MODULE *Stuttering*

This module is explained in Section 5 of the paper "Auxiliary Variables in TLA+". It defines operators used to add a stuttering variable s to a specification *Spec* to form a specification *SpecS*. It is mean to be instantiated with s replaced by the stuttering variable to be added and *vars* replaced by the tuple of all variables in the original specification.

If *Init* is the initial predicate of *Spec* , then the initial predicate of *SpecS* is $Init \land (s = top)$, where top is defined below.

The next-state action of *SpecS* is obtained by replacing each subaction A of a disjunctive representation of the next-state action Next of *Spec* with an action AS written in terms of operators defined below. (Disjunctive representations are described in Section 3.2 of "Auxiliary Variables in TLA+".) Action AS executes A and stuttering steps added either before or after an A step. The basic idea is that s equals top except while stuttering steps are being taken, when it is a record with the following fields:

id: A value that identifies the action A .

ctxt: A value identifying the context under which A is executed. For example, if A appears in a formula $\exists\, i, j \in S : A$, this would equal the value of the pair $\langle i, j \rangle$ for which A is being executed.

val: A value that is decremented by each stuttering step, until it reaches a minimum value.

The arguments of the operators defined in this module have the following meanings.

A
The subaction A of Next.

id
A string identifying action A .

Sigma
A set of values ordered by some "less-than" relation. This is the set of possible values of $s.val$ when executing stuttering steps before or after subaction A .

bot
The minimum element of *Sigma* under its less-than relation.

initVal
The initial value to which $s.val$ is set for executing stuttering steps before or after A .

decr
An operator such that each stuttering step changes $s.val$ to $decr(s.val)$.

context
The context in which A appears. It is the expression that is evaluated to determine the value to which $s.ctxt$ is set.

enabled
A formula equivalent to ENABLED A . You can always take it to be
ENABLED A , but you can usually find an expression that equals
ENABLED A in every reachable state of *Spec* but is easier for *TLC* to compute.

```
EXTENDS Naturals, TLC
top ≜ [top ↦ "top"]

VARIABLES s, vars
```

```
$NoStutter(A) \triangleq (s = top) \land A \land (s' = s)$

$PostStutter(A, actionId, context, bot, initVal, decr(\_)) \triangleq$
  IF $s = top$ THEN $\land A$
                    $\land s' = [id \mapsto actionId, ctxt \mapsto context, val \mapsto initVal]$
               ELSE $\land s.id = actionId$
                    $\land$ UNCHANGED $vars$
                    $\land s' =$ IF $s.val = bot$ THEN $top$
                                                ELSE $[s$ EXCEPT $!.val = decr(s.val)]$

$PreStutter(A, enabled, actionId, context, bot, initVal, decr(\_)) \triangleq$
  IF $s = top$
    THEN $\land enabled$
         $\land$ UNCHANGED $vars$
         $\land s' = [id \mapsto actionId, ctxt \mapsto context, val \mapsto initVal]$
    ELSE $\land s.id = actionId$
         $\land$ IF $s.val = bot$ THEN $\land s.ctxt = context$
                                     $\land A$
                                     $\land s' = top$
                                ELSE $\land$ UNCHANGED $vars$
                                     $\land s' = [s$ EXCEPT $!.val = decr(s.val)]$
```

```
$MayPostStutter(A, actionId, context, bot, initVal, decr(\_)) \triangleq$
  IF $s = top$ THEN $\land A$
                    $\land s' =$ IF $initVal = bot$
                                   THEN $s$
                                   ELSE $[id \mapsto actionId, ctxt \mapsto context,$
                                          $val \mapsto initVal]$
               ELSE $\land s.id = actionId$
                    $\land$ UNCHANGED $vars$
                    $\land s' =$ IF $decr(s.val) = bot$
                                   THEN $top$
                                   ELSE $[s$ EXCEPT $!.val = decr(s.val)]$

$MayPreStutter(A, enabled, actionId, context, bot, initVal, decr(\_)) \triangleq$
  IF $s = top$
    THEN $\land enabled$
         $\land$ UNCHANGED $vars$
         $\land$ IF $initVal = bot$
              THEN $A \land (s' = s)$
              ELSE $s' = [id \mapsto actionId, ctxt \mapsto context, val \mapsto initVal]$
    ELSE $\land s.id = actionId$
         $\land$ IF $s.val = bot$ THEN $\land s.ctxt = context$
                                     $\land A$
                                     $\land s' = top$
                                ELSE $\land$ UNCHANGED $vars$
```

```
                                        ∧ s′ = [s EXCEPT !.val = decr(s.val)]

StutterConstantCondition(Sigma, bot, decr(_)) ≜
  LET InverseDecr(S) ≜ {sig ∈ Sigma \ S : decr(sig) ∈ S}
      R[n ∈ Nat] ≜ IF n = 0 THEN {bot}
                                ELSE LET T ≜ R[n − 1]
                                     IN  T ∪ InverseDecr(T)

  IN  Sigma = UNION {R[n] : n ∈ Nat}
  LET InverseDecr(S) ≜ {sig ∈ Sigma \ S : decr(sig) ∈ S}
      ReachBot[S ∈ SUBSET Sigma] ≜
        IF InverseDecr(S) = {} THEN S
                           ELSE ReachBot[S ∪ InverseDecr(S)]
  IN  ReachBot[{bot}] = Sigma
```

* Modification History
* Last modified *Thu Oct* 20 06:27:41 *PDT* 2016 by *lamport*
* Created *Tue Dec* 08 11:51:34 *PST* 2015 by *lamport*