```
#|

 Copyright (C) 1994 by Alex Bronstein and Carolyn Talcott.  All Rights
 Reserved.

 You may copy and distribute verbatim copies of this Nqthm-1992 event script as
 you receive it, in any medium, including embedding it verbatim in derivative
 works, provided that you conspicuously and appropriately publish on each copy
 a valid copyright notice "Copyright (C) 1994 by Alex Bronstein and Carolyn
 Talcott.  All Rights Reserved."

 NO WARRANTY

 Alex Bronstein and Carolyn Talcott PROVIDE ABSOLUTELY NO WARRANTY.  THE EVENT
 SCRIPT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR
 IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF
 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO
 THE QUALITY AND PERFORMANCE OF THE SCRIPT IS WITH YOU.  SHOULD THE SCRIPT
 PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR
 CORRECTION.

 IN NO EVENT WILL Alex Bronstein or Carolyn Talcott BE LIABLE TO YOU FOR ANY
 DAMAGES, ANY LOST PROFITS, LOST MONIES, OR OTHER SPECIAL, INCIDENTAL OR
 CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THIS SCRIPT
 (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE
 OR LOSSES SUSTAINED BY THIRD PARTIES), EVEN IF YOU HAVE ADVISED US OF THE
 POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

|#
```

EVENT: Start with the library "mlp" using the compiled version.

```
;(setq theta '(

; theta.bm
;   This is a 2nd order circuit constructor, derived from all the "Accumulator"
; examples, as well as the simple counter example.  Basically, it arises in
; a case where the SPEC is easily expressible: as the iteration of "last-char"
; function which can be expressed only using the input string (x).  So in other
; words, we have a string to char function "last-char" which defines the
; circuit, for example: the sum of all inputs, the length of input, etc...
; and we just build the corresponding string function in a "standard" fashion.
; This standard fashion is the THETA operator.
```

```
; This file just attempts to abstract the construction, and the correctness
; proof that goes with it, to facilitate future instantiations.
;
; Clearly, there is no sugar involved.
;
; Standard hints necessary for a PROVE-LEMMA have been removed in the
; corresponding ADD-AXIOM.

;;; DEFINITION OF CIRCUIT:
```

EVENT: Introduce the function symbol *sysd-theta* of 2 arguments.

```
;sysd-stringp is normally deduced by BM.
```

AXIOM: sysd-stringp
stringp (sysd-theta (*line*, x))

```
;;; SPEC definition:
```

EVENT: Introduce the function symbol *spec-theta-lastchar* of one argument.

```
; this is the standard extension from last-char-fun to MLP-string-fun.
```

DEFINITION:
spec-theta (x)
= **if** empty (x) **then** E
else a (spec-theta (p (x)), spec-theta-lastchar (x)) **endif**

```
;;; PROOF of equivalence with spec:

;;; 2nd order instantiations for circuits:
;; theta-begin
```

AXIOM: a2-empty-theta
empty (sysd-theta (*line*, x)) = empty (x)

AXIOM: a2-e-theta
(sysd-theta (*line*, x) = E) = empty (x)

AXIOM: a2-lp-theta
len (sysd-theta (*line*, x)) = len (x)

AXIOM: a2-lpe-theta
eqlen (sysd-theta (*line*, x), x)

AXIOM: a2-pc-theta
$(\neg\ \text{empty}\,(x)) \rightarrow (\text{p}\,(\text{sysd-theta}\,(\mathit{line},\ x)) = \text{sysd-theta}\,(\mathit{line},\ \text{p}\,(x)))$

```
;; theta-end

;;; Circuit CORRECTNESS:

; Theta-correct-ax is a "predicative correctness statement",i.e. what we would
; do if we didn't have functional equality as a specification method, but
; instead used a purely axiomatic approach.  It matches the intuitive view
; of just looking at the last char.
```

AXIOM: theta-correct-ax
$(\neg\ \text{empty}\,(x)) \rightarrow (\text{l}\,(\text{sysd-theta}\,(\texttt{'ytheta},\ x)) = \text{spec-theta-lastchar}\,(x))$

```
; To go to a functional equality once we have the "last" (ax) statement is
; a trivial induction, if we start out with an P-L split which is unnatural
; for BM, so we force it w/ a USE hint of A-p-l-split
```

THEOREM: a-p-l-split
$(\neg\ \text{empty}\,(x))$
$\rightarrow\ (\text{sysd-theta}\,(\texttt{'ytheta},\ x)$
$\quad = \ \text{a}\,(\text{p}\,(\text{sysd-theta}\,(\texttt{'ytheta},\ x)),\ \text{l}\,(\text{sysd-theta}\,(\texttt{'ytheta},\ x))))$

```
; Interestingly: A-P-L needs to be disabled for theta-correct to go through.
; yet in more specific cases such as macc, or funacc, it is not needed, and
; in fact just makes a very minor time improvement.
```

THEOREM: theta-correct
$\text{sysd-theta}\,(\texttt{'ytheta},\ x) = \text{spec-theta}\,(x)$

```
; eof: theta.bm
;))
```

Index

a, 2, 3
a-p-l-split, 3
a2-e-theta, 2
a2-empty-theta, 2
a2-lp-theta, 2
a2-lpe-theta, 2
a2-pc-theta, 3

e, 2
empty, 2, 3
eqlen, 2

l, 3
len, 2

p, 2, 3

spec-theta, 2, 3
spec-theta-lastchar, 2, 3
stringp, 2
sysd-stringp, 2
sysd-theta, 2, 3

theta-correct, 3
theta-correct-ax, 3