MODULE *Harmony*

EXTENDS *Integers*, *Sequences*, *FiniteSets*

VARIABLE *CTXBAG*, *SHARED*, *FAILEDASSERT*

some helper functions

$Neg \triangleq [F \mapsto \text{“T”},\ T \mapsto \text{“F”}]$

add *var* with *val* to map

$NMap(var,\ val,\ map) \triangleq [x \in ((\text{DOMAIN } map) \cup \{var\}) \setminus \{\text{“FALSE”}\} \mapsto \text{IF } x = var \text{ THEN } val \text{ ELSE } map[x]]$

remove *var* from map, until empty map, *i.e.*, FALSE ↦ FALSE

$NMap2(var,\ map) \triangleq [x \in ((\text{DOMAIN } map) \setminus \{var\}) \cup \{\text{“FALSE”}\} \mapsto \text{IF } x \in \text{DOMAIN } map \text{ THEN } map[x] \text{ ELSE}$

remove *var* from map

$NMapReturn(var,\ map) \triangleq [x \in ((\text{DOMAIN } map) \setminus \{var\}) \mapsto map[x]]$

RECURSIVE $NTail(\_,\ \_)$

RECURSIVE $NHead(\_,\ \_)$

RECURSIVE $AddMult(\_,\ \_,\ \_)$

$AddMult(var\_tup,\ val\_tup,\ map) \triangleq \text{IF } Len(var\_tup) = 1 \text{ THEN } [x \in ((\text{DOMAIN } map) \cup \{Head(var\_tup)\}) \setminus \{$
$\text{ELSE } [x \in ((\text{DOMAIN } AddMult(Tail(var\_tup),\ Tail(val\_tup),\ map)) \cup \{$

the last n elements of the list

$NTail(n,\ tup) \triangleq \text{IF } n = 1 \text{ THEN } Tail(tup) \text{ ELSE } NTail(n-1,\ Tail(tup))$

the first n elements of a *tup*

$NHead(n,\ tup) \triangleq \text{IF } n = 1 \text{ THEN } \langle Head(tup)\rangle \text{ ELSE } NHead(n-1,\ Tail(tup)) \circ \langle Head(tup)\rangle$

nth element of a *tup*

$SpawnHead(ctx) \triangleq NHead(3,\ CTXBAG[ctx].stack)$

$SpawnTail(ctx) \triangleq NTail(3,\ CTXBAG[ctx].stack)$

Number of contexts with specified *PC*

$countLabel(This\_PC) \triangleq Cardinality(\{x \in \text{DOMAIN } CTXBAG : CTXBAG[x].pc = This\_PC\})$

$DefaultStateCheckPartial(ctx,\ PC) \triangleq$
$\quad \land (CTXBAG[ctx].atomic = \text{TRUE} \lor (\forall\, x \in \text{DOMAIN } CTXBAG : CTXBAG[x].atomic = \text{FALSE}))$
$\quad \land CTXBAG[ctx].pc = PC$
$\quad \land CTXBAG[ctx].active = \text{TRUE}$

$DefaultStateCheck(ctx,\ PC) \triangleq$
$\quad \land DefaultStateCheckPartial(ctx,\ PC)$
$\quad \land \text{UNCHANGED } SHARED$
$\quad \land \text{UNCHANGED } FAILEDASSERT$

empty record

$e\_rec \triangleq [\text{FALSE} \mapsto \text{FALSE}]$

a new context

$new\_ctx \triangleq [pc \mapsto 0,\ stack \mapsto \langle\langle\rangle\rangle,\ vars \mapsto e\_rec,\ active \mapsto \text{FALSE},\ spn \mapsto \text{FALSE},\ atomic \mapsto \text{FALSE}]$

initial context is marked as spawned;
return checks if context is either in a “spawn state” or “applied state”

$init\_ctx \triangleq [pc \mapsto 0,\ stack \mapsto \langle\langle\rangle\rangle,\ vars \mapsto e\_rec,\ active \mapsto \text{TRUE},\ spn \mapsto \text{TRUE},\ atomic \mapsto \text{FALSE}]$

Harmony Initial State

$HarmonyInit \triangleq$ global variable

$\land\ SHARED = e\_rec$ start empty

$\land\ CTXBAG = [c0 \mapsto init\_ctx,$
$\quad c1 \mapsto new\_ctx,$
$\quad c2 \mapsto new\_ctx]$

$\land\ FAILEDASSERT = \text{FALSE}$

push *val* onto head of *ctx* stack

$Push(ctx,\ PC,\ val) \triangleq$

$\land\ \ DefaultStateCheck(ctx,\ PC)$

$\land\ \ CTXBAG' = [CTXBAG\ \text{EXCEPT}$
$\quad ![ctx].pc = PC + 1,$
$\quad ![ctx].stack = \langle val\rangle \circ CTXBAG[ctx].stack]$

thread store

$StoreVar(ctx,\ PC,\ var) \triangleq$

$\land\ DefaultStateCheck(ctx,\ PC)$

$\land\ CTXBAG' = [CTXBAG\ \text{EXCEPT}$
$\quad ![ctx].pc = PC + 1,$
$\quad ![ctx].stack = Tail(CTXBAG[ctx].stack),$
$\quad ![ctx].vars = NMap(var,$
$\qquad Head(CTXBAG[ctx].stack),$
$\qquad CTXBAG[ctx].vars)$
$\quad]$

shared store

$Store(ctx,\ PC,\ var) \triangleq$

$\land\ DefaultStateCheckPartial(ctx,\ PC)$

$\land\ CTXBAG' = [CTXBAG\ \text{EXCEPT}$
$\quad ![ctx].pc = PC + 1,$
$\quad ![ctx].stack = Tail(CTXBAG[ctx].stack)]$

$\land\ SHARED' = \text{IF}\ var = \text{“”}$
$\quad \text{THEN}\ NMap($
$\qquad Head(Tail(Tail(CTXBAG[ctx].stack))),$
$\qquad NMap(Head(Tail(CTXBAG[ctx].stack)),$
$\qquad\quad Head(CTXBAG[ctx].stack),$
$\qquad\quad SHARED[Head(Tail(Tail(CTXBAG[ctx].stack)))]$
$\qquad),$
$\qquad SHARED$
$\quad)$
$\quad \text{ELSE}\ NMap($
$\qquad var,$
$\qquad Head(CTXBAG[ctx].stack),$

```
                    SHARED
                )
  ∧ UNCHANGED FAILEDASSERT

Jump(ctx, PC, PC_new) ≜
  ∧   DefaultStateCheck(ctx, PC)
  ∧   CTXBAG' = [CTXBAG EXCEPT ![ctx].pc = PC_new]

 push the value of a shared variable onto the context stack
Load(ctx, PC, var_name) ≜
  ∧   DefaultStateCheck(ctx, PC)
   push the value of a shared variable onto the stack
  ∧ CTXBAG' = IF var_name = ""
                THEN [CTXBAG EXCEPT
                        ![ctx].pc = PC + 1,
                        ![ctx].stack = ⟨SHARED[Head(Tail(CTXBAG[ctx].stack))][Head(CTXBAG[ctx].s
                ELSE  [CTXBAG EXCEPT
                        ![ctx].pc = PC + 1,
                        ![ctx].stack = ⟨SHARED[var_name]⟩ ∘ CTXBAG[ctx].stack]

 push the value of a thread variable onto the stack
LoadVar(ctx, PC, var_name) ≜
  ∧ DefaultStateCheck(ctx, PC)
  ∧ CTXBAG' = [CTXBAG EXCEPT
                  ![ctx].pc = PC + 1,
                  ![ctx].stack = ⟨CTXBAG[ctx].vars[var_name]⟩ ∘ CTXBAG[ctx].stack]

Spawn(ctxa, PC) ≜
  ∧ DefaultStateCheck(ctxa, PC)
  ∧ LET SpStk ≜ SpawnHead(ctxa)IN
    LET ctxb ≜ CHOOSE x ∈ DOMAIN CTXBAG : CTXBAG[x].active = FALSEIN
     ∧ CTXBAG' = [CTXBAG EXCEPT
                    ![ctxa].pc = PC + 1,
                    ![ctxa].stack = SpawnTail(ctxa),
                    ![ctxb].pc = Head(SpStk),
                    ![ctxb].stack = Tail(SpStk),
                    ![ctxb].active = TRUE,
                    ![ctxb].spn = TRUE]

   delete thread variable var
DelVar(ctx, PC, var) ≜
  ∧ DefaultStateCheck(ctx, PC)
  ∧ CTXBAG' = [CTXBAG EXCEPT
                  ![ctx].pc = PC + 1,
                  ![ctx].vars = NMap2(var, CTXBAG[ctx].vars)]

 take top of the context's stack and assign it to Frame instruction arguments args
```

TODO want to do store *var* on possibly a tuple, only works for single *var* now

$Frame(ctx,\ PC,\ args)\ \triangleq$
$\ \wedge\ DefaultStateCheck(ctx,\ PC)$
$\ \wedge\ CTXBAG[ctx].spn = \text{TRUE}$
$\ \wedge\ CTXBAG' = [CTXBAG\ \text{EXCEPT}$
$\qquad ![ctx].pc = PC + 1,$
$\qquad ![ctx].stack = Tail(CTXBAG[ctx].stack),$
$\qquad ![ctx].vars = AddMult(args,\ CTXBAG[ctx].stack,\ CTXBAG[ctx].vars)]$

$Return(ctx,\ PC)\ \triangleq$
$\ \wedge\ DefaultStateCheck(ctx,\ PC)$
$\ \wedge\ \text{IF}\ CTXBAG[ctx].spn = \text{TRUE}$
$\quad \text{THEN}$
$\quad \wedge\ CTXBAG' = [CTXBAG\ \text{EXCEPT}\ ![ctx].active = \text{FALSE}]$
$\quad \text{ELSE}$
$\quad \wedge\ CTXBAG' = [CTXBAG\ \text{EXCEPT}$
$\qquad ![ctx].pc = Head(CTXBAG[ctx].stack),$
$\qquad ![ctx].stack = Tail(CTXBAG[ctx].stack)]$

$AssertH(ctx,\ PC)\ \triangleq$
$\ \wedge\ DefaultStateCheckPartial(ctx,\ PC)$
$\ \wedge\ \text{UNCHANGED}\ SHARED$
$\ \wedge\ \text{IF}\ Head(CTXBAG[ctx].stack) = \text{TRUE}$
$\quad \text{THEN}$
$\quad (\ \wedge\ CTXBAG' = [CTXBAG\ \text{EXCEPT}$
$\qquad ![ctx].pc = PC + 1,$
$\qquad ![ctx].stack = Tail(CTXBAG[ctx].stack)]$
$\quad \wedge\ \text{UNCHANGED}\ FAILEDASSERT)$
$\quad \text{ELSE}$
$\quad (\ \wedge\ CTXBAG' = [x \in \text{DOMAIN}\ CTXBAG \mapsto\ [$
$\qquad pc \mapsto CTXBAG[x].pc,$
$\qquad stack \mapsto CTXBAG[x].stack,$
$\qquad vars \mapsto CTXBAG[x].vars,$
$\qquad active \mapsto \text{FALSE},$
$\qquad spn \mapsto CTXBAG[x].spn,$
$\qquad atomic \mapsto CTXBAG[x].atomic$
$\qquad\ ]$
$\qquad]$
$\quad \wedge\ FAILEDASSERT' = \text{TRUE})$

$JumpCond(ctx,\ PC,\ exp,\ PC\_new)\ \triangleq$
$\ \wedge\ DefaultStateCheck(ctx,\ PC)$
$\ \wedge\ \text{IF}\ Head(CTXBAG[ctx].stack) = exp\ \text{THEN}$
$\quad (\ \wedge\ CTXBAG' = [CTXBAG\ \text{EXCEPT}\ ![ctx].pc = PC\_new,\ ![ctx].stack = Tail(CTXBAG[ctx].stack)])$
$\quad \text{ELSE}$
$\quad (\ \wedge\ CTXBAG' = [CTXBAG\ \text{EXCEPT}\ ![ctx].pc = PC + 1,\ ![ctx].stack = Tail(CTXBAG[ctx].stack)])$

$AtomicInc(ctx,\ PC)\ \triangleq$
$\land\ DefaultStateCheck(ctx,\ PC)$
$\land\ CTXBAG' = [CTXBAG\ \text{EXCEPT}$
$!\,[ctx].pc = PC + 1,$
$!\,[ctx].atomic = \text{TRUE}]$

$AtomicDec(ctx,\ PC)\ \triangleq$
$\land\ DefaultStateCheck(ctx,\ PC)$
$\land\ CTXBAG' = [CTXBAG\ \text{EXCEPT}$
$!\,[ctx].pc = PC + 1,$
$!\,[ctx].atomic = \text{FALSE}]$

$NotOp(ctx,\ PC)\ \triangleq$
$\land\ DefaultStateCheck(ctx,\ PC)$
$\land\ CTXBAG' = [CTXBAG\ \text{EXCEPT}$
$!\,[ctx].pc = PC + 1,$
$!\,[ctx].stack = \langle Neg[Head(CTXBAG[ctx].stack)]\rangle \circ Tail(CTXBAG[ctx].stack)]$

$EqOp(ctx,\ PC)\ \triangleq$
$\land\quad DefaultStateCheck(ctx,\ PC)$
$\land\quad CTXBAG' = [CTXBAG\ \text{EXCEPT}$
$!\,[ctx].pc = PC + 1,$
$!\,[ctx].stack = \langle (Head(CTXBAG[ctx].stack) = Head(Tail(CTXBAG[ctx].stack)))\rangle \circ Ta$

$Dummy(ctx,\ PC)\ \triangleq$
$\land\ DefaultStateCheck(ctx,\ PC)$
$\land\ CTXBAG' = [CTXBAG\ \text{EXCEPT}\ !\,[ctx].pc = PC + 1]$

* Modification History
* Last modified *Fri Dec* 10 21:18:32 *EST* 2021 by *katyblumer*
* Last modified *Fri Dec* 10 19:54:36 *EST* 2021 by *noah*
* Last modified *Thu Nov* 18 16:26:44 *EST* 2021 by *arielkellison*
* Created *Tue Nov* 02 18:59:20 *EDT* 2021 by *arielkellison*