```
#|

 Copyright (C) 1994 by Computational Logic, Inc.  All Rights Reserved.

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

 NO WARRANTY

 Computational Logic, Inc. PROVIDES 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 Computational Logic, Inc. 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.
|#
```

```
;; Matt Kaufmann

;; Here are some games with "partial functions".....
```

EVENT: Start with the initial **nqthm** theory.

DEFINITION:
s-plus (x, y)
= **if** x
then if y **then** $x + y$
else f endif
else f endif

EVENT: Introduce the function symbol *apply* of 2 arguments.

```
;; Example 1:  a simple total reflexive function that's actually
;; the identity function on natural numbers.
```

```
;; dcls, add-axioms, and rewrite rules for g-cost and g (6 events)
```

EVENT: Introduce the function symbol *g-cost* of one argument.

EVENT: Introduce the function symbol *g* of one argument.

AXIOM: g-defn

$$
\begin{aligned}
& \mathrm{g}\,(x) \\
= \quad & \textbf{if } \text{g-cost}\,(x) \\
& \textbf{then if } x \simeq \mathtt{0} \ \textbf{ then } \mathtt{0} \\
& \qquad\quad \textbf{else } 1 + \mathrm{g}\,(\mathrm{g}\,(x - 1)) \textbf{ endif} \\
& \textbf{else } \text{apply}\,(\mathtt{'g}, \text{list}\,(x)) \textbf{ endif}
\end{aligned}
$$

AXIOM: g-cost-defn

$$
\begin{aligned}
& \text{g-cost}\,(x) \\
= \quad & \textbf{if } x \simeq \mathtt{0} \ \textbf{ then } \mathtt{1} \\
& \textbf{else } \text{s-plus}\,(\mathtt{1}, \text{s-plus}\,(\text{g-cost}\,(x - 1), \text{g-cost}\,(\mathrm{g}\,(x - 1)))) \textbf{ endif}
\end{aligned}
$$

THEOREM: g-cost-opener

$$
\begin{aligned}
& ((x \simeq \mathtt{0}) \rightarrow (\text{g-cost}\,(x) = \mathtt{1})) \\
\wedge \quad & ((x \not\simeq \mathtt{0}) \\
& \quad \rightarrow \quad (\text{g-cost}\,(x) \\
& \qquad\qquad = \quad \text{s-plus}\,(\mathtt{1}, \text{s-plus}\,(\text{g-cost}\,(x - 1), \text{g-cost}\,(\mathrm{g}\,(x - 1))))))
\end{aligned}
$$

THEOREM: g-opener

$$
\begin{aligned}
& ((x \simeq \mathtt{0}) \\
& \rightarrow \quad (\mathrm{g}\,(x) \\
& \qquad = \quad \textbf{if } \text{g-cost}\,(x) \ \textbf{ then } \mathtt{0} \\
& \qquad\qquad \textbf{else } \text{apply}\,(\mathtt{'g}, \text{list}\,(x)) \textbf{ endif})) \\
\wedge \quad & ((x \not\simeq \mathtt{0}) \\
& \quad \rightarrow \quad (\mathrm{g}\,(x) \\
& \qquad\qquad = \quad \textbf{if } \text{g-cost}\,(x) \ \textbf{ then } 1 + \mathrm{g}\,(\mathrm{g}\,(x - 1)) \\
& \qquad\qquad\qquad \textbf{else } \text{apply}\,(\mathtt{'g}, \text{list}\,(x)) \textbf{ endif}))
\end{aligned}
$$

THEOREM: g-theorem

$$\text{g-cost}\,(x) \wedge (\mathrm{g}\,(x) = \text{fix}\,(x))$$

```
;; Example 2:  Silly factorial
```

DEFINITION:

$$
\begin{aligned}
& \text{isub1}\,(x) \\
= \quad & \textbf{if } \text{negativep}\,(x) \ \textbf{ then } -\,(1 + \text{negative-guts}\,(x)) \\
& \textbf{else } x - 1 \textbf{ endif}
\end{aligned}
$$

```
;; dcls, add-axioms, and rewrite rules for fact-cost and fact (5 events).
;; Note that we don't try to make a rewrite rule for the nonterminating
;; case.  Also, since our function isn't reflexive and we are only interested
;; in termination, we only bother to prove a rewrite rule for opening up
;; fact-cost, not one for opening up fact.
```

EVENT: Introduce the function symbol *fact-cost* of one argument.

EVENT: Introduce the function symbol *fact* of one argument.

AXIOM: fact-defn

$$\begin{aligned} & \text{fact}\,(x) \\ = \quad & \textbf{if } \text{fact-cost}\,(x) \\ & \textbf{then if } x = \mathtt{0} \textbf{ then } \mathtt{1} \\ & \qquad \textbf{else } \text{fact}\,(\text{isub1}\,(x)) \textbf{ endif} \\ & \textbf{else } \text{apply}\,(\texttt{'fact}, \text{list}\,(x)) \textbf{ endif} \end{aligned}$$

AXIOM: fact-cost-defn

$$\begin{aligned} & \text{fact-cost}\,(x) \\ = \quad & \textbf{if } x = \mathtt{0} \textbf{ then } \mathtt{1} \\ & \textbf{else } \text{s-plus}\,(\mathtt{1}, \text{fact-cost}\,(\text{isub1}\,(x))) \textbf{ endif} \end{aligned}$$

THEOREM: fact-cost-opener-numberp

$$\begin{aligned} & ((x = \mathtt{0}) \rightarrow (\text{fact-cost}\,(x) = \mathtt{1})) \\ \wedge \quad & ((x \not\simeq \mathtt{0}) \rightarrow (\text{fact-cost}\,(x) = \text{s-plus}\,(\mathtt{1}, \text{fact-cost}\,(x - 1)))) \end{aligned}$$

```
;; Now let's first note when fact IS defined.
```

THEOREM: fact-defined-numberp

$$(x \in \mathbf{N}) \rightarrow \text{fact-cost}\,(x)$$

THEOREM: fact-defined-other

$$((x \simeq \mathtt{0}) \wedge (\neg\ \text{negativep}\,(x))) \rightarrow \text{fact-cost}\,(x)$$

```
;; Next, let's show that fact is undefined on the negatives by
;; showing that the cost is arbitrarily high (the usual trick
;; used for analogous v&c$ proofs).
```

DEFINITION:

$$\begin{aligned} & \text{fact-undefined-ind}\,(x, n) \\ = \quad & \textbf{if } n \simeq \mathtt{0} \textbf{ then } \mathbf{t} \\ & \textbf{else } \text{fact-undefined-ind}\,(1 + x, n - 1) \textbf{ endif} \end{aligned}$$

THEOREM: fact-undefined-numberp-lemma-inductive-step

$$((n \not\simeq 0) \wedge \text{fact-cost}(-\ x) \wedge (\text{fact-cost}(-\ (1+x)) \rightarrow ((n-1) \le \text{fact-cost}(-\ (1+x))))) \rightarrow ((\text{fact-cost}(-\ x) < n) = \mathbf{f})$$

THEOREM: fact-undefined-negativep-lemma

$\text{fact-cost}(-\ x) \rightarrow ((\text{fact-cost}(-\ x) < n) = \mathbf{f})$

THEOREM: fact-undefined-negativep

$\text{negativep}(z) \rightarrow (\text{fact-cost}(z) = \mathbf{f})$

```
;; finally, we put this all together
```

THEOREM: fact-domain

$\text{fact-cost}(x) \leftrightarrow (\neg\ \text{negativep}(x))$

```
;; Example 3:  triple reverse

;; First, ordinary reverse, and proper list recognizer
```

DEFINITION:

rev (x)

= **if** listp (x) **then** append (rev (cdr (x)), list (car (x)))
else nil endif

DEFINITION:

plistp (x)

= **if** listp (x) **then** plistp (cdr (x))
else x = **nil endif**

DEFINITION:

length (x)

= **if** listp (x) **then** $1 +$ length (cdr (x))
else 0 **endif**

```
;; dcls, add-axioms, and rewrite rules for rev3-cost and rev (6 events)
```

EVENT: Introduce the function symbol *rev3-cost* of one argument.

EVENT: Introduce the function symbol *rev3* of one argument.

AXIOM: rev3-defn

$$
\begin{array}{l}
\text{rev3}(x) \\
= \quad \textbf{if } \text{rev3-cost}(x) \\
\qquad \textbf{then if } \text{listp}(\text{cdr}(x)) \\
\qquad\qquad \textbf{then } \text{cons}(\text{car}(\text{rev3}(\text{cdr}(x))), \\
\qquad\qquad\qquad\qquad \text{rev3}(\text{cons}(\text{car}(x),\ \text{rev3}(\text{cdr}(\text{rev3}(\text{cdr}(x))))))) \\
\qquad\qquad \textbf{else } x \textbf{ endif} \\
\qquad \textbf{else } \text{apply}(\texttt{'rev3},\ \text{list}(x)) \textbf{ endif}
\end{array}
$$

AXIOM: rev3-cost-defn

$$
\begin{array}{l}
\text{rev3-cost}(x) \\
= \quad \textbf{if } \text{listp}(\text{cdr}(x)) \\
\qquad \textbf{then } \text{s-plus}(\texttt{1}, \\
\qquad\qquad\qquad \text{s-plus}(\text{rev3-cost}(\text{cdr}(x)), \\
\qquad\qquad\qquad\qquad \text{s-plus}(\text{rev3-cost}(\text{cdr}(\text{rev3}(\text{cdr}(x)))), \\
\qquad\qquad\qquad\qquad\qquad \text{rev3-cost}(\text{cons}(\text{car}(x), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{rev3}(\text{cdr}(\text{rev3}(\text{cdr}(x))))))))) \\
\qquad \textbf{else } \texttt{1} \textbf{ endif}
\end{array}
$$

THEOREM: rev3-cost-opener

$$
\begin{array}{l}
(\text{listp}(\text{cdr}(x)) \\
\quad \to \quad (\text{rev3-cost}(x) \\
\qquad\qquad = \quad \text{s-plus}(\texttt{1}, \\
\qquad\qquad\qquad\qquad \text{s-plus}(\text{rev3-cost}(\text{cdr}(x)), \\
\qquad\qquad\qquad\qquad\qquad \text{s-plus}(\text{rev3-cost}(\text{cdr}(\text{rev3}(\text{cdr}(x)))), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{rev3-cost}(\text{cons}(\text{car}(x), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{rev3}(\text{cdr}(\text{rev3}(\text{cdr}(x))))))))))) \\
\land \quad ((\text{cdr}(x) \simeq \textbf{nil}) \to (\text{rev3-cost}(x) = \texttt{1}))
\end{array}
$$

THEOREM: rev3-defn-opener

$$
\begin{array}{l}
(\text{listp}(\text{cdr}(x)) \\
\quad \to \quad (\text{rev3}(x) \\
\qquad\qquad = \quad \textbf{if } \text{rev3-cost}(x) \\
\qquad\qquad\qquad \textbf{then if } \text{listp}(\text{cdr}(x)) \\
\qquad\qquad\qquad\qquad \textbf{then } \text{cons}(\text{car}(\text{rev3}(\text{cdr}(x))), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{rev3}(\text{cons}(\text{car}(x),\ \text{rev3}(\text{cdr}(\text{rev3}(\text{cdr}(x))))))) \\
\qquad\qquad\qquad\qquad \textbf{else } x \textbf{ endif} \\
\qquad\qquad\qquad \textbf{else } \text{apply}(\texttt{'rev3},\ \text{list}(x)) \textbf{ endif})) \\
\land \quad ((\text{cdr}(x) \simeq \textbf{nil}) \to (\text{rev3}(x) = x))
\end{array}
$$

DEFINITION:

$$
\begin{array}{l}
\text{rev3-induction}(x,\ n) \\
= \quad \textbf{if } (n \simeq \texttt{0}) \lor ((n - 1) \simeq \texttt{0}) \ \textbf{ then t} \\
\qquad \textbf{else } \text{rev3-induction}(\text{cdr}(x),\ n - 1) \\
\qquad\qquad \land \quad \text{rev3-induction}(\text{cdr}(\text{rev3}(\text{cdr}(x))),\ (n - 1) - 1)
\end{array}
$$

$\wedge \quad$ rev3-induction (cons (car (x), rev3 (cdr (rev3 (cdr (x))))), $n - 1$) **endif**

THEOREM: length-0
$((\text{length}(x) = 0) = (\neg\ \text{listp}(x)))$
$\wedge \quad ((0 = \text{length}(x)) = (\neg\ \text{listp}(x)))$

THEOREM: rev3-length-and-definedness-lemma
$(\text{length}(x) = n) \rightarrow (\text{rev3-cost}(x) \wedge (\text{length}(\text{rev3}(x)) = n))$

THEOREM: rev3-defined
rev3-cost (x)

```
;; Now, just for fun, we'll show in the rest of these "rev" events
;; that rev3 is rev.  Note that we've already shown that rev3 is
;; "total" in the event just above.
```

EVENT: Disable rev3-cost-opener.

THEOREM: app-assoc
$\text{append}(\text{append}(x, y), z) = \text{append}(x, \text{append}(y, z))$

THEOREM: rev-rev
$\text{plistp}(x) \rightarrow (\text{rev}(\text{rev}(x)) = x)$

THEOREM: plistp-rev
plistp (rev (x))

THEOREM: plistp-append
$\text{plistp}(\text{append}(x, y)) = \text{plistp}(y)$

THEOREM: plistp-cdr
$(\text{plistp}(x) \wedge \text{listp}(x)) \rightarrow \text{plistp}(\text{cdr}(x))$

THEOREM: listp-append
$\text{listp}(\text{append}(x, y)) = (\text{listp}(x) \vee \text{listp}(y))$

THEOREM: rev-prop
plistp (x)
$\rightarrow$ (rev (x)
= **if** listp (cdr (x))
then cons (car (rev (cdr (x))),
rev (cons (car (x), rev (cdr (rev (cdr (x)))))))
else x **endif**)

THEOREM: rev-prop-rewrite

$$\begin{array}{l} \text{plistp}\,(x) \\ \rightarrow \quad ((\text{listp}\,(\text{cdr}\,(x)) \\ \qquad \rightarrow \quad (\text{rev}\,(x) \\ \qquad\qquad = \quad \text{cons}\,(\text{car}\,(\text{rev}\,(\text{cdr}\,(x))), \\ \qquad\qquad\qquad\qquad \text{rev}\,(\text{cons}\,(\text{car}\,(x),\ \text{rev}\,(\text{cdr}\,(\text{rev}\,(\text{cdr}\,(x))))))))) \\ \qquad \wedge \quad ((\text{cdr}\,(x) \simeq \mathbf{nil}) \rightarrow (\text{rev}\,(x) = x))) \end{array}$$

EVENT: Disable rev.

THEOREM: listp-rev

$\text{listp}\,(\text{rev}\,(x)) = \text{listp}\,(x)$

THEOREM: length-rev3

$\text{length}\,(\text{rev3}\,(x)) = \text{length}\,(x)$

THEOREM: rev3-nil

$(\text{rev3}\,(x) = \mathbf{nil}) = (x = \mathbf{nil})$

THEOREM: length-cdr-rev3

$\text{listp}\,(x) \rightarrow (\text{length}\,(\text{cdr}\,(\text{rev3}\,(x))) = (\text{length}\,(x) - 1))$

THEOREM: rev3-rev-lemma

$(\text{plistp}\,(x) \wedge (\text{length}\,(x) = n)) \rightarrow (\text{rev3}\,(x) = \text{rev}\,(x))$

THEOREM: rev3-rev

$\text{plistp}\,(x) \rightarrow (\text{rev3}\,(x) = \text{rev}\,(x))$

Index