EVENT: Start with the initial {f nqthm} theory.

;; Sets; Matt Kaufmann, Dec. 1989, revised March 1990. The first few events
;; are some basic events about lists. I’ll take the approach that all these
;; basic functions will be disabled once enough algebraic properties have
;; been proved.

;; Theories:

;; (deftheory set-defns
;; (length properp fix-properp member append subsetp delete
;; (disjoint intersection set-diff setp))

DEFINITION:
length (x)
= if listp (x) then 1 + length (cdr (x))
  else 0 endif

THEOREM: length-nlistp
(x ≃ nil) → (length (x) = 0)

THEOREM: length-cons
length (cons (a, x)) = (1 + length (x))

THEOREM: length-append
length (append (x, y)) = (length (x) + length (y))

EVENT: Disable length.

THEOREM: append-assoc
append (append (x, y), z) = append (x, append (y, z))

THEOREM: member-cons
(a ∈ cons (x, l)) = ((a = x) ∨ (a ∈ l))

THEOREM: member-nlistp
(l ≃ nil) → (a ∉ l)

EVENT: Disable member.

DEFINITION:
subsetp (x, y)
= if x ≃ nil then t
  else (car (x) ∈ y) ∧ subsetp (cdr (x), y) endif
Definition:
\text{subsetp-wit} (x, y) = \begin{cases} \text{t} & \text{if } x \simeq \text{nil} \\ \text{subsetp-wit} (\text{cdr} (x), y) & \text{elseif } \text{car} (x) \in y \\ \text{car} (x) & \text{endif} \end{cases}

Theorem: subsetp-wit-witnesses
\text{subsetp} (x, y) = (\sim ((\text{subsetp-wit} (x, y) \in x) \land (\text{subsetp-wit} (x, y) \not\in y)))

Theorem: subsetp-wit-witnesses-general-1
((\text{subsetp-wit} (x, y) \not\in x) \land (a \in x)) \rightarrow (a \in y)

Theorem: subsetp-wit-witnesses-general-2
((\text{subsetp-wit} (x, y) \in y) \land (a \in x)) \rightarrow (a \in y)

Event: Disable subsetp-wit-witnesses.

Event: Disable subsetp-wit-witnesses-general-1.


Theorem: subsetp-cons-1
\text{subsetp} (\text{cons} (a, x), y) = ((a \in y) \land \text{subsetp} (x, y))

Theorem: subsetp-cons-2
\text{subsetp} (l, m) \rightarrow \text{subsetp} (l, \text{cons} (a, m))

Theorem: subsetp-reflexivity
\text{subsetp} (x, x)

Theorem: cdr-subsetp
\text{subsetp} (\text{cdr} (x), x)

Theorem: member-subsetp
((x \in y) \land \text{subsetp} (y, z)) \rightarrow (x \in z)

Theorem: subsetp-is-transitive
(\text{subsetp} (x, y) \land \text{subsetp} (y, z)) \rightarrow \text{subsetp} (x, z)

Theorem: member-append
(a \in \text{append} (x, y)) = ((a \in x) \lor (a \in y))

Theorem: subsetp-append
\text{subsetp} (\text{append} (x, y), z) = (\text{subsetp} (x, z) \land \text{subsetp} (y, z))
THEOREM: subsetp-of-append-sufficiency
(subsetp (a, b) ∨ subsetp (a, c)) → subsetp (a, append (b, c))

THEOREM: subsetp-nilp
(x ≃ nil) → (subsetp (x, y) ∧ (subsetp (y, x) = (y ≃ nil)))

THEOREM: subsetp-cons-not-member
(z ∉ x) → (subsetp (x, cons (z, v)) = subsetp (x, v))

EVENT: Disable subsetp.

;;;;;; Other set-theoretic and list-theoretic definitions, and properp observations.

DEFINITION:
properp (x)
= if listp (x) then properp (cdr (x))
   else x = nil endif

DEFINITION:
fix-properp (x)
= if listp (x) then cons (car (x), fix-properp (cdr (x)))
   else nil endif

THEOREM: properp-fix-properp
properp (fix-properp (x))

THEOREM: fix-properp-properp
properp (x) → (fix-properp (x) = x)

THEOREM: properp-cons
properp (cons (x, y)) = properp (y)

THEOREM: properp-nilp
(x ≃ nil) → (properp (x) = (x = nil))

THEOREM: fix-properp-cons
fix-properp (cons (x, y)) = cons (x, fix-properp (y))

THEOREM: fix-properp-nilp
(x ≃ nil) → (fix-properp (x) = nil)

THEOREM: properp-append
properp (append (x, y)) = properp (y)

THEOREM: fix-properp-append
fix-properp (append (x, y)) = append (x, fix-properp (y))
Theorem: append-nil
\[ \text{append} \ (x, \text{nil}) = \text{fix-properp} \ (x) \]

Definition:
\[
\text{delete} \ (x, \ l) =
\begin{cases} 
\text{if} \ \text{listp} \ (l) \\
\quad \text{then} \ \text{if} \ x = \text{car} \ (l) \ \text{then} \ \text{cdr} \ (l) \\
\quad \quad \text{else} \ \text{cons} \ (\text{car} \ (l), \ \text{delete} \ (x, \ \text{cdr} \ (l))) \ \text{endif} \\
\quad \text{else} \ l \ \text{endif}
\end{cases}
\]

Theorem: properp-delete
\[
\text{properp} \ (\text{delete} \ (x, \ l)) = \text{properp} \ (l)
\]

Definition:
\[
\text{disjoint} \ (x, \ y) =
\begin{cases} 
\text{if} \ \text{listp} \ (x) \\
\quad \text{then} \ (\text{car} \ (x) \notin y) \land \text{disjoint} \ (\text{cdr} \ (x), \ y) \\
\quad \text{else} \ \text{t} \ \text{endif}
\end{cases}
\]

Definition:
\[
\text{disjoint-wit} \ (x, \ y) =
\begin{cases} 
\text{if} \ \text{listp} \ (x) \\
\quad \text{then} \ \text{if} \ \text{car} \ (x) \in y \ \text{then} \ \text{car} \ (x) \\
\quad \quad \text{else} \ \text{disjoint-wit} \ (\text{cdr} \ (x), \ y) \ \text{endif} \\
\quad \text{else} \ \text{t} \ \text{endif}
\end{cases}
\]

Theorem: disjoint-wit-witnesses
\[
\text{disjoint} \ (x, \ y) = \neg ((\text{disjoint-wit} \ (x, \ y) \in x) \land (\text{disjoint-wit} \ (x, \ y) \in y))
\]

Event: Disable disjoint-wit.

Event: Disable disjoint-wit-witnesses.

Definition:
\[
\text{intersection} \ (x, \ y) =
\begin{cases} 
\text{if} \ \text{listp} \ (x) \\
\quad \text{then} \ \text{if} \ \text{car} \ (x) \in y \ \text{then} \ \text{cons} \ (\text{car} \ (x), \ \text{intersection} \ (\text{cdr} \ (x), \ y)) \\
\quad \quad \text{else} \ \text{intersection} \ (\text{cdr} \ (x), \ y) \ \text{endif} \\
\quad \text{else} \ \text{nil} \ \text{endif}
\end{cases}
\]

Theorem: properp-intersection
\[
\text{properp} \ (\text{intersection} \ (x, \ y))
\]
**Definition:**

\[
\text{set-diff}(x, y) = \begin{cases} 
\text{if } \text{listp}(x) \text{ then if } \text{car}(x) \in y \text{ then } \text{set-diff}(\text{cdr}(x), y) \
\text{else cons}(\text{car}(x), \text{set-diff}(\text{cdr}(x), y)) \end{cases} \text{ endif} \\
\text{else nil endif}
\]

**Theorem:** properp-set-diff

\[\text{properp } \text{set-diff}(x, y)\]

**Definition:**

\[
\text{setp}(x) = \begin{cases} 
\text{if } \neg \text{listp}(x) \text{ then } x = \text{nil} \\
\text{else } (\text{car}(x) \not\in \text{cdr}(x)) \land \text{setp}(\text{cdr}(x)) \end{cases} \text{ endif}
\]

**Theorem:** setp-implies-properp

\[\text{setp}(x) \rightarrow \text{properp}(x)\]

**Event:** Disable properp.

**Event:** Let us define the theory \textit{set-defns} to consist of the following events:
 length, properp, fix-properp, member, append, subsetp, delete, disjoint, intersection, set-diff, setp, properp.

;; Set theory lemmas

**Theorem:** delete-cons

\[\text{delete}(a, \text{cons}(b, x)) = \begin{cases} 
\text{if } a = b \text{ then } x \\
\text{else cons}(b, \text{delete}(a, x)) \end{cases} \text{ endif}\]

**Theorem:** delete-nlistp

\[(x \approx \text{nil}) \rightarrow (\text{delete}(a, x) = x)\]

**Theorem:** listp-delete

\[\text{listp}(\text{delete}(x, l)) = \begin{cases} 
\text{if } \text{listp}(l) \text{ then } (x \neq \text{car}(l)) \lor \text{listp}(\text{cdr}(l)) \\
\text{else } \text{false} \end{cases} \text{ endif}\]

**Theorem:** delete-non-member

\[(x \not\in y) \rightarrow (\text{delete}(x, y) = y)\]

**Theorem:** delete-delete

\[\text{delete}(y, \text{delete}(x, z)) = \text{delete}(x, \text{delete}(y, z))\]
Theorem: member-delete
setp (x) \rightarrow ((a \in \text{delete}(b, x)) = ((a \neq b) \land (a \in x)))

Theorem: setp-delete
setp (x) \rightarrow \text{setp} (\text{delete}(a, x))

Event: Disable delete.

Theorem: disjoint-cons-1
\text{disjoint}(\text{cons}(a, x), y) = ((a \notin y) \land \text{disjoint}(x, y))

Theorem: disjoint-cons-2
\text{disjoint}(x, \text{cons}(a, y)) = ((a \notin x) \land \text{disjoint}(x, y))

Theorem: disjoint-nlistp
((x \simeq \textsf{nil}) \lor (y \simeq \textsf{nil})) \rightarrow \text{disjoint}(x, y)

Theorem: disjoint-symmetry
\text{disjoint}(x, y) = \text{disjoint}(y, x)

Theorem: disjoint-append-right
\text{disjoint}(u, \text{append}(y, z)) = (\text{disjoint}(u, y) \land \text{disjoint}(u, z))

Theorem: disjoint-append-left
\text{disjoint}(\text{append}(y, z), u) = (\text{disjoint}(y, u) \land \text{disjoint}(z, u))

Theorem: disjoint-non-member
((a \in x) \land (a \in y)) \rightarrow (\neg \text{disjoint}(x, y))

Theorem: disjoint-subsetp-monotone-second
(\text{subsetp}(y, z) \land \text{disjoint}(x, z)) \rightarrow \text{disjoint}(x, y)

Theorem: subsetp-disjoint-2
(\text{subsetp}(x, y) \land \text{disjoint}(y, z)) \rightarrow \text{disjoint}(z, x)

Theorem: subsetp-disjoint-1
(\text{subsetp}(x, y) \land \text{disjoint}(y, z)) \rightarrow \text{disjoint}(x, z)

Theorem: subsetp-disjoint-3
(\text{subsetp}(x, y) \land \text{disjoint}(z, y)) \rightarrow \text{disjoint}(x, z)

Event: Disable disjoint.

Theorem: intersection-disjoint
(\text{intersection}(x, y) = \textsf{nil}) = \text{disjoint}(x, y)
Theorem: intersection-nilistp

\((x \simeq \texttt{nil}) \lor (y \simeq \texttt{nil})) \rightarrow (\text{intersection}(x, y) = \texttt{nil})\)

Theorem: member-intersection

\((a \in \text{intersection}(x, y)) = ((a \in x) \land (a \in y))\)

Theorem: subsetp-intersection

\(\text{subsetp}(x, \text{intersection}(y, z)) = (\text{subsetp}(x, y) \land \text{subsetp}(x, z))\)

Theorem: intersection-symmetry

\(\text{subsetp}(\text{intersection}(x, y), y) \land \text{subsetp}(x, y)\)

Theorem: intersection-cons-1

\(\text{intersection}(\text{cons}(a, x), y) = \begin{cases} \text{cons}(a, \text{intersection}(x, y)) & \text{if } a \in y \\ \text{intersection}(x, y) & \text{else} \end{cases}\)

Theorem: intersection-cons-2

\((a \not\in y) \rightarrow (\text{intersection}(y, \text{cons}(a, x)) = \text{intersection}(y, x))\)

;; The following is needed because DISJOINT-INTERSECTION-COMMUTER, added during polishing, caused the proof of ;; DISJOINT-DOMAIN-CO-RESTRICT (in "alists.events") to fail.

Theorem: intersection-cons-3

\((w \in x) \rightarrow (\text{subsetp}(\text{intersection}(y, \text{cons}(w, z)), x) = \text{subsetp}(\text{intersection}(y, z), x))\)

Theorem: intersection-cons-subsetp

\(\text{subsetp}(\text{intersection}(x, y), \text{intersection}(x, \text{cons}(a, y)))\)

Theorem: subsetp-intersection-left-1

\(\text{subsetp}(\text{intersection}(x, y), x)\)

Theorem: subsetp-intersection-left-2

\(\text{subsetp}(\text{intersection}(x, y), y)\)

Theorem: subsetp-intersection-sufficiency-1

\(\text{subsetp}(y, z) \rightarrow \text{subsetp}(\text{intersection}(x, y), z)\)

Theorem: subsetp-intersection-sufficiency-2

\(\text{subsetp}(y, z) \rightarrow \text{subsetp}(\text{intersection}(y, x), z)\)

Theorem: intersection-associative

\(\text{intersection}(\text{intersection}(x, y), z) = \text{intersection}(x, \text{intersection}(y, z))\)
**Theorem:** intersection-elimination
\[ \text{subsetp} (x, y) \rightarrow (\text{intersection} (x, y) = \text{fix-properp} (x)) \]

**Theorem:** length-intersection
\[ \text{length (x) \not< \text{length (intersection} (x, y))} \]

**Theorem:** subsetp-intersection-member
\[ \text{subsetp (intersection} (x, y), z) \land (a \not\in z) \rightarrow ((a \in x) \rightarrow (a \not\in y)) \land ((a \in y) \rightarrow (a \not\in x)) \]

;; The following wasn’t needed in the proof about generalization, but it’s a nice rule.

**Theorem:** intersection-append
\[ \text{intersection (append} (x, y), z) = \text{append (intersection} (x, z), \text{intersection} (y, z)) \]

;; I’d rather just prove that intersection distributes over append on the right but that isn’t true. Congruence relations would probably help a lot with that problem. In the meantime, I content myself with the following.

**Theorem:** disjoint-intersection-append
\[ \text{disjoint} (x, \text{intersection} (y, \text{append} (z1, z2))) = (\text{disjoint} (x, \text{intersection} (y, z1)) \land \text{disjoint} (x, \text{intersection} (y, z2))) \]

;; See comment just above DISJOINT-INTERSECTION-APPEND

**Theorem:** subsetp-intersection-append
\[ \text{subsetp} (\text{intersection} (u, \text{append} (x, y)), z) = (\text{subsetp} (\text{intersection} (u, x), z) \land \text{subsetp} (\text{intersection} (u, y), z)) \]

**Theorem:** subsetp-intersection-elimination-lemma
\[ (\text{subsetp} (y, x) \land \neg \text{subsetp} (y, z)) \rightarrow \neg \text{subsetp} (\text{intersection} (x, y), z) \]

**Theorem:** subsetp-intersection-elimination
\[ \text{subsetp} (y, x) \rightarrow (\text{subsetp} (\text{intersection} (x, y), z) \leftrightarrow \text{subsetp} (y, z)) \]

**Theorem:** disjoint-intersection
\[ \text{disjoint} (\text{intersection} (x, y), z) = \text{disjoint} (x, \text{intersection} (y, z)) \]

**Theorem:** subsetp-intersection-monotone-1
\[ (\text{subsetp} (\text{intersection} (x, y), z) \land \text{subsetp} (xI, x)) \rightarrow \text{subsetp} (\text{intersection} (xI, y), z) \]
The lemma SUBSETP-INTERSECTION-MONOTONE-2 below was added during polishing of the final proof in "generalize.events", since the lemma immediately above wasn't enough at that point. Actually, I realized at this point that intersection commutes from the point of view of subsetp:

**Theorem**: subsetp-intersection-commuter
\[ \text{subsetp}\ (\text{intersection}\ (x, y), z) = \text{subsetp}\ (\text{intersection}\ (y, x), z) \]

**Theorem**: subsetp-intersection-monotone-2
\[ (\text{subsetp}\ (\text{intersection}\ (y, x), z) \land \text{subsetp}\ (x1, x)) \rightarrow \text{subsetp}\ (\text{intersection}\ (x1, y), z) \]

**Theorem**: disjoint-intersection-commuter
\[ \text{disjoint}\ (x, \text{intersection}\ (y, z)) = \text{disjoint}\ (x, \text{intersection}\ (z, y)) \]

**Theorem**: disjoint-intersection3
\[ \text{disjoint}\ (\text{free}, \text{intersection}\ (vars, x)) \rightarrow (\text{intersection}\ (x, \text{intersection}\ (vars, free)) = \text{nil}) \]

**Event**: Disable intersection.

**Theorem**: member-set-diff
\[ (a \in \text{set-diff}\ (y, z)) = ((a \in y) \land (a \notin z)) \]

**Theorem**: subsetp-set-diff-1
\[ \text{subsetp}\ (\text{set-diff}\ (x, y), x) \]

**Theorem**: disjointp-set-diff
\[ \text{disjoint}\ (\text{set-diff}\ (x, y), y) \]

**Theorem**: subsetp-set-diff-2
\[ \text{subsetp}\ (x, \text{set-diff}\ (y, z)) = (\text{subsetp}\ (x, y) \land \text{disjoint}\ (x, z)) \]

**Theorem**: set-diff-cons
\[ \text{set-diff}\ (\text{cons}\ (a, x), y) = \begin{cases} \text{set-diff}\ (x, y) & \text{if } a \in y \\ \text{cons}\ (a, \text{set-diff}\ (x, y)) & \text{else} \end{cases} \text{endif} \]

**Theorem**: set-diff-nlistp
\[ (x \simeq \text{nil}) \rightarrow (\text{set-diff}\ (x, y) = \text{nil}) \]

;; The following was discovered during final polishing, for the proof of MAIN-HYPS-RELIEVED-6-FIRST.
Theorem: disjoint-set-diff-general
\[ \text{disjoint} (x, \text{set-diff} (y, z)) = \text{subsetp} (\text{intersection} (x, y), z) \]

;; No longer relevant:
(prove-lemma disjoint-set-diff-subsetp (rewrite)
; (implies (and (disjoint x (set-diff y z))
; (subsetp z z1))
; (disjoint x (set-diff y z1)))
; ((use (disjoint-wit-witnesses (y (set-diff y z1)))))
; (disable member-set-diff set-diff))

;; Instead of the following I’ll prove the corresponding (in light of
;; DISJOINT-SET-DIFF-GENERAL) fact INTERSECTION-X-X about intersection.
(prove-lemma disjoint-set-diff (rewrite)
; (disjoint x (set-diff y x)))

Theorem: intersection-subsetp-identity
\[ (\text{properp} (x) \land \text{subsetp} (x, y)) \rightarrow (\text{intersection} (x, y) = x) \]

Theorem: intersection-x-x
\[ \text{properp} (x) \rightarrow (\text{intersection} (x, x) = x) \]

Theorem: subsetp-set-diff-monotone-2
\[ \text{subsetp} (\text{set-diff} (x, \text{append} (y, z)), \text{set-diff} (x, z)) \]

Theorem: subsetp-set-diff-monotone-second
\[ \text{subsetp} (\text{set-diff} (x, y), \text{set-diff} (x, z)) = \text{subsetp} (\text{intersection} (x, z), y) \]

Theorem: set-diff-nil
\[ \text{set-diff} (x, \text{nil}) = \text{fix-properp} (x) \]

Theorem: set-diff-cons-non-member-1
\[ (a \not\in x) \rightarrow (\text{set-diff} (x, \text{cons} (a, y)) = \text{set-diff} (x, y)) \]

Theorem: length-intersection-set-diff
\[ \text{length} (x) = (\text{length} (\text{set-diff} (x, y)) + \text{length} (\text{intersection} (x, y))) \]

Theorem: length-set-diff-opener
\[ \text{length} (\text{set-diff} (x, y)) = (\text{length} (x) - \text{length} (\text{intersection} (x, y))) \]

Theorem: listp-set-diff
\[ \text{listp} (\text{set-diff} (x, y)) = (\neg \text{subsetp} (x, y)) \]

;; Here is a messy lemma about disjoint and such
THEOREM: disjoint-intersection-set-diff-intersection
disjoint \((x, \text{intersection}(y, \text{set-diff}(z, \text{intersection}(y, x))))\)

EVENT: Disable set-diff.

THEOREM: member-fix-properp
\((a \in \text{fix-properp}(x)) = (a \in x)\)

THEOREM: setp-append
\(\text{setp}(\text{append}(x, y)) = (\text{disjoint}(x, y) \land \text{setp}(\text{fix-properp}(x)) \land \text{setp}(y))\)

THEOREM: setp-cons
\(\text{setp}(\text{cons}(a, x)) = ((a \not\in x) \land \text{setp}(x))\)

THEOREM: setp-nlistp
\((x \simeq \text{nil}) \rightarrow (\text{setp}(x) = (x = \text{nil}))\)

DEFINITION:
\[
\text{make-set}(l) = \begin{cases} 
\text{nil} & \text{if } \neg \text{listp}(l) \\
\text{cons}(\text{car}(l), \text{make-set}(\text{cdr}(l))) & \text{else if } \text{car}(l) \in \text{cdr}(l) \\
\text{cons}(\text{car}(l), \text{make-set}(\text{cdr}(l))) & \text{else}
\end{cases}
\]

THEOREM: make-set-preserves-member
\((x \in \text{make-set}(l)) = (x \in l)\)

THEOREM: make-set-preserves-subsetp-1
\(\text{subsetp}(\text{make-set}(x), \text{make-set}(y)) = \text{subsetp}(x, y)\)

THEOREM: make-set-preserves-subsetp-2
\(\text{subsetp}(x, \text{make-set}(y)) = \text{subsetp}(x, y)\)

THEOREM: make-set-preserves-subsetp-3
\(\text{subsetp}(\text{make-set}(x), y) = \text{subsetp}(x, y)\)

THEOREM: make-set-gives-setp
\(\text{setp}(\text{make-set}(x))\)

THEOREM: make-set-set-diff
\(\text{make-set}(\text{set-diff}(x, y)) = \text{set-diff}(\text{make-set}(x), \text{make-set}(y))\)

THEOREM: set-diff-make-set
\(\text{set-diff}(x, \text{make-set}(y)) = \text{set-diff}(x, y)\)

THEOREM: listp-make-set
\(\text{listp}(\text{make-set}(x)) = \text{listp}(x)\)
Event: Disable setp.

;;;;;; The following were proved in the course of the final run
;;;;;; through the generalization proof. There are a couple or
;;;;;; so noted above here, too.

**Theorem:** set-diff-append
\[
\text{set-diff}(x, \text{append}(y, z)) = \text{set-diff}(\text{set-diff}(x, z), y)
\]

**Theorem:** length-set-diff-leq
\[
\text{length}(x) \not< \text{length}(\text{set-diff}(x, y))
\]

**Theorem:** lessp-length
\[
\text{listp}(x) \rightarrow (0 < \text{length}(x))
\]

**Theorem:** listp-intersection
\[
\text{listp}(\text{intersection}(x, y)) = (\neg \text{disjoint}(x, y))
\]

**Theorem:** length-set-diff-lessp
\[
(\neg \text{disjoint}(x, \text{new})) \rightarrow (\text{length}(\text{set-diff}(x, \text{new})) < \text{length}(x))
\]

**Theorem:** disjoint-implies-empty-intersection
\[
\text{disjoint}(x, y) \rightarrow (\text{intersection}(x, y) = \text{nil})
\]

;;;; The following lemma DISJOINT-INTERSECTION3-MIDDLE is needed for the
;;;; proof of ALL-VARS-DISJOINT-OR-SUBSETP-GEN-CLOSURE in
;;;; generalize.events. I think I could avoid lemmas like this one
;;;; INTERSECTION were actually commutative-associative (in which case
;;;; I’d get rid of disjoint and rely on normalization).

**Theorem:** disjoint-intersection3-middle
\[
\neg \text{disjoint}(x, \text{intersection}(u, v)) \land \text{subsetp}(w, x) \rightarrow \neg \text{disjoint}(u, \text{intersection}(w, v))
\]

;;;; Maybe I should redo the notion of disjoint sometime, perhaps using
;;;; the fact that intersection is commutative and associative when it’s
;;;; equated with nil.
THEOREM: subsetp-set-diff-sufficiency
\[ \text{subsetp} (x, y) \rightarrow \text{subsetp} (\text{set-diff} (x, z), y) \]

;; The following lemma SETP-INTERSECTION-SUFFICIENCY is needed for
;; MAPPING-RESTRICT from "alists.events", because (I believe)
;; DOMAIN-RESTRICT, which was added during polishing, changed the
;; course of the previous proof. Similarly for
;; SETP-SET-DIFF-SUFFICIENCY and the lemma MAPPING-CO-RESTRICT.

THEOREM: setp-intersection-sufficiency
\[ \text{setp} (x) \rightarrow \text{setp} (\text{intersection} (x, y)) \]

THEOREM: setp-set-diff-sufficiency
\[ \text{setp} (x) \rightarrow \text{setp} (\text{set-diff} (x, y)) \]

;; The definition of FIX-PROPERP was also added in polishing because
;; of a problem with the proof of GEN-CLOSURE-ACCEPT in
;; "generalize.events". Here are a couple of lemmas about it that
;; might or might not be useful; all other lemmas about it above, and
;; the definition, were added during polishing.

EVENT: Disable fix-properp.

THEOREM: subsetp-fix-properp-1
\[ \text{subsetp} (\text{fix-properp} (x), y) = \text{subsetp} (x, y) \]

THEOREM: subsetp-fix-properp-2
\[ \text{subsetp} (x, \text{fix-properp} (y)) = \text{subsetp} (x, y) \]

EVENT: Make the library "sets".

13
Index

append-assoc, 1
append-nil, 4

cdr-subsetp, 2
delete, 4–6
delete-cons, 5
delete-delete, 5
delete-nlistp, 5
delete-non-member, 5
disjoint, 4, 6, 8–12
disjoint-append-left, 6
disjoint-append-right, 6
disjoint-cons-1, 6
disjoint-cons-2, 6

disjoint-implies-empty-intersection, 12
disjoint-intersection, 8
disjoint-intersection-append, 8
disjoint-intersection-commuter, 9
disjoint-intersection-set-diff
intersection, 11
disjoint-intersection3, 9
disjoint-intersection3-middle, 12
disjoint-nlistp, 6
disjoint-non-member, 6
disjoint-set-diff-general, 10
disjoint-subsetp-hack, 12
disjoint-subsetp-monotone-secon
d, 6
disjoint-symmetry, 6
disjoint-wit, 4
disjoint-wit-witnesses, 4
disjointp-set-diff, 9

fix-properp, 3, 4, 8, 10, 11, 13
fix-properp-append, 3
fix-properp-cons, 3
fix-properp-nlistp, 3
fix-properp-properp, 3

intersection, 4, 6–13
intersection-append, 8
intersection-associative, 7
intersection-cons-1, 7
intersection-cons-2, 7
intersection-cons-3, 7
intersection-cons-subsetp, 7
intersection-disjoint, 6
intersection-elimination, 8
intersection-nlistp, 7
intersection-subsetp-identity, 10
intersection-symmetry, 7
intersection-x-x, 10

length, 1, 8, 10, 12
length-append, 1
length-cons, 1
length-intersection, 8
length-intersection-set-diff, 10
length-nlistp, 1
length-set-diff-leq, 12
length-set-diff-lessp, 12
length-set-diff-opener, 10
lessp-length, 12
listp-delete, 5
listp-intersection, 12
listp-make-set, 11
listp-set-diff, 10

make-set, 11
make-set-gives-setp, 11
make-set-preserves-member, 11
make-set-preserves-subsetp-1, 11
make-set-preserves-subsetp-2, 11
make-set-preserves-subsetp-3, 11
make-set-set-diff, 11
member-append, 2
member-cons, 1
member-delete, 6
member-fix-properp, 11
member-intersection, 7
member-nlistp, 1