Feature: Generic Measure

Books: GeNoC-scheduling, GeNoC
Version: 1.0

In GeNoC 1.0, the termination of the main function $GeNoC_t$ is proven by a *sum of attempts*-approach (for more information, see [1]). This approach assumes that with each node a finite number of attempts is associated, from which the termination of GeNoC follows directly. This assumption does not hold in general and therefore a generic measure for the termination of GeNoC is added. This measure is to be instantiated by the scheduler in such a way that it decreases on each call of the scheduling function. It can for example be the list of pending messages, the sum of attempts or any other user-defined measure.

In the specification of function *scheduling* the `att`-parameter is replaced by a parameter called `measure`. The specification is extended with two functions, which respectively determine whether the given measure is the base measure (the measure for which GeNoC terminates) and convert the given measure to an ordinal.

$((is\text{-}base\text{-}measure\ *) => *)$
$((to\text{-}ordinal\text{-}measure\ *) => *))$

In order to prove termination of $GeNoC_t$, three proof obligations are required:

```
(defthm booleanp-is-base-measure
  (booleanp (is-base-measure x))
  :rule-classes :type-prescription)

(defthm booleanp-to-ordinal-measure
  (o-p (to-ordinal-measure x))
  :rule-classes :type-prescription)

(defthm measure-decreases
  (implies (not (is-base-measure measure))
           (mv-let (... new-measure ...)
                   (scheduling ... measure ...)
                   (O< (to-ordinal-measure new-measure)
                       (to-ordinal-measure measure)))))
```

The main function $GeNoC_t$ is altered in such a way that it uses the measure provided by the scheduler. ACL2 needs no assistance in proving termination, accept for specifying the measure.

```
(defun GeNoC_t (... measure ...)
  (declare (xargs :measure (to-ordinal-measure measure)))
  (if (is-base-measure measure)
    ; base case
    (mv-let (... new-measure)
            (scheduling ... measure ...)
            (GeNoC_t ... new-measure ...))))
```

Examples

Two simple instantiations are given that solely satisfy the three proof obligations above. Furthermore, the Spidergon instantiation is altered to comply with the full new specification, still using a sum of attempts approach. The first simple example uses a natural number as measure and *zp*

as base function. The second one uses a list of natural numbers as measure and the base case is when the sum of the elements is zero.

```
(defun function-decreasing-measure-nat (measure)
  (if (zp measure)
    0
    (- measure 1)))
(definstance GenericScheduling nat-as-measure-complies
  :functional-substitution
  ((scheduling function-decreasing-measure-nat)
   (is-base-measure zp)
   (to-ordinal-measure acl2-count)))



(defun is-base-sumoflist (x)
  (equal (sumoflist x) 0))
(defun function-decreasing-measure-sumoflist (measure)
  (if (is-base-sumoflist measure)
    nil
    (if (zp (car measure))
      (function-decreasing-measure-sumoflist (cdr measure))
      (cons (- (car measure) 1) (cdr measure)))))
(definstance GenericScheduling sumoflist-as-measure-complies
  :functional-substitution
  ((scheduling function-decreasing-measure-sumoflist)
   (is-base-measure is-base-sumoflist)
   (to-ordinal-measure sumoflist)))
```

References

[1] Julien Schmaltz. *Formalizing on chip communications in a functional style.* PhD thesis, Université Joseph Fourier, Grenoble, France, january 2006.