```
#|

 Copyright (C) 1994 by Robert S. Boyer and J Strother Moore.  All Rights
 Reserved.

 This script is hereby placed in the public domain, and therefore unlimited
 editing and redistribution is permitted.

 NO WARRANTY

 Robert S. Boyer and J Strother Moore 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 Robert S. Boyer or J Strother Moore 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.

|#

; This is the list of verification conditions for a FORTRAN
; version of our fast string searching algorithm.  For the details
; of the algorithm, see the comment at the end of the file.  Boyer and Moore.

; This list of events has been further edited, for processing by
; DO-FILE, by (1) inserting the following NOTE-LIB, (2) commenting out
; each FORTRAN-COMMENT and following the comment with the
; corresponding macroexpansion, and (3) by commenting out each
; (COMMENT ...).
```

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

```
; (FORTRAN-COMMENT FORTRAN)
```

AXIOM: fortran
t

EVENT: Introduce the function symbol *a$0* of 0 arguments.

EVENT: Introduce the function symbol *blk-delta1$0* of 0 arguments.

EVENT: Introduce the function symbol *blk-delta1$1* of 0 arguments.

EVENT: Introduce the function symbol *blk-delta1$2* of 0 arguments.

EVENT: Introduce the function symbol *blk-delta1$3* of 0 arguments.

EVENT: Introduce the function symbol *blk-delta1$4* of 0 arguments.

EVENT: Introduce the function symbol *c$0* of 0 arguments.

EVENT: Introduce the function symbol *c$1* of 0 arguments.

EVENT: Introduce the function symbol *c$2* of 0 arguments.

EVENT: Introduce the function symbol *c$3* of 0 arguments.

EVENT: Introduce the function symbol *c$4* of 0 arguments.

EVENT: Introduce the function symbol *i$1* of 0 arguments.

EVENT: Introduce the function symbol *i$2* of 0 arguments.

EVENT: Introduce the function symbol *i$3* of 0 arguments.

EVENT: Introduce the function symbol *i$4* of 0 arguments.

EVENT: Introduce the function symbol *maxi$0* of 0 arguments.

EVENT: Introduce the function symbol *asize&* of 0 arguments.

AXIOM: asize&-numberp

ASIZE& $\in \mathbf{N}$

DEFINITION:
delta1 (a, c, *maxi*)
= **if** *maxi* $\simeq$ `0` **then** `0`
elseif c = elt1 (a, *maxi*) **then** `0`
else 1 + delta1 (a, c, *maxi* − 1) **endif**

DEFINITION:
stringp (a, i, *size*)
= **if** $i \simeq$ `0` **then t**
else (elt1 (a, i) $\in \mathbf{N}$)
$\wedge$ (elt1 (a, i) $\neq$ `0`)
$\wedge$ (*size* $\not<$ elt1 (a, i))
$\wedge$ stringp (a, i − 1, *size*) **endif**

AXIOM: input-conditions
`'*1*true`

DEFINITION:
GLOBAL-HYPS
= ((ASIZE& $\neq$ `'0`)
$\wedge$ (stringp (A$0, MAXI$0, ASIZE&)
$\wedge$ ((MAXI$0 $\neq$ `'0`)
$\wedge$ (((1 + ASIZE&)
< LEAST-INEXPRESSIBLE-POSITIVE-INTEGER)
$\wedge$ (((1 + MAXI$0)
< LEAST-INEXPRESSIBLE-POSITIVE-INTEGER)
$\wedge$ (MAXI$0 $\in \mathbf{N}$))))))

THEOREM: stringp-is-a-univ-quantifier
(stringp (a, i, ASIZE&) $\wedge$ ($j \neq$ `0`) $\wedge$ ($j \in \mathbf{N}$) $\wedge$ ($i \not< j$))
$\rightarrow$ ((elt1 (a, j) $\in \mathbf{N}$) $\wedge$ (ASIZE& $\not<$ elt1 (a, j)) $\wedge$ (elt1 (a, j) $\neq$ `0`))

THEOREM: input-definedness
GLOBAL-HYPS $\rightarrow$ `'*1*true`

```
#| (COMMENT) |#

; (FORTRAN-COMMENT INPUT)
```

AXIOM: input
t

```
; (FORTRAN-COMMENT LOGICAL-IF-F)
```

AXIOM: logical-if-f
t

AXIOM: assignment
$(\text{I\$1} = \texttt{'1}) \wedge ((\text{C\$1} = \text{C\$0}) \wedge (\text{BLK-DELTA1\$1} = \text{BLK-DELTA1\$0}))$

THEOREM: pass1-invrt
GLOBAL-HYPS
$\rightarrow \quad (((\text{I\$1} \in \mathbf{N})$
$\wedge \quad ((\text{I\$1} \neq \texttt{'0})$
$\wedge \quad (((1 + \text{ASIZE\&}) \not< \text{I\$1})$
$\wedge \quad (((j \in \mathbf{N}) \wedge ((j \neq \texttt{'0}) \wedge (j < \text{I\$1})))$
$\rightarrow \quad (\text{elt1}\,(\text{BLK-DELTA1\$1}, j) = \text{MAXI\$0})))))$
$\wedge \quad \text{lex}\,(\text{cons}\,((\texttt{'2} + (\text{ASIZE\&} + \text{MAXI\$0})) - \text{I\$1}, \texttt{'nil}),$
$\text{cons}\,(\texttt{'2} + (\text{ASIZE\&} + \text{MAXI\$0}), \texttt{'nil})))$

```
#| (COMMENT INPUT F) |#
```

EVENT: Undo back through the event named 'input'.

AXIOM: paths-from-pass1-invrt
$((j \in \mathbf{N}) \wedge ((j \neq \texttt{'0}) \wedge (j < \text{I\$1})))$
$\rightarrow \quad (\text{elt1}\,(\text{BLK-DELTA1\$1}, j) = \text{MAXI\$0})$

DEFINITION:
PATH-HYPS
$= \quad (\text{GLOBAL-HYPS}$
$\wedge \quad ((\text{I\$1} \in \mathbf{N}) \wedge ((\text{I\$1} \neq \texttt{'0}) \wedge ((1 + \text{ASIZE\&}) \not< \text{I\$1}))))$

THEOREM: definedness
PATH-HYPS $\rightarrow$ znumberp (I\$1)

```
#| (COMMENT PASS1-INVRT) |#

; (FORTRAN-COMMENT LOGICAL-IF-T)
```

AXIOM: logical-if-t
t

AXIOM: effects-of-undefiner
$(\text{C\$2} = \text{C\$1}) \wedge (\text{BLK-DELTA1\$2} = \text{BLK-DELTA1\$1})$

```
; (FORTRAN-COMMENT LOGICAL-IF-F1)
```

AXIOM: logical-if-f1
t

AXIOM: assignment1
(I\$3 = '1) ∧ ((C\$3 = C\$2) ∧ (BLK-DELTA1\$3 = BLK-DELTA1\$2))

THEOREM: pass2-invrt
(I\$1 ∧ (zgreaterp (I\$1, ASIZE&) ∧ PATH-HYPS))
→ (((I\$3 ∈ **N**)
∧ ((I\$3 ≠ '0)
∧ (((1 + MAXI\$0) ≮ I\$3)
∧ (((*j* ≠ 0) ∧ (ASIZE& ≮ *j*))
→ (elt1 (BLK-DELTA1\$3, *j*)
= (delta1 (A\$0, *j*, I\$3 − 1)
+ (MAXI\$0
− (I\$3 − 1))))))))
∧ lex (cons ((1 + MAXI\$0) − I\$3, 'nil),
cons (('2 + (ASIZE& + MAXI\$0)) − I\$1, 'nil)))

```
#| (COMMENT PASS1-INVRT T F) |#
```

EVENT: Undo back through the event named 'logical-if-t'.

```
; (FORTRAN-COMMENT LOGICAL-IF-F1)
```

AXIOM: logical-if-f1
t

THEOREM: array-bounds-check-for-delta1
((¬ zgreaterp (I\$1, ASIZE&)) ∧ PATH-HYPS)
→ (('0 < I\$1) ∧ (ASIZE& ≮ I\$1))

```
#| (COMMENT PASS1-INVRT F) |#
```

AXIOM: assignment1
(I\$2 = I\$1)
∧ ((C\$2 = C\$1)
∧ (elt1 (BLK-DELTA1\$2, *i*)
= **if** *i* = I\$1 **then** MAXI\$0
else elt1 (BLK-DELTA1\$1, *i*) **endif**))

THEOREM: input-cond-of-zplus
(almost-equal1 (BLK-DELTA1\$1, BLK-DELTA1\$2, '1, ASIZE&, I\$1, MAXI\$0)
∧ ((¬ zgreaterp (I\$1, ASIZE&)) ∧ PATH-HYPS))
→ expressible-znumberp (zplus (I\$2, '1))

```
#| (COMMENT PASS1-INVRT F) |#
```

AXIOM: assignment2
(I\$3 = zplus (I\$2, '1))
∧ ((C\$3 = C\$2) ∧ (BLK-DELTA1\$3 = BLK-DELTA1\$2))

THEOREM: pass1-invrt1
(almost-equal1 (BLK-DELTA1\$1, BLK-DELTA1\$2, '1, ASIZE&, I\$1, MAXI\$0)
∧ ((¬ zgreaterp (I\$1, ASIZE&)) ∧ PATH-HYPS))
→ (((I\$3 ∈ **N**)
∧ ((I\$3 ≠ '0)
∧ (((1 + ASIZE&) ≮ I\$3)
∧ (((j ∈ **N**) ∧ ((j ≠ '0) ∧ (j < I\$3)))
→ (elt1 (BLK-DELTA1\$3, j) = MAXI\$0)))))
∧ lex (cons (('2 + (ASIZE& + MAXI\$0)) − I\$3, 'nil),
cons (('2 + (ASIZE& + MAXI\$0)) − I\$1, 'nil)))

```
#| (COMMENT PASS1-INVRT F) |#
```

EVENT: Undo back through the event named 'paths-from-pass1-invrt'.

AXIOM: paths-from-pass2-invrt
((j ≉ 0) ∧ (ASIZE& ≮ j))
→ (elt1 (BLK-DELTA1\$1, j)
= (delta1 (A\$0, j, I\$1 − 1) + (MAXI\$0 − (I\$1 − 1))))

DEFINITION:
PATH-HYPS
= (GLOBAL-HYPS
∧ ((I\$1 ∈ **N**) ∧ ((I\$1 ≠ '0) ∧ ((1 + MAXI\$0) ≮ I\$1))))

THEOREM: definedness
PATH-HYPS → znumberp (I\$1)

```
#| (COMMENT PASS2-INVRT) |#

; (FORTRAN-COMMENT LOGICAL-IF-T)
```

AXIOM: logical-if-t
t

AXIOM: effects-of-undefiner
(C\$2 = C\$1) ∧ (BLK-DELTA1\$2 = BLK-DELTA1\$1)

THEOREM: output
(I$1 ∧ (zgreaterp (I$1, MAXI$0) ∧ PATH-HYPS))
→ (((c ∈ **N**) ∧ ((c ≠ '0) ∧ (ASIZE& ≮ c)))
→ (elt1 (BLK-DELTA1$2, c) = delta1 (A$0, c, MAXI$0)))

```
#| (COMMENT PASS2-INVRT T) |#
```

EVENT: Undo back through the event named 'logical-if-t'.

```
; (FORTRAN-COMMENT LOGICAL-IF-F1)
```

AXIOM: logical-if-f1
t

THEOREM: array-bounds-check-for-a
((¬ zgreaterp (I$1, MAXI$0)) ∧ PATH-HYPS)
→ (('0 < I$1) ∧ (MAXI$0 ≮ I$1))

```
#| (COMMENT PASS2-INVRT F) |#
```

THEOREM: definedness1
((¬ zgreaterp (I$1, MAXI$0)) ∧ PATH-HYPS) → znumberp (elt1 (A$0, I$1))

```
#| (COMMENT PASS2-INVRT F) |#
```

AXIOM: assignment1
(I$2 = I$1)
∧ ((C$2 = elt1 (A$0, I$1)) ∧ (BLK-DELTA1$2 = BLK-DELTA1$1))

THEOREM: input-cond-of-zdifference
((¬ zgreaterp (I$1, MAXI$0)) ∧ PATH-HYPS)
→ expressible-znumberp (zdifference (MAXI$0, I$2))

```
#| (COMMENT PASS2-INVRT F) |#
```

THEOREM: array-bounds-check-for-delta1
((¬ zgreaterp (I$1, MAXI$0)) ∧ PATH-HYPS)
→ (('0 < C$2) ∧ (ASIZE& ≮ C$2))

```
#| (COMMENT PASS2-INVRT F) |#
```

AXIOM: assignment2
(I$3 = I$2)
∧ ((C$3 = C$2)
∧ (elt1 (BLK-DELTA1$3, i)
= **if** i = C$2 **then** zdifference (MAXI$0, I$2)
else elt1 (BLK-DELTA1$2, i) **endif**))

THEOREM: input-cond-of-zplus
(almost-equal1 (BLK-DELTA1$2,
BLK-DELTA1$3,
'1,
ASIZE&,
C$2,
zdifference (MAXI$0, I$2))
∧ ((¬ zgreaterp (I$1, MAXI$0)) ∧ PATH-HYPS))
→ expressible-znumberp (zplus (I$3, '1))

```
#| (COMMENT PASS2-INVRT F) |#
```

AXIOM: assignment3
(I$4 = zplus (I$3, '1))
∧ ((C$4 = C$3) ∧ (BLK-DELTA1$4 = BLK-DELTA1$3))

THEOREM: delta1-skip
((*j* ≠ elt1 (*pat*, *i*)) ∧ (*i* ≯ 0))
→ (delta1 (*pat*, *j*, *i*) = (1 + delta1 (*pat*, *j*, *i* − 1)))

THEOREM: pass2-invrt1
(almost-equal1 (BLK-DELTA1$2,
BLK-DELTA1$3,
'1,
ASIZE&,
C$2,
zdifference (MAXI$0, I$2))
∧ ((¬ zgreaterp (I$1, MAXI$0)) ∧ PATH-HYPS))
→ (((I$4 ∈ **N**)
∧ ((I$4 ≠ '0)
∧ (((1 + MAXI$0) ≮ I$4)
∧ (((*j* ≯ 0) ∧ (ASIZE& ≮ *j*))
→ (elt1 (BLK-DELTA1$4, *j*)
= (delta1 (A$0, *j*, I$4 − 1)
+ (MAXI$0
− (I$4 − 1))))))))
∧ lex (cons ((1 + MAXI$0) − I$4, 'nil),
cons ((1 + MAXI$0) − I$1, 'nil)))

```
#| (COMMENT PASS2-INVRT F) |#
```

EVENT: Undo back through the event named 'paths-from-pass2-invrt'.

EVENT: Undo back through the event named 'fortran'.

```
; (FORTRAN-COMMENT FORTRAN)
```

AXIOM: fortran
t

EVENT: Introduce the function symbol *blk-delta1$1* of 0 arguments.

EVENT: Introduce the function symbol *blk-delta1$2* of 0 arguments.

EVENT: Introduce the function symbol *blk-delta1$3* of 0 arguments.

EVENT: Introduce the function symbol *blk-delta1$4* of 0 arguments.

EVENT: Introduce the function symbol *c$0* of 0 arguments.

EVENT: Introduce the function symbol *c$1* of 0 arguments.

EVENT: Introduce the function symbol *c$2* of 0 arguments.

EVENT: Introduce the function symbol *c$3* of 0 arguments.

EVENT: Introduce the function symbol *c$4* of 0 arguments.

EVENT: Introduce the function symbol *i$0* of 0 arguments.

EVENT: Introduce the function symbol *i$1* of 0 arguments.

EVENT: Introduce the function symbol *i$2* of 0 arguments.

EVENT: Introduce the function symbol *i$3* of 0 arguments.

EVENT: Introduce the function symbol *i$4* of 0 arguments.

EVENT: Introduce the function symbol *j$0* of 0 arguments.

EVENT: Introduce the function symbol *j$1* of 0 arguments.

EVENT: Introduce the function symbol *j$2* of 0 arguments.

EVENT: Introduce the function symbol *j$3* of 0 arguments.

EVENT: Introduce the function symbol *j$4* of 0 arguments.

EVENT: Introduce the function symbol *nexti$0* of 0 arguments.

EVENT: Introduce the function symbol *nexti$1* of 0 arguments.

EVENT: Introduce the function symbol *nexti$2* of 0 arguments.

EVENT: Introduce the function symbol *nexti$3* of 0 arguments.

EVENT: Introduce the function symbol *nexti$4* of 0 arguments.

EVENT: Introduce the function symbol *pat$0* of 0 arguments.

EVENT: Introduce the function symbol *patlen$0* of 0 arguments.

EVENT: Introduce the function symbol *str$0* of 0 arguments.

EVENT: Introduce the function symbol *strlen$0* of 0 arguments.

EVENT: Introduce the function symbol *x$0* of 0 arguments.

EVENT: Introduce the function symbol *x$1* of 0 arguments.

EVENT: Introduce the function symbol *x$2* of 0 arguments.

EVENT: Introduce the function symbol *x$3* of 0 arguments.

EVENT: Introduce the function symbol *x$4* of 0 arguments.

EVENT: Introduce the function symbol *asize&* of 0 arguments.

AXIOM: asize&-numberp
ASIZE& $\in$ **N**

DEFINITION:
delta1 (a, c, *maxi*)
= **if** *maxi* $\simeq$ 0 **then** 0
elseif c = elt1 (a, *maxi*) **then** 0
else 1 + delta1 (a, c, *maxi* $-$ 1) **endif**

DEFINITION:
stringp (a, i, *size*)
= **if** $i \simeq$ 0 **then t**
else (elt1 (a, i) $\in$ **N**)
$\wedge$ (elt1 (a, i) $\neq$ 0)
$\wedge$ (*size* $\not<$ elt1 (a, i))
$\wedge$ stringp (a, $i-1$, *size*) **endif**

THEOREM: stringp-is-a-univ-quantifier
(stringp (a, i, ASIZE&) $\wedge$ ($j \neq$ 0) $\wedge$ ($j \in$ **N**) $\wedge$ ($i \not< j$))
$\rightarrow$ ((elt1 (a, j) $\in$ **N**) $\wedge$ (ASIZE& $\not<$ elt1 (a, j)) $\wedge$ (elt1 (a, j) $\neq$ 0))

DEFINITION:
match (*pat*, j, *patlen*, *str*, i, *strlen*)
= **if** *patlen* $< j$ **then t**
elseif *strlen* $< i$ **then f**
else (elt1 (*pat*, j) = elt1 (*str*, i))
$\wedge$ match (*pat*, $1+j$, *patlen*, *str*, $1+i$, *strlen*) **endif**

DEFINITION:
search (*pat*, *str*, *patlen*, *strlen*, i)
= **if** *strlen* $< i$ **then** 1 + *strlen*
elseif match (*pat*, 1, *patlen*, *str*, i, *strlen*) **then** i
else search (*pat*, *str*, *patlen*, *strlen*, $1+i$) **endif**

AXIOM: input-conditions
'*1*true

DEFINITION:
GLOBAL-HYPS
$$
\begin{array}{ll}
= & ((\text{ASIZE\&} \neq \texttt{'0}) \\
\wedge & (((1 + \text{ASIZE\&}) < \text{LEAST-INEXPRESSIBLE-POSITIVE-INTEGER}) \\
\wedge & (\text{stringp}\,(\text{PAT\$0}, \text{PATLEN\$0}, \text{ASIZE\&}) \\
\wedge & ((\texttt{'0} < \text{PATLEN\$0}) \\
\wedge & (\text{stringp}\,(\text{STR\$0}, \text{STRLEN\$0}, \text{ASIZE\&}) \\
\wedge & ((\texttt{'0} < \text{STRLEN\$0}) \\
\wedge & (((\text{PATLEN\$0} \\
& + \text{STRLEN\$0}) \\
& < \text{LEAST-INEXPRESSIBLE-POSITIVE-INTEGER}) \\
\wedge & ((\text{PATLEN\$0} \in \mathbf{N}) \\
\wedge & (\text{STRLEN\$0} \in \mathbf{N}))))))))))
\end{array}
$$

THEOREM: search-non-zero
$(i \not\simeq \texttt{0}) \rightarrow (\texttt{0} < \text{search}\,(pat, str, patlen, strlen, i))$

THEOREM: match-at-patlen
$(patlen < j) \rightarrow \text{match}\,(pat, j, patlen, str, i, strlen)$

THEOREM: match-at-strlen
$(strlen < i) \rightarrow (\text{match}\,(pat, j, patlen, str, i, strlen) = (patlen < j))$

THEOREM: match-needs-patlen-chars
$$
\begin{array}{ll}
 & ((patlen \in \mathbf{N}) \\
\wedge & (strlen \in \mathbf{N}) \\
\wedge & (patlen \not< j) \\
\wedge & (strlen \not< i) \\
\wedge & ((j + strlen) < (i + patlen))) \\
\rightarrow & (\neg\, \text{match}\,(pat, j, patlen, str, i, strlen))
\end{array}
$$

THEOREM: search-boundary
$$
\begin{array}{ll}
 & ((patlen \not\simeq \texttt{0}) \\
\wedge & (strlen \in \mathbf{N}) \\
\wedge & (strlen \not< i) \\
\wedge & ((1 + strlen) < (patlen + \text{search}\,(pat, str, patlen, strlen, i)))) \\
\rightarrow & (\text{search}\,(pat, str, patlen, strlen, i) = (1 + strlen))
\end{array}
$$

THEOREM: delta1-lesseqp-patlen
$patlen \not< \text{delta1}\,(pat, char, patlen)$

THEOREM: input-definedness
GLOBAL-HYPS $\rightarrow$ `'*1*true`

```
#| (COMMENT) |#

; (FORTRAN-COMMENT INPUT)
```

AXIOM: input
t

THEOREM: call-of-setup
(PATLEN$0 ∧ GLOBAL-HYPS)
→ (stringp (PAT$0, PATLEN$0, ASIZE&)
∧ ((PATLEN$0 ≠ '0)
∧ ((PATLEN$0 ∈ **N**)
∧ (((1 + ASIZE&)
< LEAST-INEXPRESSIBLE-POSITIVE-INTEGER)
∧ ((1 + PATLEN$0)
< LEAST-INEXPRESSIBLE-POSITIVE-INTEGER)))))

```
#| (COMMENT INPUT) |#
```

AXIOM: effects-of-setup
(((*c* ∈ **N**) ∧ ((*c* ≠ '0) ∧ (ASIZE& ⊀ *c*)))
→ (elt1 (BLK-DELTA1$1, *c*) = delta1 (PAT$0, *c*, PATLEN$0)))
∧ ((I$1 = I$0)
∧ ((J$1 = J$0)
∧ ((C$1 = C$0)
∧ ((NEXTI$1 = NEXTI$0) ∧ (X$1 = X$0))))))

AXIOM: assignment
(I$2 = PATLEN$0)
∧ ((J$2 = J$1)
∧ ((C$2 = C$1)
∧ ((NEXTI$2 = NEXTI$1)
∧ ((X$2 = X$1)
∧ (BLK-DELTA1$2 = BLK-DELTA1$1))))))

THEOREM: outer-invrt
(PATLEN$0 ∧ GLOBAL-HYPS)
→ (((I$2 ∈ **N**)
∧ ((I$2 ⊀ PATLEN$0)
∧ ((I$2
< (PATLEN$0
+ search (PAT$0,
STR$0,
PATLEN$0,
STRLEN$0,
'1)))
∧ (((*c* ∈ **N**)
∧ ((*c* ≠ '0) ∧ (ASIZE& ⊀ *c*)))

$$\begin{array}{l} \rightarrow \quad (\text{elt1}\,(\text{BLK-DELTA1\$2},\ c) \\ \qquad = \quad \text{delta1}\,(\text{PAT\$0},\ c,\ \text{PATLEN\$0}))))))) \\ \wedge \quad \text{lex}\,(\text{cons}\,((\texttt{'1} + (\text{STRLEN\$0} + \text{PATLEN\$0})) - \text{I\$2}, \\ \qquad \text{cons}\,(1 + \text{PATLEN\$0},\ \texttt{'nil})), \\ \qquad \text{cons}\,(\texttt{'1} + (\text{STRLEN\$0} + \text{PATLEN\$0}), \\ \qquad \text{cons}\,(1 + \text{PATLEN\$0},\ \texttt{'nil})))) \end{array}$$

```
#| (COMMENT INPUT) |#
```

EVENT: Undo back through the event named 'input'.

AXIOM: paths-from-outer-invrt
$((c \in \mathbf{N}) \wedge ((c \neq \texttt{'0}) \wedge (\text{ASIZE\&} \not< c)))$
$\rightarrow \quad (\text{elt1}\,(\text{BLK-DELTA1\$1},\ c) = \text{delta1}\,(\text{PAT\$0},\ c,\ \text{PATLEN\$0}))$

DEFINITION:
PATH-HYPS
$$\begin{array}{l} = \quad (\text{GLOBAL-HYPS} \\ \quad \wedge \quad ((\text{I\$1} \in \mathbf{N}) \\ \qquad \wedge \quad ((\text{I\$1} \not< \text{PATLEN\$0}) \\ \qquad\quad \wedge \quad (\text{I\$1} \\ \qquad\qquad < \quad (\text{PATLEN\$0} \\ \qquad\qquad\quad + \quad \text{search}\,(\text{PAT\$0}, \\ \qquad\qquad\qquad\qquad \text{STR\$0}, \\ \qquad\qquad\qquad\qquad \text{PATLEN\$0}, \\ \qquad\qquad\qquad\qquad \text{STRLEN\$0}, \\ \qquad\qquad\qquad\qquad \texttt{'1}))))))) \end{array}$$

THEOREM: definedness
PATH-HYPS $\rightarrow$ znumberp (I\$1)

```
#| (COMMENT OUTER-INVRT) |#

; (FORTRAN-COMMENT LOGICAL-IF-T)
```

AXIOM: logical-if-t
t

THEOREM: input-cond-of-zplus
(zgreaterp (I\$1, STRLEN\$0) $\wedge$ PATH-HYPS)
$\rightarrow$ expressible-znumberp (zplus (STRLEN\$0, 'T))

```
#| (COMMENT OUTER-INVRT T) |#
```

AXIOM: assignment1
(I$2 = I$1)
∧ ((J$2 = J$1)
∧ ((C$2 = C$1)
∧ ((NEXTI$2 = NEXTI$1)
∧ ((X$2 = zplus (STRLEN$0, '1))
∧ (BLK-DELTA1$2 = BLK-DELTA1$1)))))

THEOREM: output
(zgreaterp (I$1, STRLEN$0) ∧ PATH-HYPS)
→ (X$2 = search (PAT$0, STR$0, PATLEN$0, STRLEN$0, '1))

```
#| (COMMENT OUTER-INVRT T) |#
```

EVENT: Undo back through the event named 'logical-if-t'.

```
; (FORTRAN-COMMENT LOGICAL-IF-F)
```

AXIOM: logical-if-f
t

AXIOM: assignment1
(I$2 = I$1)
∧ ((J$2 = PATLEN$0)
∧ ((C$2 = C$1)
∧ ((NEXTI$2 = NEXTI$1)
∧ ((X$2 = X$1)
∧ (BLK-DELTA1$2 = BLK-DELTA1$1)))))

THEOREM: input-cond-of-zplus
((¬ zgreaterp (I$1, STRLEN$0)) ∧ PATH-HYPS)
→ expressible-znumberp (zplus ('1, I$2))

```
#| (COMMENT OUTER-INVRT F) |#
```

AXIOM: assignment2
(I$3 = I$2)
∧ ((J$3 = J$2)
∧ ((C$3 = C$2)
∧ ((NEXTI$3 = zplus ('1, I$2))
∧ ((X$3 = X$2)
∧ (BLK-DELTA1$3 = BLK-DELTA1$2)))))

```
(PROVE-LEMMA INNER-INVRT NIL
 (IMPLIES
  (AND (NOT (ZGREATERP (I$1) (STRLEN$0)))
       (PATH-HYPS))
  (AND
   (AND
    (NOT (LESSP (NEXTI$3) (ADD1 (PATLEN$0))))
    (AND
     (LESSP (NEXTI$3)
            (ADD1 (PLUS (PATLEN$0)
                        (SEARCH (PAT$0)
                                (STR$0)
                                (PATLEN$0)
                                (STRLEN$0)
                                '1))))
     (AND
      (IMPLIES (AND (NUMBERP C)
                    (AND (NOT (EQUAL C '0))
                         (NOT (LESSP (ASIZE&) C))))
               (EQUAL (ELT1 (BLK-DELTA1$3) C)
                      (DELTA1 (PAT$0) C (PATLEN$0))))
      (AND
       (NUMBERP (I$3))
       (AND
        (NOT (EQUAL (I$3) '0))
        (AND
         (NUMBERP (J$3))
         (AND
          (NOT (EQUAL (J$3) '0))
          (AND
           (NUMBERP (NEXTI$3))
           (AND
            (NOT (LESSP (PATLEN$0) (J$3)))
            (AND
             (NOT (LESSP (STRLEN$0) (I$3)))
             (AND
              (EQUAL (NEXTI$3)
                     (PLUS (ADD1 (PATLEN$0))
                           (DIFFERENCE (I$3) (J$3))))
              (AND
               (NOT (LESSP (ADD1 (STRLEN$0)) (NEXTI$3)))
               (AND
```

```
                  (NOT (LESSP (I$3) (J$3)))
                  (AND (MATCH (PAT$0)
                              (ADD1 (J$3))
                              (PATLEN$0)
                              (STR$0)
                              (ADD1 (I$3))
                              (STRLEN$0))
                       (AND (NUMBERP (ELT1 (BLK-DELTA1$3)
                                           (ELT1 (STR$0) (I$3))))
                            (AND (NUMBERP (ELT1 (STR$0) (I$3)))
                                 (NUMBERP (ELT1 (PAT$0) (J$3)))))))
   (LEX (CONS (DIFFERENCE (PLUS '1
                                (PLUS (STRLEN$0) (PATLEN$0)))
                          (SUB1 (NEXTI$3)))
              (CONS (J$3) 'NIL))
        (CONS (DIFFERENCE (PLUS '1
                                (PLUS (STRLEN$0) (PATLEN$0)))
                          (I$1))
              (CONS (ADD1 (PATLEN$0)) 'NIL))))))

#| (COMMENT OUTER-INVRT F) |#
```

EVENT: Undo back through the event named 'paths-from-outer-invrt'.

AXIOM: paths-from-inner-invrt
$(((c \in \mathbf{N}) \wedge ((c \neq \texttt{'0}) \wedge (\textsc{asize\&} \not< c)))$
$\rightarrow \quad (\text{elt1}\,(\textsc{blk-delta1\$1},\ c) = \text{delta1}\,(\textsc{pat\$0},\ c,\ \textsc{patlen\$0})))$
$\wedge \quad (\textsc{nexti\$1} = ((1 + \textsc{patlen\$0}) + (\textsc{i\$1} - \textsc{j\$1})))$

```
(DEFN PATH-HYPS NIL
 (AND
  (GLOBAL-HYPS)
  (AND
   (NOT (LESSP (NEXTI$1) (ADD1 (PATLEN$0))))
   (AND
    (LESSP (NEXTI$1)
           (ADD1 (PLUS (PATLEN$0)
                       (SEARCH (PAT$0)
                               (STR$0)
                               (PATLEN$0)
                               (STRLEN$0)
```

```
                                ’1))))
    (AND
     (NUMBERP (I$1))
     (AND
      (NOT (EQUAL (I$1) ’0))
      (AND
       (NUMBERP (J$1))
       (AND
        (NOT (EQUAL (J$1) ’0))
        (AND
         (NUMBERP (NEXTI$1))
         (AND
          (NOT (LESSP (PATLEN$0) (J$1)))
          (AND
           (NOT (LESSP (STRLEN$0) (I$1)))
           (AND
             (NOT (LESSP (ADD1 (STRLEN$0)) (NEXTI$1)))
             (AND (NOT (LESSP (I$1) (J$1)))
                  (AND (MATCH (PAT$0)
                              (ADD1 (J$1))
                              (PATLEN$0)
                              (STR$0)
                              (ADD1 (I$1))
                              (STRLEN$0))
                       (AND (NUMBERP (ELT1 (BLK-DELTA1$1)
                                           (ELT1 (STR$0) (I$1))))
                            (AND (NUMBERP (ELT1 (STR$0) (I$1)))
                                 (NUMBERP (ELT1 (PAT$0) (J$1))))))))))))))))))))
```

THEOREM: array-bounds-check-for-str
PATH-HYPS $\rightarrow$ ((’0 $<$ I$1) $\wedge$ (STRLEN$0 $\not<$ I$1))

```
#| (COMMENT INNER-INVRT) |#
```

THEOREM: definedness
PATH-HYPS $\rightarrow$ znumberp (elt1 (STR$0, I$1))

```
#| (COMMENT INNER-INVRT) |#
```

AXIOM: assignment1
(I$2 = I$1)
$\wedge$ ((J$2 = J$1)
$\wedge$ ((C$2 = elt1 (STR$0, I$1))

$\wedge \quad ((\text{NEXTI}\$2 = \text{NEXTI}\$1)$
$\wedge \quad ((\text{X}\$2 = \text{X}\$1)$
$\wedge \quad (\text{BLK-DELTA1}\$2 = \text{BLK-DELTA1}\$1)))))$

THEOREM: array-bounds-check-for-pat
PATH-HYPS $\rightarrow ((\text{'0} < \text{J}\$2) \wedge (\text{PATLEN}\$0 \not< \text{J}\$2))$

```
#| (COMMENT INNER-INVRT) |#
```

THEOREM: definedness1
PATH-HYPS $\rightarrow$ znumberp (elt1 (PAT$0, J$2))

```
#| (COMMENT INNER-INVRT) |#
```

```
; (FORTRAN-COMMENT LOGICAL-IF-T)
```

AXIOM: logical-if-t
t

THEOREM: array-bounds-check-for-delta1
(zneqp (C$2, elt1 (PAT$0, J$2)) $\wedge$ PATH-HYPS)
$\rightarrow \quad ((\text{'0} < \text{C}\$2) \wedge (\text{ASIZE\&} \not< \text{C}\$2))$

```
#| (COMMENT INNER-INVRT T) |#
```

THEOREM: definedness2
(zneqp (C$2, elt1 (PAT$0, J$2)) $\wedge$ PATH-HYPS)
$\rightarrow$ znumberp (elt1 (BLK-DELTA1$2, C$2))

```
#| (COMMENT INNER-INVRT T) |#
```

THEOREM: input-cond-of-zplus
(zneqp (C$2, elt1 (PAT$0, J$2)) $\wedge$ PATH-HYPS)
$\rightarrow$ expressible-znumberp (zplus (I$2, elt1 (BLK-DELTA1$2, C$2)))

```
#| (COMMENT INNER-INVRT T) |#
```

THEOREM: definedness3
(zneqp (C$2, elt1 (PAT$0, J$2)) $\wedge$ PATH-HYPS) $\rightarrow$ znumberp (NEXTI$2)

```
#| (COMMENT INNER-INVRT T) |#
```

AXIOM: assignment2
$(\text{I\$3} = \max0\,(\text{zplus}\,(\text{I\$2}, \text{elt1}\,(\text{BLK-DELTA1\$2}, \text{C\$2})), \text{NEXTI\$2}))$
$\wedge \quad ((\text{J\$3} = \text{J\$2})$
$\wedge \quad ((\text{C\$3} = \text{C\$2})$
$\wedge \quad ((\text{NEXTI\$3} = \text{NEXTI\$2})$
$\wedge \quad ((\text{X\$3} = \text{X\$2})$
$\wedge \quad (\text{BLK-DELTA1\$3} = \text{BLK-DELTA1\$2})))))$

THEOREM: delta1-plus-j-lesseqp-patlen
$((j \in \mathbf{N}) \wedge (j \neq 0) \wedge (patlen \not< j))$
$\rightarrow \quad (patlen \not< (j + \text{delta1}\,(pat, \text{elt1}\,(pat, j), patlen)))$

THEOREM: difference-plus-id
$((k + j) - j) = \text{fix}\,(k)$

THEOREM: match-implies-eq-chars
$(\text{match}\,(pat, j, patlen, str, k, strlen)$
$\wedge \quad (k \in \mathbf{N})$
$\wedge \quad (j \in \mathbf{N})$
$\wedge \quad (patlen \in \mathbf{N})$
$\wedge \quad (strlen \in \mathbf{N})$
$\wedge \quad (strlen \not< k)$
$\wedge \quad (patlen \not< j)$
$\wedge \quad (k \neq 0)$
$\wedge \quad (j \neq 0)$
$\wedge \quad (l \in \mathbf{N})$
$\wedge \quad (l \not< k)$
$\wedge \quad ((k + patlen) \not< (j + l)))$
$\rightarrow \quad (\text{elt1}\,(str, l) = \text{elt1}\,(pat, (j + l) - k))$

THEOREM: neq-chars-in-region-means-search-skips
$((\text{elt1}\,(str, i) \neq \text{elt1}\,(pat, j))$
$\wedge \quad (i \in \mathbf{N})$
$\wedge \quad (i \neq 0)$
$\wedge \quad (strlen \in \mathbf{N})$
$\wedge \quad (strlen \not< i)$
$\wedge \quad (j \in \mathbf{N})$
$\wedge \quad (j \neq 0)$
$\wedge \quad (patlen \in \mathbf{N})$
$\wedge \quad (patlen \not< j)$
$\wedge \quad (i \not< j)$
$\wedge \quad (k \in \mathbf{N})$
$\wedge \quad (k \neq 0)$
$\wedge \quad (strlen \not< k)$
$\wedge \quad ((\text{search}\,(pat, str, patlen, strlen, k) - 1) \not< (i - j)))$
$\rightarrow \quad ((i - j) < (\text{search}\,(pat, str, patlen, strlen, k) - 1))$

THEOREM: delta1-doesnt-go-beyond-end-of-match
$$
\begin{array}{ll}
(\text{match}\,(\mathit{pat},\ \mathtt{1},\ \mathit{patlen},\ \mathit{str},\ k,\ \mathit{strlen}) & \\
\wedge & (i \in \mathbf{N}) \\
\wedge & (\mathit{strlen} \not< i) \\
\wedge & (\mathit{patlen} \in \mathbf{N}) \\
\wedge & (\mathit{strlen} \in \mathbf{N}) \\
\wedge & (k \in \mathbf{N}) \\
\wedge & (k \neq \mathtt{0}) \\
\wedge & (i < (k + \mathit{patlen}))) \\
\rightarrow & (((i + \text{delta1}\,(\mathit{pat},\ \text{elt1}\,(\mathit{str},\ i),\ \mathit{patlen})) < (k + \mathit{patlen})) \\
 & = \quad ((i < k) \vee (i = k) \vee (k < i)))
\end{array}
$$

EVENT: Disable match-implies-eq-chars.

THEOREM: delta1-doesnt-go-beyond-search
$$
\begin{array}{ll}
((\text{elt1}\,(\mathit{str},\ i) \neq \text{elt1}\,(\mathit{pat},\ j)) & \\
\wedge & (i \in \mathbf{N}) \\
\wedge & (\mathit{strlen} \not< i) \\
\wedge & (j \in \mathbf{N}) \\
\wedge & (\mathit{patlen} \not< j) \\
\wedge & (\mathit{patlen} \in \mathbf{N}) \\
\wedge & (\mathit{strlen} \in \mathbf{N}) \\
\wedge & ((i - j) < \text{search}\,(\mathit{pat},\ \mathit{str},\ \mathit{patlen},\ \mathit{strlen},\ k)) \\
\wedge & (\mathit{strlen} \not< k) \\
\wedge & (i \not< j) \\
\wedge & (k \in \mathbf{N}) \\
\wedge & (k \neq \mathtt{0})) \\
\rightarrow & ((i + \text{delta1}\,(\mathit{pat},\ \text{elt1}\,(\mathit{str},\ i),\ \mathit{patlen})) \\
 & < \quad (\mathit{patlen} + \text{search}\,(\mathit{pat},\ \mathit{str},\ \mathit{patlen},\ \mathit{strlen},\ k)))
\end{array}
$$

THEOREM: outer-invrt1
$$
\begin{array}{ll}
(\text{zneqp}\,(\text{C\$2},\ \text{elt1}\,(\text{PAT\$0},\ \text{J\$2})) \wedge \text{PATH-HYPS}) & \\
\rightarrow & (((\text{I\$3} \in \mathbf{N}) \\
 & \wedge \quad ((\text{I\$3} \not< \text{PATLEN\$0}) \\
 & \qquad \wedge \quad ((\text{I\$3} \\
 & \qquad\quad < \quad (\text{PATLEN\$0} \\
 & \qquad\qquad + \quad \text{search}\,(\text{PAT\$0}, \\
 & \qquad\qquad\qquad\qquad \text{STR\$0}, \\
 & \qquad\qquad\qquad\qquad \text{PATLEN\$0}, \\
 & \qquad\qquad\qquad\qquad \text{STRLEN\$0}, \\
 & \qquad\qquad\qquad\qquad \mathtt{'1}))) \\
 & \qquad\quad \wedge \quad (((c \in \mathbf{N}) \\
 & \qquad\qquad\quad \wedge \quad ((c \neq \mathtt{'0}) \wedge (\text{ASIZE\&} \not< c))) \\
 & \qquad\qquad\quad \rightarrow \quad (\text{elt1}\,(\text{BLK-DELTA1\$3},\ c)
\end{array}
$$

$$
\begin{array}{ll}
 & \quad = \quad \text{delta1}\,(\text{PAT\$0},\ c,\ \text{PATLEN\$0}))))))\\
\wedge & \quad \text{lex}\,(\text{cons}\,((\texttt{'1} + (\text{STRLEN\$0} + \text{PATLEN\$0})) - \text{I\$3},\\
 & \qquad\quad \text{cons}\,(1 + \text{PATLEN\$0},\ \texttt{'nil})),\\
 & \qquad \text{cons}\,((\texttt{'1} + (\text{STRLEN\$0} + \text{PATLEN\$0})) - (\text{NEXTI\$1} - 1),\\
 & \qquad\quad \text{cons}\,(\text{J\$1},\ \texttt{'nil}))))
\end{array}
$$

```
#| (COMMENT INNER-INVRT T) |#
```

EVENT: Undo back through the event named 'logical-if-t'.

```
; (FORTRAN-COMMENT LOGICAL-IF-F)
```

AXIOM: logical-if-f
t

```
; (FORTRAN-COMMENT LOGICAL-IF-T)
```

AXIOM: logical-if-t
t

THEOREM: first-match-is-search
$$
\begin{array}{ll}
 & (\text{match}\,(\mathit{pat},\ \texttt{1},\ \mathit{patlen},\ \mathit{str},\ i,\ \mathit{strlen})\\
\wedge & \quad (\mathit{strlen} \in \mathbf{N})\\
\wedge & \quad (i \in \mathbf{N})\\
\wedge & \quad (\mathit{strlen} \not< i)\\
\wedge & \quad (k \in \mathbf{N})\\
\wedge & \quad (i \not< k)\\
\wedge & \quad (k \neq \texttt{0})\\
\wedge & \quad (\text{search}\,(\mathit{pat},\ \mathit{str},\ \mathit{patlen},\ \mathit{strlen},\ k) \not< i))\\
\rightarrow & \quad ((i = \text{search}\,(\mathit{pat},\ \mathit{str},\ \mathit{patlen},\ \mathit{strlen},\ k)) = \mathbf{t})
\end{array}
$$

AXIOM: assignment2
$$
\begin{array}{ll}
 & (\text{I\$3} = \text{I\$2})\\
\wedge & \quad ((\text{J\$3} = \text{J\$2})\\
 & \qquad \wedge \quad ((\text{C\$3} = \text{C\$2})\\
 & \qquad\quad \wedge \quad ((\text{NEXTI\$3} = \text{NEXTI\$2})\\
 & \qquad\qquad \wedge \quad ((\text{X\$3} = \text{I\$2})\\
 & \qquad\qquad\quad \wedge \quad (\text{BLK-DELTA1\$3} = \text{BLK-DELTA1\$2})))))
\end{array}
$$

THEOREM: output
$$
\begin{array}{ll}
 & (\text{zeqp}\,(\text{J\$2},\ \texttt{'1}) \wedge ((\neg\ \text{zneqp}\,(\text{C\$2},\ \text{elt1}\,(\text{PAT\$0},\ \text{J\$2}))) \wedge \text{PATH-HYPS}))\\
\rightarrow & \quad (\text{X\$3} = \text{search}\,(\text{PAT\$0},\ \text{STR\$0},\ \text{PATLEN\$0},\ \text{STRLEN\$0},\ \texttt{'1}))
\end{array}
$$

```
#| (COMMENT INNER-INVRT F T) |#
```

EVENT: Undo back through the event named 'logical-if-t'.

```
; (FORTRAN-COMMENT LOGICAL-IF-F1)
```

AXIOM: logical-if-f1
t

THEOREM: input-cond-of-zdifference
((¬ zeqp (J$2, '1)) ∧ ((¬ zneqp (C$2, elt1 (PAT$0, J$2))) ∧ PATH-HYPS))
→ expressible-znumberp (zdifference (J$2, '1))

```
#| (COMMENT INNER-INVRT F F) |#
```

AXIOM: assignment2
(I$3 = I$2)
∧ ((J$3 = zdifference (J$2, '1))
∧ ((C$3 = C$2)
∧ ((NEXTI$3 = NEXTI$2)
∧ ((X$3 = X$2)
∧ (BLK-DELTA1$3 = BLK-DELTA1$2)))))

THEOREM: input-cond-of-zdifference1
((¬ zeqp (J$2, '1)) ∧ ((¬ zneqp (C$2, elt1 (PAT$0, J$2))) ∧ PATH-HYPS))
→ expressible-znumberp (zdifference (I$3, '1))

```
#| (COMMENT INNER-INVRT F F) |#
```

AXIOM: assignment3
(I$4 = zdifference (I$3, '1))
∧ ((J$4 = J$3)
∧ ((C$4 = C$3)
∧ ((NEXTI$4 = NEXTI$3)
∧ ((X$4 = X$3)
∧ (BLK-DELTA1$4 = BLK-DELTA1$3)))))

THEOREM: open-up-difference
(x − 1) = (x − 1)

```
(PROVE-LEMMA INNER-INVRT1 NIL
 (IMPLIES
  (AND (NOT (ZEQP (J$2) '1))
```

```
     (AND (NOT (ZNEQP (C$2) (ELT1 (PAT$0) (J$2))))
          (PATH-HYPS)))
(AND
 (AND
  (NOT (LESSP (NEXTI$4) (ADD1 (PATLEN$0))))
  (AND
   (LESSP (NEXTI$4)
          (ADD1 (PLUS (PATLEN$0)
                      (SEARCH (PAT$0)
                              (STR$0)
                              (PATLEN$0)
                              (STRLEN$0)
                              '1))))
   (AND
    (IMPLIES (AND (NUMBERP C)
                  (AND (NOT (EQUAL C '0))
                       (NOT (LESSP (ASIZE&) C))))
             (EQUAL (ELT1 (BLK-DELTA1$4) C)
                    (DELTA1 (PAT$0) C (PATLEN$0))))
    (AND
     (NUMBERP (I$4))
     (AND
      (NOT (EQUAL (I$4) '0))
      (AND
       (NUMBERP (J$4))
       (AND
        (NOT (EQUAL (J$4) '0))
        (AND
         (NUMBERP (NEXTI$4))
         (AND
          (NOT (LESSP (PATLEN$0) (J$4)))
          (AND
           (NOT (LESSP (STRLEN$0) (I$4)))
           (AND
            (EQUAL (NEXTI$4)
                   (PLUS (ADD1 (PATLEN$0))
                         (DIFFERENCE (I$4) (J$4))))
            (AND
             (NOT (LESSP (ADD1 (STRLEN$0)) (NEXTI$4)))
             (AND
                (NOT (LESSP (I$4) (J$4)))
                (AND (MATCH (PAT$0)
                            (ADD1 (J$4))
                            (PATLEN$0)
```

```
                                (STR$0)
                                (ADD1 (I$4))
                                (STRLEN$0))
                          (AND (NUMBERP (ELT1 (BLK-DELTA1$4)
                                              (ELT1 (STR$0) (I$4))))
                               (AND (NUMBERP (ELT1 (STR$0) (I$4)))
                                    (NUMBERP (ELT1 (PAT$0) (J$4))))))))))))))))))))
   (LEX (CONS (DIFFERENCE (PLUS '1
                                (PLUS (STRLEN$0) (PATLEN$0)))
                          (SUB1 (NEXTI$4)))
              (CONS (J$4) 'NIL))
        (CONS (DIFFERENCE (PLUS '1
                                (PLUS (STRLEN$0) (PATLEN$0)))
                          (SUB1 (NEXTI$1)))
              (CONS (J$1) 'NIL))))))

#| (COMMENT INNER-INVRT F F) |#
```

EVENT: Undo back through the event named 'paths-from-inner-invrt'.

EVENT: Undo back through the event named 'fortran'.

```
#|


The correctness of the program depends upon the following events:

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Definition.
        (DELTA1 A C MAXI)
           =
        (IF (ZEROP MAXI)
            0
            (IF (EQUAL C (ELT1 A MAXI))
                0
                (ADD1 (DELTA1 A C (SUB1 MAXI)))))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
```

```
        Definition.
        (STRINGP A I SIZE)
           =
        (IF (ZEROP I)
            T
            (AND (NUMBERP (ELT1 A I))
                 (NOT (EQUAL (ELT1 A I) 0))
                 (NOT (LESSP SIZE (ELT1 A I)))
                 (STRINGP A (SUB1 I) SIZE)))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        (FORTRAN-COMMENT USEFUL-LEMMAS)
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Definition.
        (MATCH PAT J PATLEN STR I STRLEN)
           =
        (IF (LESSP PATLEN J)
            T
            (IF (LESSP STRLEN I)
                F
                (AND (EQUAL (ELT1 PAT J) (ELT1 STR I))
                     (MATCH PAT
                            (ADD1 J)
                            PATLEN STR
                            (ADD1 I)
                            STRLEN))))
        Hint:  ((LESSP (DIFFERENCE (ADD1 PATLEN) J))
                (LESSP (DIFFERENCE (ADD1 STRLEN) I)))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Definition.
        (SEARCH PAT STR PATLEN STRLEN I)
           =
        (IF (LESSP STRLEN I)
```

```
            (ADD1 STRLEN)
            (IF (MATCH PAT 1 PATLEN STR I STRLEN)
                I
                (SEARCH PAT STR PATLEN STRLEN
                        (ADD1 I))))
        Hint:  Consider the well-founded relation LESSP
               and the measure (DIFFERENCE (ADD1 STRLEN) I)
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        (FORTRAN-COMMENT USEFUL-LEMMAS-ABOUT-SEARCHING)
@END(VERBATIM)
@END(GROUP)
```

```
Specification for routine SETUP

The input assertion:
        (AND (STRINGP (A STATE)
                      (MAXI STATE)
                      (ASIZE&))
             (NOT (EQUAL (MAXI STATE) 0))
             (NUMBERP (MAXI STATE))
             (LESSP (ADD1 (ASIZE&))
                    (LEAST-INEXPRESSIBLE-POSITIVE-INTEGER))
             (LESSP (ADD1 (MAXI STATE))
                    (LEAST-INEXPRESSIBLE-POSITIVE-INTEGER)))


The output assertion:
        (IMPLIES (AND (NUMBERP C)
                      (NOT (EQUAL C 0))
                      (NOT (LESSP (ASIZE&) C)))
                 (EQUAL (ELT1 (BLK-DELTA1 NEWSTATE) C)
                        (DELTA1 (A STATE) C (MAXI STATE))))
```

```
Specification for routine FSRCH

The input assertion:
        (AND (LESSP (ADD1 (ASIZE&))
                    (LEAST-INEXPRESSIBLE-POSITIVE-INTEGER))
             (STRINGP (PAT STATE)
                      (PATLEN STATE)
                      (ASIZE&))
             (NUMBERP (PATLEN STATE))
             (LESSP 0 (PATLEN STATE))
             (STRINGP (STR STATE)
                      (STRLEN STATE)
                      (ASIZE&))
             (NUMBERP (STRLEN STATE))
             (LESSP 0 (STRLEN STATE))
             (LESSP (PLUS (PATLEN STATE) (STRLEN STATE))
                    (LEAST-INEXPRESSIBLE-POSITIVE-INTEGER)))


The output assertion:
        (EQUAL (X NEWSTATE)
               (SEARCH (PAT STATE)
                       (STR STATE)
                       (PATLEN STATE)
                       (STRLEN STATE)
                       1))
```

```
      INTEGER DELTA1
      DIMENSION DELTA1(ASIZE&)
      COMMON /BLK/DELTA1
      END
      SUBROUTINE SETUP(A, MAXI)
      INTEGER DELTA1
      INTEGER A
      INTEGER MAXI
      INTEGER I
      INTEGER C
      DIMENSION DELTA1(ASIZE&)
      DIMENSION A(MAXI)
      COMMON /BLK/DELTA1
      DO 50 I = 1, ASIZE&
C     DOJUNK PASS1
      DELTA1(I) = MAXI
50    CONTINUE
      DO 100 I = 1, MAXI
C     DOJUNK PASS2
      C = A(I)
      DELTA1(C) = (MAXI - I)
100   CONTINUE
      RETURN
      END
      SUBROUTINE FSRCH(PAT, STR, PATLEN, STRLEN, X)
      INTEGER DELTA1
      INTEGER MAX0
      INTEGER PATLEN
      INTEGER STRLEN
      INTEGER PAT
      INTEGER STR
      INTEGER I
      INTEGER J
      INTEGER C
      INTEGER NEXTI
      INTEGER X
      DIMENSION DELTA1(ASIZE&)
      DIMENSION PAT(PATLEN)
      DIMENSION STR(STRLEN)
      COMMON /BLK/DELTA1
      CALL SETUP(PAT, PATLEN)
      I = PATLEN
200   CONTINUE
```

```
C     XXX INNER-TO-OUTER-HINTS
      CONTINUE
C     ASSERTION OUTER-INVRT
      IF ((I .GT. STRLEN)) GOTO 500
      J = PATLEN
      NEXTI = (1 + I)
300   CONTINUE
C     ASSERTION INNER-INVRT
      C = STR(I)
      IF ((C .NE. PAT(J))) GOTO 400
      IF ((J .EQ. 1)) GOTO 600
      J = (J - 1)
      I = (I - 1)
C     XXX INNER-TO-INNER-HINTS
      GOTO 300
400   I = MAX0((I + DELTA1(C)), NEXTI)
      GOTO 200
500   X = (STRLEN + 1)
      RETURN
600   CONTINUE
C     XXX INNER-TO-EXIT-HINTS
      X = I
      RETURN
      END
```

```
The XXX at USEFUL-LEMMAS.
@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  STRINGP-IS-A-UNIV-QUANTIFIER (rewrite):
        (IMPLIES (AND (STRINGP A I (ASIZE&))
                      (NOT (EQUAL J 0))
                      (NUMBERP J)
                      (NOT (LESSP I J)))
                 (AND (NUMBERP (ELT1 A J))
                      (NOT (LESSP (ASIZE&) (ELT1 A J)))
                      (NOT (EQUAL (ELT1 A J) 0))))
@END(VERBATIM)
@END(GROUP)
```

```
The XXX at USEFUL-LEMMAS-ABOUT-SEARCHING.
@BEGIN(GROUP)
```

```
@BEGIN(VERBATIM)
        Theorem.  SEARCH-NON-ZERO (rewrite):
        (IMPLIES (NOT (ZEROP I))
                 (LESSP 0
                        (SEARCH PAT STR PATLEN STRLEN I)))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  MATCH-AT-PATLEN (rewrite):
        (IMPLIES (LESSP PATLEN J)
                 (MATCH PAT J PATLEN STR I STRLEN))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  MATCH-AT-STRLEN (rewrite):
        (IMPLIES (LESSP STRLEN I)
                 (EQUAL (MATCH PAT J PATLEN STR I STRLEN)
                        (LESSP PATLEN J)))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  MATCH-NEEDS-PATLEN-CHARS (rewrite):
        (IMPLIES (AND (NUMBERP PATLEN)
                      (NUMBERP STRLEN)
                      (NOT (LESSP PATLEN J))
                      (NOT (LESSP STRLEN I))
                      (LESSP (PLUS J STRLEN)
                             (PLUS I PATLEN)))
                 (NOT (MATCH PAT J PATLEN STR I STRLEN)))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  SEARCH-BOUNDARY (rewrite):
        (IMPLIES
               (AND (NOT (ZEROP PATLEN))
                    (NUMBERP STRLEN)
```

```
                    (NOT (LESSP STRLEN I))
                    (LESSP (ADD1 STRLEN)
                           (PLUS PATLEN
                                 (SEARCH PAT STR PATLEN STRLEN I))))
               (EQUAL (SEARCH PAT STR PATLEN STRLEN I)
                      (ADD1 STRLEN)))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  DELTA1-LESSEQP-PATLEN (rewrite):
        (NOT (LESSP PATLEN
                    (DELTA1 PAT CHAR PATLEN)))
@END(VERBATIM)
@END(GROUP)
```

```
Hints for routine SETUP



The input clock:
        (LIST (PLUS 2 (ASIZE&) (MAXI (START))))

The DO junk named PASS1:
        ((TEST PASS1-INVRT))

The invariant and clock named PASS1-INVRT.

        (AND (NUMBERP (I STATE))
             (NOT (EQUAL (I STATE) 0))
             (NOT (LESSP (ADD1 (ASIZE&)) (I STATE)))
             (IMPLIES (AND (NUMBERP J)
                           (NOT (EQUAL J 0))
                           (LESSP J (I STATE)))
                      (EQUAL (ELT1 (BLK-DELTA1 STATE) J)
                             (MAXI (START)))))

        (LIST (DIFFERENCE (PLUS 2 (ASIZE&) (MAXI (START)))
                          (I STATE)))

The DO junk named PASS2:
        ((TEST PASS2-INVRT)
         (JUMP PASS2-TO-PASS2-HINT))

The invariant and clock named PASS2-INVRT.

        (AND (NUMBERP (I STATE))
             (NOT (EQUAL (I STATE) 0))
             (NOT (LESSP (ADD1 (MAXI (START)))
                         (I STATE)))
             (IMPLIES (AND (NOT (ZEROP J))
                           (NOT (LESSP (ASIZE&) J)))
                      (EQUAL (ELT1 (BLK-DELTA1 STATE) J)
                             (PLUS (DELTA1 (A (START))
                                           J
                                           (SUB1 (I STATE)))
                                   (DIFFERENCE (MAXI (START))
                                               (SUB1 (I STATE)))))))
```

```
        (LIST (DIFFERENCE (ADD1 (MAXI (START)))
                          (I STATE)))

The XXX named PASS2-TO-PASS2-HINT:
@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  DELTA1-SKIP (rewrite):
        (IMPLIES (AND (NOT (EQUAL J (ELT1 PAT I)))
                      (NOT (ZEROP I)))
                 (EQUAL (DELTA1 PAT J I)
                        (ADD1 (DELTA1 PAT J (SUB1 I)))))
@END(VERBATIM)
@END(GROUP)
```

```
Hints for routine FSRCH



The input clock:
        (LIST (PLUS 1
                    (STRLEN (START))
                    (PATLEN (START)))
              (ADD1 (PATLEN (START))))

The XXX named INNER-TO-OUTER-HINTS:
        Path encryption:
        ((INNER-INVRT T))
@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  DELTA1-PLUS-J-LESSEQP-PATLEN (rewrite):
        (IMPLIES
              (AND (NUMBERP J)
                   (NOT (EQUAL J 0))
                   (NOT (LESSP PATLEN J)))
              (NOT (LESSP PATLEN
                          (PLUS J
                                (DELTA1 PAT (ELT1 PAT J) PATLEN)))))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  DIFFERENCE-PLUS-ID (rewrite):
        (EQUAL (DIFFERENCE (PLUS K J) J)
               (FIX K))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  MATCH-IMPLIES-EQ-CHARS (rewrite):
        (IMPLIES (AND (MATCH PAT J PATLEN STR K STRLEN)
                      (NUMBERP K)
                      (NUMBERP J)
                      (NUMBERP PATLEN)
                      (NUMBERP STRLEN)
                      (NOT (LESSP STRLEN K))
```

```
                    (NOT (LESSP PATLEN J))
                    (NOT (EQUAL K 0))
                    (NOT (EQUAL J 0))
                    (NUMBERP L)
                    (NOT (LESSP L K))
                    (NOT (LESSP (PLUS K PATLEN) (PLUS J L))))
               (EQUAL (ELT1 STR L)
                      (ELT1 PAT (DIFFERENCE (PLUS J L) K))))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  NEQ-CHARS-IN-REGION-MEANS-SEARCH-SKIPS (rewrite):
        (IMPLIES
            (AND (NOT (EQUAL (ELT1 STR I) (ELT1 PAT J)))
                 (NUMBERP I)
                 (NOT (EQUAL I 0))
                 (NUMBERP STRLEN)
                 (NOT (LESSP STRLEN I))
                 (NUMBERP J)
                 (NOT (EQUAL J 0))
                 (NUMBERP PATLEN)
                 (NOT (LESSP PATLEN J))
                 (NOT (LESSP I J))
                 (NUMBERP K)
                 (NOT (EQUAL K 0))
                 (NOT (LESSP STRLEN K))
                 (NOT (LESSP (SUB1 (SEARCH PAT STR PATLEN STRLEN K))
                             (DIFFERENCE I J))))
            (LESSP (DIFFERENCE I J)
                   (SUB1 (SEARCH PAT STR PATLEN STRLEN K))))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  DELTA1-DOESNT-GO-BEYOND-END-OF-MATCH (rewrite):
        (IMPLIES
               (AND (MATCH PAT 1 PATLEN STR K STRLEN)
                    (NUMBERP I)
                    (NOT (LESSP STRLEN I))
                    (NUMBERP PATLEN)
                    (NUMBERP STRLEN)
```

```
                    (NUMBERP K)
                    (NOT (EQUAL K 0))
                    (LESSP I (PLUS K PATLEN)))
               (EQUAL (LESSP (PLUS I
                                   (DELTA1 PAT (ELT1 STR I) PATLEN))
                             (PLUS K PATLEN))
                      (OR (LESSP I K)
                          (EQUAL I K)
                          (LESSP K I))))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Enable MATCH-IMPLIES-EQ-CHARS.
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  DELTA1-DOESNT-GO-BEYOND-SEARCH (rewrite):
        (IMPLIES (AND (NOT (EQUAL (ELT1 STR I) (ELT1 PAT J)))
                      (NUMBERP I)
                      (NOT (LESSP STRLEN I))
                      (NUMBERP J)
                      (NOT (LESSP PATLEN J))
                      (NUMBERP PATLEN)
                      (NUMBERP STRLEN)
                      (LESSP (DIFFERENCE I J)
                             (SEARCH PAT STR PATLEN STRLEN K))
                      (NOT (LESSP STRLEN K))
                      (NOT (LESSP I J))
                      (NUMBERP K)
                      (NOT (EQUAL K 0)))
                 (LESSP (PLUS I
                              (DELTA1 PAT (ELT1 STR I) PATLEN))
                        (PLUS PATLEN
                              (SEARCH PAT STR PATLEN STRLEN K))))
@END(VERBATIM)
@END(GROUP)
```

The invariant and clock named OUTER-INVRT.

```
(AND (NUMBERP (I STATE))
     (NOT (LESSP (I STATE) (PATLEN (START))))
     (LESSP (I STATE)
            (PLUS (PATLEN (START))
                  (SEARCH (PAT (START))
                          (STR (START))
                          (PATLEN (START))
                          (STRLEN (START))
                          1)))
     (IMPLIES (AND (NUMBERP C)
                   (NOT (EQUAL C 0))
                   (NOT (LESSP (ASIZE&) C)))
              (EQUAL (ELT1 (BLK-DELTA1 STATE) C)
                     (DELTA1 (PAT (START))
                             C
                             (PATLEN (START))))))

(LIST (DIFFERENCE (PLUS 1
                        (STRLEN (START))
                        (PATLEN (START)))
                  (I STATE))
      (ADD1 (PATLEN (START))))
```

The invariant and clock named INNER-INVRT.

```
(AND (NOT (LESSP (NEXTI STATE)
                 (ADD1 (PATLEN (START)))))
     (LESSP (NEXTI STATE)
            (ADD1 (PLUS (PATLEN (START))
                        (SEARCH (PAT (START))
                                (STR (START))
                                (PATLEN (START))
                                (STRLEN (START))
                                1))))
     (IMPLIES (AND (NUMBERP C)
                   (NOT (EQUAL C 0))
                   (NOT (LESSP (ASIZE&) C)))
              (EQUAL (ELT1 (BLK-DELTA1 STATE) C)
                     (DELTA1 (PAT (START))
                             C
                             (PATLEN (START)))))
     (NUMBERP (I STATE))
     (NOT (EQUAL (I STATE) 0))
```

```
              (NUMBERP (J STATE))
              (NOT (EQUAL (J STATE) 0))
              (NUMBERP (NEXTI STATE))
              (NOT (LESSP (PATLEN (START)) (J STATE)))
              (NOT (LESSP (STRLEN (START)) (I STATE)))
              (EQUAL (NEXTI STATE)
                     (PLUS (ADD1 (PATLEN (START)))
                           (DIFFERENCE (I STATE) (J STATE))))
              (NOT (LESSP (ADD1 (STRLEN (START)))
                          (NEXTI STATE)))
              (NOT (LESSP (I STATE) (J STATE)))
              (MATCH (PAT (START))
                     (ADD1 (J STATE))
                     (PATLEN (START))
                     (STR (START))
                     (ADD1 (I STATE))
                     (STRLEN (START)))
              (NUMBERP (ELT1 (BLK-DELTA1 STATE)
                             (ELT1 (STR (START)) (I STATE))))
              (NUMBERP (ELT1 (STR (START)) (I STATE)))
              (NUMBERP (ELT1 (PAT (START)) (J STATE))))

         (LIST (DIFFERENCE (PLUS 1
                                 (STRLEN (START))
                                 (PATLEN (START)))
                           (SUB1 (NEXTI STATE)))
               (J STATE))

The XXX named INNER-TO-INNER-HINTS:
@BEGIN(GROUP)
@BEGIN(VERBATIM)
         Theorem.  OPEN-UP-DIFFERENCE (rewrite):
         (EQUAL (DIFFERENCE X 1) (SUB1 X))
@END(VERBATIM)
@END(GROUP)



The XXX named INNER-TO-EXIT-HINTS:
@BEGIN(GROUP)
@BEGIN(VERBATIM)
         Theorem.  FIRST-MATCH-IS-SEARCH (rewrite):
         (IMPLIES (AND (MATCH PAT 1 PATLEN STR I STRLEN)
                       (NUMBERP STRLEN)
```

```
                    (NUMBERP I)
                    (NOT (LESSP STRLEN I))
                    (NUMBERP K)
                    (NOT (LESSP I K))
                    (NOT (EQUAL K 0))
                    (NOT (LESSP (SEARCH PAT STR PATLEN STRLEN K)
                                I)))
               (EQUAL (EQUAL I
                             (SEARCH PAT STR PATLEN STRLEN K))
                      T))
@END(VERBATIM)
@END(GROUP)

|#
```

Index