| | $\mid$ | *mu_action* | M positive, sequenced b |
| | $\mid$ | $\neg\,(\textit{mu\_action})$ | M negative, only sequen |
| *memop* | ::= | | operations involving the |
| | $\mid$ | *pointer-equality-operator* | pointer equality comp |
| | $\mid$ | *pointer-relational-operator* | pointer relational con |
| | $\mid$ | `ptrdiff` | pointer subtraction |
| | $\mid$ | `intFromPtr` | cast of pointer value |
| | $\mid$ | `ptrFromInt` | cast of integer value t |
| | $\mid$ | `ptrValidForDeref` | dereferencing validity |
| | $\mid$ | `ptrWellAligned` | |
| | $\mid$ | `ptrArrayShift` | |
| | $\mid$ | `memcpy` | |
| | $\mid$ | `memcmp` | |
| | $\mid$ | `realloc` | TODO: not sure abou |
| | $\mid$ | `va_start` | |
| | $\mid$ | `va_copy` | |
| | $\mid$ | `va_arg` | |
| | $\mid$ | `va_end` | |
| *tyvarsym_base_type_pair* | ::= | | |
| | $\mid$ | $\textit{ident} : \textit{bTy}$ | |
| *base_type_pexpr_pair* | ::= | | |
| | $\mid$ | $\textit{bTy} := e$ | |
| $E$ | ::= | | (effectful) expression |
| | $\mid$ | $\texttt{pure}\,(e)$ | |
| | $\mid$ | $\texttt{memop}\,(\textit{memop}, e_1, \ldots, e_n)$ | pointer op involving |
| | $\mid$ | *mu_paction* | memory action |
| | $\mid$ | $\texttt{case}\,e\,\texttt{with}\,\overline{\mid \textit{mu\_pattern}_i \Rightarrow E_i}^{\,i}\,\texttt{end}$ | pattern matching |
| | $\mid$ | $\texttt{let}\,\textit{mu\_pattern} = e\,\texttt{in}\,E$ | |
| | $\mid$ | $\texttt{if}\,e\,\texttt{then}\,E_1\,\texttt{else}\,E_2$ | |
| | $\mid$ | `skip` | |
| | $\mid$ | $\texttt{ccall}\,(e_1, e_2, \overline{e_i}^{\,i})$ | C function call |
| | $\mid$ | $\texttt{pcall}\,(\textit{name}, \overline{e_i}^{\,i})$ | Core procedure call |
| | $\mid$ | $\texttt{unseq}\,(E_1, \ldots, E_n)$ | unsequenced expressi |

| | | |
|---|---|---|
| | $\mid$ | $\texttt{let weak}\ mu\_pattern = E_1\ \in\ E_2$ | weak sequenci |
| | $\mid$ | $\texttt{let strong}\ mu\_pattern = E_1\ \in\ E_2$ | strong sequenc |
| | $\mid$ | $\texttt{let atomic}\ tyvarsym\_base\_type\_pair = mu\_action_1\ \in\ mu\_paction_2$ | atomic sequen |
| | $\mid$ | $\texttt{indet}\,[n](E)$ | indeterminate |
| | $\mid$ | $\texttt{bound}\,[n](E)$ | ...and bounda |
| | $\mid$ | $\texttt{nd}\,(E_1, .., E_n)$ | nondeterminis |
| | $\mid$ | $\texttt{save}\ tyvarsym\_base\_type\_pair(\ \overline{ident_i : base\_type\_pexpr\_pair_i}^{\,i}\ )\ \in\ E$ | save label |
| | $\mid$ | $\texttt{run}\ ident(\ \overline{e_i}^{\,i}\ )$ | run from label |
| | $\mid$ | $\texttt{par}\,(E_1, .., E_n)$ | cppmem-like t |
| | $\mid$ | $\texttt{wait}\,(thread\text{-}id)$ | wait for thread |

| | | |
|---|---|---|
| $E$ | ::= | |
| | $\mid$ | $annots\ E$ |

| | | |
|---|---|---|
| $terminals$ | ::= | |
| | $\mid$ | $\lambda$ |
| | $\mid$ | $\longrightarrow$ |
| | $\mid$ | $\rightarrow$ |
| | $\mid$ | $\vdash$ |
| | $\mid$ | $\in$ |
| | $\mid$ | $\Pi$ |
| | $\mid$ | $\forall$ |
| | $\mid$ | $\multimap$ |
| | $\mid$ | $\supset$ |
| | $\mid$ | $\Sigma$ |
| | $\mid$ | $\exists$ |
| | $\mid$ | $\star$ |
| | $\mid$ | $\wedge$ |
| | $\mid$ | $\neg$ |
| | $\mid$ | $=$ |

| | | |
|---|---|---|
| $bt$ | ::= | OCaml type vari |
| | $\mid$ | |

| | | |
|---|---|---|
| $ocaml\_bool$ | ::= | |
| | $\mid$ | $\texttt{true}$ |
| | $\mid$ | $\texttt{false}$ |

| | | |
|---|---|---|
| $ocaml\_int$ | ::= | |
| | $\mid$ | $k^{\,\mathrm{th}}$ |

| | | |
|---|---|---|
| $lit$ | ::= | |
| | $\mid$ | $ident$ |
| | $\mid$ | $\texttt{Unit}$ |
| | $\mid$ | $ocaml\_bool$ |

| | | |
|---|---|---|
| $bool\_op$ | ::= | |

| | | | |
|---|---|---|---|
| | $\neg\, index\_term$ | | |
| | $index\_term_1 = index\_term_2$ | | |
| | $\wedge(index\_term_1, .., index\_term_n)$ | | |
| *list_op* ::= | | | |
| | $[index\_term_1, .., index\_term_n]$ | | |
| | $ocaml\_int\ index\_term$ | | |
| *index_term_aux* ::= | | | |
| | $bool\_op$ | | |
| | $list\_op$ | | |
| *index_term* ::= | | | |
| | $lit$ | | |
| | $index\_term\_aux\ bt$ | | |
| | $(index\_term)$ | S | parentheses |
| | $index\_term[index\_term_1/ident_1, .., index\_term_n/ident_n]$ | M | |
| *arg* ::= | | | argument types |
| | $\Pi\, ident : bTy.arg$ | | |
| | $\forall\, ident : \texttt{logSort}.arg$ | | |
| | $\texttt{resource} \multimap arg$ | | |
| | $index\_term \supset arg$ | | |
| | $\texttt{I}$ | | |
| *ret* ::= | | | return types |
| | $\Sigma\, ident : bTy.ret$ | | |
| | $\exists\, ident : \texttt{logSort}.ret$ | | |
| | $\texttt{resource} \star ret$ | | |
| | $index\_term \wedge ret$ | | |
| | $\texttt{I}$ | | |
| $\Gamma$ ::= | | | computational var en |
| | $\texttt{empty}$ | | |
| | $\Gamma, x : bTy$ | | |
| $\Lambda$ ::= | | | logical var env |
| | $\texttt{empty}$ | | |
| | $\Lambda, x$ | | |
| $\Xi$ ::= | | | constraints env |
| | $\texttt{empty}$ | | |
| | $\Xi, \texttt{phi}$ | | |
| *formula* ::= | | | |
| | $judgement$ | | |
| | $\texttt{not}\,(formula)$ | | |

| | $ident : bTy \in \Gamma$
| | $formula_1 \quad .. \quad formula_n$

*Jtype* ::=
| | $\Gamma; \Lambda; \Xi \vdash mu\_pexpr\_aux : ret$

*judgement* ::=
| | *Jtype*

*user_syntax* ::=
| | *ident*
| | *tag*
| | *n*
| | <*impl-const*>
| | *x*
| | *intval*
| | *floatval*
| | *memval*
| | *member*
| | $\tau$
| | *bty*
| | *annots*
| | *Mem_mem_iv_constraint*
| | *ub-name*
| | *string*
| |
| | *n*
| | *bool*
| | *Loc_t*
| |
| | *memory-order*
| | *linux-memory-order*
| | *thread-id*
| | *oTy*
| | *bTy*
| | *coreTy*
| | *binop*
| | *polarity*
| | *name*
| | *ptrval*
| | *object_value*
| | *loaded_value*
| | *value*
| | *ctor*
| | *maybesym_base_type*
| | *mu_pattern_aux*
| | *mu_pattern*

| *mu_pexpr_aux*
| *e*
| *mu_tpexpr*
| *mu_action_aux*
| *mu_action*
| *mu_paction*
| *memop*
| *tyvarsym_base_type_pair*
| *base_type_pexpr_pair*
| *E*
| *E*
| *terminals*
| *bt*
| *ocaml_bool*
| *ocaml_int*
| *lit*
| *bool_op*
| *list_op*
| *index_term_aux*
| *index_term*
| *arg*
| *ret*
| $\Gamma$
| $\Lambda$
| $\Xi$
| *formula*

$\boxed{\Gamma; \Lambda; \Xi \vdash \mathit{mu\_pexpr\_aux} : \mathit{ret}}$

$$\frac{x : \mathit{bTy} \in \Gamma}{\Gamma; \Lambda; \Xi \vdash x : \Sigma\, y : \mathit{bTy}.\mathtt{I}} \quad \textsc{GtT\_var}$$

$$\frac{x : \mathit{bool} \in \Gamma}{\Gamma; \Lambda; \Xi \vdash \mathtt{not}\,(x) : \Sigma\, y : \mathit{bool}.y = (\neg\, x\,) \wedge \mathtt{I}} \quad \textsc{GtT\_not}$$

$$\frac{}{\Gamma; \Lambda; \Xi \vdash \mathtt{Unit} : \Sigma\, y : \mathtt{unit}.y = \mathtt{Unit} \wedge \mathtt{I}} \quad \textsc{GtT\_value\_unit}$$

$$\frac{}{\Gamma; \Lambda; \Xi \vdash \mathtt{True} : \Sigma\, y : \mathit{bool}.y = \mathtt{true} \wedge \mathtt{I}} \quad \textsc{GtT\_value\_true}$$

$$\frac{}{\Gamma; \Lambda; \Xi \vdash \mathtt{False} : \Sigma\, y : \mathit{bool}.y = \mathtt{false} \wedge \mathtt{I}} \quad \textsc{GtT\_value\_false}$$

$$\frac{\Gamma; \Lambda; \Xi \vdash \mathit{value}_1 : \Sigma\, y_1 : \mathit{bTy}.\mathit{ret}_1 \quad .. \quad \Gamma; \Lambda; \Xi \vdash \mathit{value}_n : \Sigma\, y_n : \mathit{bTy}.\mathit{ret}_n}{\Gamma; \Lambda; \Xi \vdash [\mathit{value}_1, .., \mathit{value}_i] : \Sigma\, y : [\mathit{bTy}]. \wedge (y_1, \, .. \, , y_n)\,[\,] \wedge \mathtt{I}} \quad \textsc{GtT\_value\_list}$$

```
Definition rules:         6 good    0 bad
Definition rule clauses: 9 good    0 bad
```