EVENT: Start with the library "interpreter".

```
;;;this files defines a token passing solution to the n processor
;;;mutual exclusion problem.
```

DEFINITION:
status (*state*, *index*) = cdr (assoc (cons ('me, *index*), *state*))

DEFINITION:
wait (*state*, *index*) = (status (*state*, *index*) = 'wait)

DEFINITION:
non-critical (*state*, *index*) = (status (*state*, *index*) = 'non-critical)

DEFINITION:
critical (*state*, *index*)
= ((¬ non-critical (*state*, *index*)) ∧ (¬ wait (*state*, *index*)))

DEFINITION: ticks (*state*, *index*) = fix (status (*state*, *index*))

DEFINITION:
critical-ticks (*state*, *index*, *ticks*)
= (critical (*state*, *index*) ∧ (ticks (*state*, *index*) = fix (*ticks*)))

DEFINITION:
channel (*state*, *index*) = cdr (assoc (cons ('c, *index*), *state*))

DEFINITION: token (*state*, *index*) = listp (channel (*state*, *index*))

DEFINITION:
me (*old*, *new*, *index*, *size*)
= **if** non-critical (*old*, *index*)
**then if** non-critical (*new*, *index*)
**then if** token (*old*, *index*)
**then if** add1-mod (*size*, *index*) = *index*
**then** (length (channel (*new*, *index*))
= length (channel (*old*, *index*)))
∧ changed (*old*, *new*, list (cons ('c, *index*)))
**else** (channel (*new*, *index*)
= cdr (channel (*old*, *index*)))
∧ (length (channel (*new*,
add1-mod (*size*, *index*)))
= (1 + length (channel (*old*,
add1-mod (*size*,

*index*)))))
∧ changed (*old*,
*new*,
list (cons (’c, *index*),
cons (’c,
add1-mod (*size*,
*index*)))) **endif**
**else** changed (*old*, *new*, **nil**) **endif**
**else** wait (*new*, *index*)
∧ changed (*old*, *new*, list (cons (’me, *index*))) **endif**
**elseif** wait (*old*, *index*)
**then if** token (*old*, *index*)
**then** (channel (*new*, *index*) = cdr (channel (*old*, *index*)))
∧ critical (*new*, *index*)
∧ changed (*old*,
*new*,
list (cons (’me, *index*), cons (’c, *index*)))
**else** changed (*old*, *new*, **nil**) **endif**
**else** ((ticks (*new*, *index*) < ticks (*old*, *index*))
∧ critical (*new*, *index*)
∧ changed (*old*, *new*, list (cons (’me, *index*))))
∨ (non-critical (*new*, *index*)
∧ (length (channel (*new*, add1-mod (*size*, *index*)))
= (1 + length (channel (*old*,
add1-mod (*size*,
*index*)))))
∧ changed (*old*,
*new*,
list (cons (’c, add1-mod (*size*, *index*)),
cons (’me, *index*)))) **endif**

DEFINITION:
me-function (*index*, *size*, *state*)
= **if** non-critical (*state*, *index*)
**then if** token (*state*, *index*)
**then** update-assoc (cons (’c, add1-mod (*size*, *index*)),
cons (’token,
channel (update-assoc (cons (’c, *index*),
cdr (channel (*state*,
*index*)),
*state*),
add1-mod (*size*, *index*))),
update-assoc (cons (’c, *index*),
cdr (channel (*state*, *index*)),

*state*))
**else** *state* **endif**
**elseif** wait (*state*, *index*)
**then if** token (*state*, *index*)
**then** update-assoc (cons (`'c`, *index*),
cdr (channel (*state*, *index*)),
update-assoc (cons (`'me`, *index*), `0`, *state*))
**else** *state* **endif**
**elseif** ticks (*state*, *index*) $\simeq$ `0`
**then** update-assoc (cons (`'c`, add1-mod (*size*, *index*)),
cons (`'token`, channel (*state*, add1-mod (*size*, *index*))),
update-assoc (cons (`'me`, *index*),
`'non-critical`,
*state*))
**else** update-assoc (cons (`'me`, *index*), ticks (*state*, *index*) $-$ 1, *state*) **endif**

THEOREM: total-me
((*index* $\in$ **N**) $\wedge$ (*index* $<$ *size*))
$\rightarrow$ me (*old*, me-function (*index*, *size*, *old*), *index*, *size*)

DEFINITION:
program (*index*, *size*)
= **if** *index* $\simeq$ `0` **then nil**
**else** cons (list (`'me`, *index* $-$ 1, *size*), program (*index* $-$ 1, *size*)) **endif**

DEFINITION: me-prg (*size*) = program (*size*, *size*)

THEOREM: member-program
(*statement* $\in$ program (*index*, *size*))
= **if** *index* $\simeq$ `0` **then f**
**else** (car (*statement*) = `'me`)
$\wedge$ (cadr (*statement*) $\in$ **N**)
$\wedge$ (cadr (*statement*) $<$ *index*)
$\wedge$ (caddr (*statement*) = *size*)
$\wedge$ (cdddr (*statement*) = **nil**) **endif**

THEOREM: member-me-prg
(*statement* $\in$ me-prg (*size*))
= **if** *size* $\simeq$ `0` **then f**
**else** (car (*statement*) = `'me`)
$\wedge$ (cadr (*statement*) $\in$ **N**)
$\wedge$ (cadr (*statement*) $<$ *size*)
$\wedge$ (caddr (*statement*) = *size*)
$\wedge$ (cdddr (*statement*) = **nil**) **endif**

EVENT: Disable me-prg.

THEOREM: total-prg
total (me-prg (*size*))

DEFINITION:
weight-of-triple (*state*, *index*, *size*)
= (**if** critical (*state*, *index*) **then** `1`
**else** `0` **endif**
+ length (channel (*state*, *index*))
+ length (channel (*state*, add1-mod (*size*, *index*))))

THEOREM: weight-of-triple-preserved
((*index* ∈ **N**)
∧ (`1` < *size*)
∧ (*index* < *size*)
∧ me (*old*, *new*, *index*, *size*))
→ (weight-of-triple (*new*, *index*, *size*)
= weight-of-triple (*old*, *index*, *size*))

DEFINITION:
weight-but-triple (*state*, *index*, *size*, *n*)
= **if** *n* ≃ `0` **then** `0`
**elseif** (*n* − 1) = *index*
**then** weight-but-triple (*state*, *index*, *size*, *n* − 1)
**else if** critical (*state*, *n* − 1) **then** `1`
**else** `0` **endif**
+ **if** (*n* − 1) = add1-mod (*size*, *index*) **then** `0`
**else** length (channel (*state*, *n* − 1)) **endif**
+ weight-but-triple (*state*, *index*, *size*, *n* − 1) **endif**

THEOREM: weight-but-triple-preserved
((*index* ∈ **N**) ∧ (*index* < *size*) ∧ me (*old*, *new*, *index*, *size*))
→ (weight-but-triple (*new*, *index*, *size*, *n*)
= weight-but-triple (*old*, *index*, *size*, *n*))

DEFINITION:
weight (*state*, *size*)
= **if** *size* ≃ `0` **then** `0`
**else if** critical (*state*, *size* − 1) **then** `1`
**else** `0` **endif**
+ length (channel (*state*, *size* − 1))
+ weight (*state*, *size* − 1) **endif**

THEOREM: weight-is-sum
((*index* ∈ **N**) ∧ (*index* < *size*) ∧ (`1` < *size*))
→ (weight (*state*, *n*)

```
=  (if index < n
      then if critical (state, index) then 1
             else 0 endif
             + length (channel (state, index))
      else 0 endif
      + if add1-mod (size, index) < n
        then length (channel (state, add1-mod (size, index)))
        else 0 endif
      + weight-but-triple (state, index, size, n)))
```

THEOREM: weight-preserved-1
$((index \in \mathbf{N}) \wedge (index < size) \wedge (1 < size) \wedge \text{me}(old, new, index, size)) \rightarrow (\text{weight}(new, size) = \text{weight}(old, size))$

THEOREM: weight-preserved-2
$\text{me}(old, new, 0, 1) \rightarrow (\text{weight}(new, 1) = \text{weight}(old, 1))$

THEOREM: about-size-index
$((index \in \mathbf{N}) \wedge (index < size)) \rightarrow (((index = 0) \wedge (size = 1)) \vee (1 < size))$

THEOREM: weight-preserved
$((index \in \mathbf{N}) \wedge (index < size) \wedge \text{me}(old, new, index, size)) \rightarrow (\text{weight}(new, size) = \text{weight}(old, size))$

EVENT: Disable weight-is-sum.

EVENT: Disable weight-preserved-1.

EVENT: Disable weight-preserved-2.

EVENT: Disable weight-of-triple-preserved.

EVENT: Disable weight-but-triple-preserved.

EVENT: Disable total-me.

DEFINITION:
mutual-exclusionp (*state*, *size*) = (weight (*state*, *size*) = 1)

THEOREM: mutual-exclusionp-unless-sufficient
unless-sufficient (*statement*,
me-prg (*size*),
*old*,
*new*,
`‘(mutual-exclusionp state ’,size)`,
`’(false)`)

EVENT: Disable weight-preserved.

THEOREM: mutual-exclusionp-prg
unless (`‘(mutual-exclusionp state ’,size)`, `’(false)`, me-prg (*size*))

THEOREM: wait-unless-critical-1
((*index* $\in$ **N**)
$\wedge$ (*index* $<$ *size*)
$\wedge$ me (*old*, *new*, *index*, *size*)
$\wedge$ wait (*old*, *index*))
$\rightarrow$ (wait (*new*, *index*) $\vee$ critical (*new*, *index*))

THEOREM: wait-unless-critical-2
((*index* $\in$ **N**)
$\wedge$ (*index* $<$ *size*)
$\wedge$ (*index1* $\in$ **N**)
$\wedge$ (*index1* $<$ *size*)
$\wedge$ (*index* $\neq$ *index1*)
$\wedge$ me (*old*, *new*, *index1*, *size*)
$\wedge$ wait (*old*, *index*))
$\rightarrow$ (wait (*new*, *index*) $\vee$ critical (*new*, *index*))

THEOREM: wait-unless-critical-unless-sufficient
((*index* $\in$ **N**) $\wedge$ (*index* $<$ *size*))
$\rightarrow$ unless-sufficient (*statement*,
me-prg (*size*),
*old*,
*new*,
`‘(wait state ’,index)`,
`‘(critical state ’,index)`)

THEOREM: wait-unless-critical
((*index* $\in$ **N**) $\wedge$ (*index* $<$ *size*) $\wedge$ (*size* $\not\simeq$ `0`))
$\rightarrow$ unless (`‘(wait state ’,index)`,
`‘(critical state ’,index)`,
me-prg (*size*))

THEOREM: celcor-ensures-key
$((index \in \mathbf{N}) \wedge (index < size))$
$\rightarrow$ ensures-key (list (`'me`, *index*, *size*),
me-prg (*size*),
*old*,
*new*,
```
`(and
  (critical state ',index)
  (lessp
   (ticks state ',index)
   ',(add1 ticks))),
`(or
  (and
   (critical state ',index)
   (lessp (ticks state ',index) ',ticks))
  (token state ',(add1-mod size index))))
```

THEOREM: celcor-ensures-rest
$((index \in \mathbf{N}) \wedge (index < size))$
$\rightarrow$ ensures-rest (*statement*,
list (`'me`, *index*, *size*),
me-prg (*size*),
*old*,
*new*,
```
`(and
  (critical state ',index)
  (lessp
   (ticks state ',index)
   ',(add1 ticks))),
```
*q*)

THEOREM: critical-ensures-less-critical-or-right
$((index \in \mathbf{N}) \wedge (index < size))$
$\rightarrow$ ensures (
```
`(and
  (critical state ',index)
  (lessp (ticks state ',index) ',(add1 ticks))),
`(or
  (and
   (critical state ',index)
   (lessp (ticks state ',index) ',ticks))
  (token state ',(add1-mod size index))),
```
me-prg (*size*))

THEOREM: critical-unless-less-critical-or-right
$((index \in \mathbf{N}) \wedge (index < size))$

→ unless (‘(and
(critical state ’,index)
(lessp (ticks state ’,index) ’,(add1 ticks))),
‘(or
(and
(critical state ’,index)
(lessp (ticks state ’,index) ’,ticks))
(token state ’,(add1-mod size index))),
me-prg (*size*))

THEOREM: wlcec-ensures-key
((*index* ∈ **N**) ∧ (*index* < *size*))
→ ensures-key (list (’me, *index*, *size*),
me-prg (*size*),
*old*,
*new*,
‘(and
(wait state ’,index)
(token state ’,index)),
‘(critical state ’,index))

THEOREM: wlcec-ensures-rest
((*index* ∈ **N**) ∧ (*index* < *size*))
→ ensures-rest (*statement*,
list (’me, *index*, *size*),
me-prg (*size*),
*old*,
*new*,
‘(and
(wait state ’,index)
(token state ’,index)),
*q*)

THEOREM: wait-and-left-channel-ensures-critial
((*index* ∈ **N**) ∧ (*index* < *size*))
→ ensures (‘(and
(wait state ’,index)
(token state ’,index)),
‘(critical state ’,index),
me-prg (*size*))

THEOREM: wait-and-left-channel-unless-critial
((*index* ∈ **N**) ∧ (*index* < *size*))
→ unless (‘(and
(wait state ’,index)

```
  (token state ',index)),
'(critical state ',index),
me-prg(size))
```

THEOREM: nclcewlcorc-ensures-key
((*index* ∈ **N**) ∧ (*index* < *size*))
→ ensures-key(list('me, *index*, *size*),
me-prg(*size*),
*old*,
*new*,

```
'(and
  (non-critical state ',index)
  (token state ',index)),
'(or
  (and
   (wait state ',index)
   (token state ',index))
  (token state ',(add1-mod size index))))
```

THEOREM: nclcewlcorc-ensures-rest
((*index* ∈ **N**) ∧ (*index* < *size*))
→ ensures-rest(*statement*,
list('me, *index*, *size*),
me-prg(*size*),
*old*,
*new*,

```
'(and
  (non-critical state ',index)
  (token state ',index)),
```

*q*)

THEOREM: non-critical-left-ensures-wait-left-or-right
((*index* ∈ **N**) ∧ (*index* < *size*))
→ ensures('(and

```
  (non-critical state ',index)
  (token state ',index)),
'(or
  (and
   (wait state ',index)
   (token state ',index))
  (token state ',(add1-mod size index))),
```

me-prg(*size*))

THEOREM: non-critical-left-unless-wait-left-or-right
((*index* ∈ **N**) ∧ (*index* < *size*))

→ unless (‘(and
(non-critical state ’,index)
(token state ’,index)),
‘(or
(and
(wait state ’,index)
(token state ’,index))
(token state ’,(add1-mod size index))),
me-prg (*size*))

THEOREM: listp-me-prg
(*size* ≉ 0) → listp (me-prg (*size*))

THEOREM: mutual-exclusionp-is-invariant
(initial-condition (‘(mutual-exclusionp state ’,size), me-prg (*size*))
∧ (*size* ≉ 0))
→ invariant (‘(mutual-exclusionp state ’,size), me-prg (*size*))

THEOREM: critical-leads-to-less-critical-or-right
((*index* ∈ **N**) ∧ (*index* < *size*))
→ leads-to (‘(and
(critical state ’,index)
(lessp
(ticks state ’,index)
’,(add1 ticks))),
‘(or
(and
(critical state ’,index)
(lessp (ticks state ’,index) ’,ticks))
(token state ’,(add1-mod size index))),
me-prg (*size*))

THEOREM: critical-ticks-leads-to-right
((*index* ∈ **N**) ∧ (*index* < *size*))
→ leads-to (‘(and
(critical state ’,index)
(lessp
(ticks state ’,index)
’,(add1 ticks))),
‘(token state ’,(add1-mod size index)),
me-prg (*size*))

THEOREM: critical-leads-to-right
((*index* ∈ **N**) ∧ (*index* < *size*))
→ leads-to (‘(critical state ’,index),

```
‘(token state ’,(add1-mod size index)),
me-prg (size))
```

THEOREM: wait-left-leads-to-critical
((*index* ∈ **N**) ∧ (*index* < *size*))
→ leads-to (‘(and
(wait state ’,index)
(token state ’,index)),
‘(critical state ’,index),
me-prg (*size*))

THEOREM: non-critical-left-leads-to-wait-left-or-right
((*index* ∈ **N**) ∧ (*index* < *size*))
→ leads-to (‘(and
(non-critical state ’,index)
(token state ’,index)),
‘(or
(and
(wait state ’,index)
(token state ’,index))
(token state ’,(add1-mod size index))),
me-prg (*size*))

THEOREM: wait-left-leads-to-right
((*index* ∈ **N**) ∧ (*index* < *size*))
→ leads-to (‘(and
(wait state ’,index)
(token state ’,index)),
‘(token state ’,(add1-mod size index)),
me-prg (*size*))

THEOREM: non-critical-left-leads-to-right
((*index* ∈ **N**) ∧ (*index* < *size*))
→ leads-to (‘(and
(non-critical state ’,index)
(token state ’,index)),
‘(token state ’,(add1-mod size index)),
me-prg (*size*))

THEOREM: critical-wait-non-critical
critical (*state*, *index*) ∨ wait (*state*, *index*) ∨ non-critical (*state*, *index*)

DEFINITION:
tokens (*state*, *size*)
= **if** *size* ≃ 0 **then nil**

**elseif** token (*state*, *size* − 1)
**then** cons (*size* − 1, tokens (*state*, *size* − 1))
**else** tokens (*state*, *size* − 1) **endif**

DEFINITION:
criticals (*state*, *size*)
= **if** *size* ≃ 0 **then nil**
**elseif** critical (*state*, *size* − 1)
**then** cons (*size* − 1, criticals (*state*, *size* − 1))
**else** criticals (*state*, *size* − 1) **endif**

THEOREM: equal-plus-1
((*a* + *b*) = 1)
= (((*a* = 1) ∧ (*b* ≃ 0)) ∨ ((*b* = 1) ∧ (*a* ≃ 0)))

THEOREM: about-mutual-exclusionp
**if** weight (*state*, *size*) ≃ 0
**then** (tokens (*state*, *size*) = **nil**) ∧ (criticals (*state*, *size*) = **nil**)
**elseif** weight (*state*, *size*) = 1
**then** ((length (tokens (*state*, *size*)) = 1)
∧ (criticals (*state*, *size*) = **nil**))
∨ ((length (criticals (*state*, *size*)) = 1)
∧ (tokens (*state*, *size*) = **nil**))
**else t endif**

EVENT: Disable equal-plus-1.

THEOREM: numberp-car-tokens
car (tokens (*state*, *size*)) ∈ **N**

THEOREM: numberp-car-criticals
car (criticals (*state*, *size*)) ∈ **N**

THEOREM: lessp-car-tokens
(car (tokens (*state*, *size*)) < *size*) = (*size* ≄ 0)

THEOREM: lessp-car-criticals
(car (criticals (*state*, *size*)) < *size*) = (*size* ≄ 0)

THEOREM: mutual-exclusionp-requires-state
mutual-exclusionp (*state*, *size*) → (*size* ≄ 0)

THEOREM: lessp-car-tokens-and-criticals
mutual-exclusionp (*state*, *size*)
→ ((car (tokens (*state*, *size*)) < *size*)
∧ (car (criticals (*state*, *size*)) < *size*))

THEOREM: about-member-list-length-one
$(\text{length}\,(\mathit{list}) = \mathtt{1}) \rightarrow ((x \in \mathit{list}) = (x = \text{car}\,(\mathit{list})))$

THEOREM: about-member-car-tokens-criticals
$\text{mutual-exclusionp}\,(\mathit{state},\ \mathit{size})$
$\rightarrow \quad (((\mathit{index} \in \text{tokens}\,(\mathit{state},\ \mathit{size}))$
$\quad = \quad (\text{listp}\,(\text{tokens}\,(\mathit{state},\ \mathit{size}))$
$\quad \wedge \quad (\mathit{index} = \text{car}\,(\text{tokens}\,(\mathit{state},\ \mathit{size})))))$
$\wedge \quad ((\mathit{index} \in \text{criticals}\,(\mathit{state},\ \mathit{size}))$
$\quad = \quad (\text{listp}\,(\text{criticals}\,(\mathit{state},\ \mathit{size}))$
$\quad \wedge \quad (\mathit{index} = \text{car}\,(\text{criticals}\,(\mathit{state},\ \mathit{size}))))))$

THEOREM: not-member-tokens
$(\mathit{index} \not< \mathit{size}) \rightarrow (\mathit{index} \notin \text{tokens}\,(\mathit{state},\ \mathit{size}))$

THEOREM: not-member-criticals
$(\mathit{index} \not< \mathit{size}) \rightarrow (\mathit{index} \notin \text{criticals}\,(\mathit{state},\ \mathit{size}))$

THEOREM: token-equals
$((\mathit{index} \in \mathbf{N}) \wedge (\mathit{index} < \mathit{size}))$
$\rightarrow \quad (\text{token}\,(\mathit{state},\ \mathit{index}) = (\mathit{index} \in \text{tokens}\,(\mathit{state},\ \mathit{size})))$

THEOREM: critical-equals
$((\mathit{index} \in \mathbf{N}) \wedge (\mathit{index} < \mathit{size}))$
$\rightarrow \quad (\text{critical}\,(\mathit{state},\ \mathit{index}) = (\mathit{index} \in \text{criticals}\,(\mathit{state},\ \mathit{size})))$

THEOREM: mtoken-equals
$(\text{mutual-exclusionp}\,(\mathit{state},\ \mathit{size}) \wedge (\mathit{index} \in \mathbf{N}) \wedge (\mathit{index} < \mathit{size}))$
$\rightarrow \quad (\text{token}\,(\mathit{state},\ \mathit{index})$
$\quad = \quad (\text{listp}\,(\text{tokens}\,(\mathit{state},\ \mathit{size}))$
$\quad \wedge \quad (\mathit{index} = \text{car}\,(\text{tokens}\,(\mathit{state},\ \mathit{size})))))$

THEOREM: mcritical-equals
$(\text{mutual-exclusionp}\,(\mathit{state},\ \mathit{size}) \wedge (\mathit{index} \in \mathbf{N}) \wedge (\mathit{index} < \mathit{size}))$
$\rightarrow \quad (\text{critical}\,(\mathit{state},\ \mathit{index})$
$\quad = \quad (\text{listp}\,(\text{criticals}\,(\mathit{state},\ \mathit{size}))$
$\quad \wedge \quad (\mathit{index} = \text{car}\,(\text{criticals}\,(\mathit{state},\ \mathit{size})))))$

THEOREM: mutual-exclusionp-implies
$\text{mutual-exclusionp}\,(\mathit{state},\ \mathit{size})$
$\rightarrow \quad (\text{listp}\,(\text{tokens}\,(\mathit{state},\ \mathit{size})) \leftrightarrow (\neg\ \text{listp}\,(\text{criticals}\,(\mathit{state},\ \mathit{size}))))$

EVENT: Disable lessp-car-criticals.

EVENT: Disable lessp-car-tokens.

EVENT: Disable about-member-list-length-one.

EVENT: Disable not-member-tokens.

EVENT: Disable not-member-criticals.

EVENT: Disable token.

EVENT: Disable critical.

EVENT: Disable non-critical.

EVENT: Disable wait.

EVENT: Disable mutual-exclusionp.

THEOREM: mutual-exclusionp-implies-prg
mutual-exclusionp (*state*, *size*) → listp (me-prg (*size*))

THEOREM: left-leads-to-right
((*index* ∈ **N**)
∧ (*index* < *size*)
∧ initial-condition (`'(mutual-exclusionp state ',size)`,
me-prg (*size*)))
→ leads-to (`'(token state ',index)`,
`'(token state ',(add1-mod size index))`,
me-prg (*size*))

DEFINITION:
walk-ring (*a*, *b*, *size*)
= **if** *a* ∈ **N**
**then if** *b* ∈ **N**
**then if** *a* < *size*
**then if** *b* < *size*
**then if** *a* = *b* **then t**
**else** walk-ring (add1-mod (*size*, *a*), *b*, *size*) **endif**
**else t endif**
**else t endif**
**else t endif**
**else t endif**

THEOREM: any-left-leads-to-right
((*left* ∈ **N**)
∧ (*left* < *size*)
∧ (*right* ∈ **N**)
∧ (*right* < *size*)
∧ initial-condition (`(mutual-exclusionp state ,size),
me-prg (*size*)))
→ leads-to (`(token state ,left),
`(token state ,right),
me-prg (*size*))

THEOREM: lessp-add1-mod-size
mutual-exclusionp (*state*, *size*) → (add1-mod (*size*, *n*) < *size*)

THEOREM: any-critical-leads-to-right
((*left* ∈ **N**)
∧ (*left* < *size*)
∧ (*right* ∈ **N**)
∧ (*right* < *size*)
∧ initial-condition (`(mutual-exclusionp state ,size),
me-prg (*size*)))
→ leads-to (`(critical state ,left),
`(token state ,right),
me-prg (*size*))

THEOREM: any-leads-to-right
((*index* ∈ **N**)
∧ (*index* < *size*)
∧ initial-condition (`(mutual-exclusionp state ,size),
me-prg (*size*)))
→ leads-to (`(true), `(token state ,index), me-prg (*size*))

THEOREM: me-prg-non-zero-size
listp (me-prg (*size*)) = (*size* ≠ 0)

THEOREM: wait-leads-to-left-wait-or-critical
((*index* ∈ **N**)
∧ (*index* < *size*)
∧ initial-condition (`(mutual-exclusionp state ,size),
me-prg (*size*)))
→ leads-to (`(and (true) (wait state ,index)),
`(or
(and
(token state ,index)
(wait state ,index))

```
    (critical state ',index)),
  me-prg (size))
```

THEOREM: wait-leads-to-critical

```
((index ∈ N)
 ∧  (index < size)
 ∧  initial-condition (`(mutual-exclusionp state ',size),
                       me-prg (size)))
 →  leads-to (`(wait state ',index),
              `(critical state ',index),
              me-prg (size))
```

# Index