```
;; Requires sets.
```

EVENT: Start with the library `"sets"`.

```
;; Alists, March 1990.  Most of the definitions and some of the lemmas
;; were contributed by Bill Bevier; the rest are by Matt Kaufmann.

;; Functions defined here:

;; (deftheory alist-defns
;;   (alistp domain range value bind rembind invert mapping
;;           restrict co-restrict))
```

DEFINITION:
alistp (x)
$=$ **if** listp (x) **then** listp $(\text{car}\,(x)) \wedge$ alistp $(\text{cdr}\,(x))$
else $x =$ **nil endif**

THEOREM: alistp-implies-properp
alistp $(x) \rightarrow$ properp (x)

THEOREM: alistp-nlistp
$(x \simeq \textbf{nil}) \rightarrow (\text{alistp}\,(x) = (x = \textbf{nil}))$

THEOREM: alistp-cons
alistp $(\text{cons}\,(a,\ x)) = (\text{listp}\,(a) \wedge \text{alistp}\,(x))$

EVENT: Disable alistp.

THEOREM: alistp-append
alistp $(\text{append}\,(x,\ y)) = (\text{alistp}\,(\text{fix-properp}\,(x)) \wedge \text{alistp}\,(y))$

DEFINITION:
domain (map)
$=$ **if** listp (map)
then if listp $(\text{car}\,(map))$ **then** cons $(\text{car}\,(\text{car}\,(map)),\ \text{domain}\,(\text{cdr}\,(map)))$
else domain $(\text{cdr}\,(map))$ **endif**
else nil endif

THEOREM: properp-domain
properp $(\text{domain}\,(map))$

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

THEOREM: domain-nlistp
($map \simeq$ **nil**) $\rightarrow$ (domain (map) = **nil**)

THEOREM: domain-cons
domain (cons (a, map))
= **if** listp (a) **then** cons (car (a), domain (map))
else domain (map) **endif**

THEOREM: member-domain-sufficiency
(cons (a, x) $\in$ y) $\rightarrow$ (a $\in$ domain (y))

THEOREM: subsetp-domain
subsetp (x, y) $\rightarrow$ subsetp (domain (x), domain (y))

EVENT: Disable domain.

DEFINITION:
range (map)
= **if** listp (map)
then **if** listp (car (map)) **then** cons (cdr (car (map)), range (cdr (map)))
else range (cdr (map)) **endif**
else **nil** **endif**

THEOREM: properp-range
properp (range (map))

THEOREM: range-append
range (append ($s1$, $s2$)) = append (range ($s1$), range ($s2$))

THEOREM: range-nlistp
($map \simeq$ **nil**) $\rightarrow$ (range (map) = **nil**)

THEOREM: range-cons
range (cons (a, map))
= **if** listp (a) **then** cons (cdr (a), range (map))
else range (map) **endif**

EVENT: Disable range.

```
;; BOUNDP has been eliminated in favor of membership in domain.
;; Notice that I have to talk about things like disjointness of
;; domains anyhow.  New definition body would be (member x (domain map)).
```

```
;(defn boundp (x map)
;  (if (listp map)
;      (if (listp (car map))
;          (if (equal x (caar map))
;              t
;            (boundp x (cdr map)))
;        (boundp x (cdr map)))
;    f))
```

DEFINITION:
value $(x,\ map)$
$=$ **if** listp (map)
then if listp (car $(map)) \wedge (x = \text{caar}\ (map))$ **then** cdar (map)
else value $(x,$ cdr $(map))$ **endif**
else 0 **endif**

THEOREM: value-nlistp
$(map \simeq \textbf{nil}) \rightarrow (\text{value}\ (x,\ map) = 0)$

THEOREM: value-cons
value $(x,$ cons $(pair,\ map))$
$=$ **if** listp $(pair) \wedge (x = \text{car}\ (pair))$ **then** cdr $(pair)$
else value $(x,\ map)$ **endif**

EVENT: Disable value.

DEFINITION:
bind $(x,\ v,\ map)$
$=$ **if** listp (map)
then if listp (car (map))
then if $x =$ caar (map) **then** cons (cons $(x,\ v)$, cdr (map))
else cons (car (map), bind $(x,\ v,$ cdr (map))) **endif**
else cons (car (map), bind $(x,\ v,$ cdr (map))) **endif**
else cons (cons $(x,\ v)$, **nil**) **endif**

DEFINITION:
rembind $(x,\ map)$
$=$ **if** listp (map)
then if listp (car (map))
then if $x =$ caar (map) **then** cdr (map)
else cons (car (map), rembind $(x,$ cdr (map))) **endif**
else cons (car (map), rembind $(x,$ cdr (map))) **endif**
else nil endif

DEFINITION:
invert (map)
$=$ **if** listp (map)
then if listp (car (map))
then cons (cons (cdr (car (map)), car (car (map))), invert (cdr (map)))
else invert (cdr (map)) **endif**
else nil endif

THEOREM: properp-invert
properp (invert (map))

THEOREM: invert-nlistp
$(map \simeq \mathbf{nil}) \rightarrow (\text{invert}(map) = \mathbf{nil})$

THEOREM: invert-cons
invert (cons $(pair,\ map)$)
$=$ **if** listp $(pair)$ **then** cons (cons (cdr $(pair)$, car $(pair)$), invert (map))
else invert (map) **endif**

THEOREM: value-invert-not-member-of-domain
$((g \in \text{range}(sg)) \wedge \text{disjoint}(\text{domain}(s),\ \text{domain}(sg)))$
$\rightarrow \quad (\text{value}(g,\ \text{invert}(sg)) \notin \text{domain}(s))$

EVENT: Disable invert.

DEFINITION: mapping $(map) = (\text{alistp}(map) \wedge \text{setp}(\text{domain}(map)))$

```
;; For when we disable mapping:
```

THEOREM: mapping-implies-alistp
mapping $(map) \rightarrow$ alistp (map)

THEOREM: mapping-implies-setp-domain
mapping $(map) \rightarrow$ setp (domain (map))

DEFINITION:
restrict $(s,\ new\text{-}domain)$
$=$ **if** listp (s)
then if listp (car (s)) $\wedge$ (caar $(s) \in new\text{-}domain$)
then cons (car (s), restrict (cdr (s), $new\text{-}domain$))
else restrict (cdr (s), $new\text{-}domain$) **endif**
else nil endif

DEFINITION:
co-restrict $(s,\ new\text{-}domain)$

$=$ **if** listp (s)
then **if** listp (car (s)) $\wedge$ (caar $(s) \notin$ *new-domain*)
then cons (car (s), co-restrict (cdr (s), *new-domain*))
else co-restrict (cdr (s), *new-domain*) **endif**
else nil endif

EVENT: Let us define the theory *alist-defns* to consist of the following events: alistp, domain, range, value, bind, rembind, invert, mapping, restrict, co-restrict.

```
;;;;; alist lemmas

; DOMAIN

;; The following was proved in the course of the final run through
;; the generalization proof.  The one after it isn't needed but
;; seems like it's worth proving too.  Actually now I see that
;; some other lemmas are now obsolete, so I'll put these both
;; early in the file and delete the others.
```

THEOREM: domain-restrict
domain (restrict $(s,\ dom)$) $=$ intersection (domain (s), dom)

THEOREM: domain-co-restrict
domain (co-restrict $(s,\ dom)$) $=$ set-diff (domain (s), dom)

THEOREM: domain-bind
domain (bind $(x,\ v,\ map)$)
$=$ **if** $x \in$ domain (map) **then** domain (map)
else append (domain (map), list (x)) **endif**

THEOREM: domain-rembind
domain (rembind $(x,\ map)$) $=$ delete (x, domain (map))

THEOREM: domain-invert
domain (invert (map)) $=$ range (map)

```
; RANGE
```

THEOREM: range-invert
range (invert (map)) $=$ domain (map)

```
; BOUNDP
```

THEOREM: boundp-bind
$(x \in \text{domain}\,(\text{bind}\,(y,\ v,\ map))) = ((x = y) \lor (x \in \text{domain}\,(map)))$

THEOREM: boundp-rembind
$$\begin{aligned}&\text{mapping}\,(map)\\ \rightarrow\quad &((x \in \text{domain}\,(\text{rembind}\,(y,\ map)))\\ &\quad = \ \textbf{if}\ x = y\ \textbf{then f}\\ &\qquad\ \ \textbf{else}\ x \in \text{domain}\,(map)\ \textbf{endif})\end{aligned}$$

THEOREM: boundp-subsetp
$$\begin{aligned}&(\text{subsetp}\,(map1,\ map2) \land (name \in \text{domain}\,(map1)))\\ \rightarrow\quad &(name \in \text{domain}\,(map2))\end{aligned}$$

THEOREM: disjoint-domain-singleton
$$\begin{aligned}&(\text{disjoint}\,(\text{domain}\,(s),\ \text{list}\,(x)) = (x \notin \text{domain}\,(s)))\\ \land\quad &(\text{disjoint}\,(\text{list}\,(x),\ \text{domain}\,(s)) = (x \notin \text{domain}\,(s)))\end{aligned}$$

THEOREM: boundp-value-invert
$(x \in \text{range}\,(map)) \rightarrow (\text{value}\,(x,\ \text{invert}\,(map)) \in \text{domain}\,(map))$

; VALUE

THEOREM: value-when-not-bound
$(name \notin \text{domain}\,(map)) \rightarrow (\text{value}\,(name,\ map) = 0)$

THEOREM: value-bind
$$\begin{aligned}&\text{value}\,(x,\ \text{bind}\,(y,\ v,\ map))\\ =\quad &\textbf{if}\ x = y\ \textbf{then}\ v\\ &\textbf{else}\ \text{value}\,(x,\ map)\ \textbf{endif}\end{aligned}$$

THEOREM: value-rembind
$$\begin{aligned}&\text{mapping}\,(map)\\ \rightarrow\quad &(\text{value}\,(x,\ \text{rembind}\,(y,\ map))\\ &\quad = \ \textbf{if}\ x = y\ \textbf{then}\ 0\\ &\qquad\ \ \textbf{else}\ \text{value}\,(x,\ map)\ \textbf{endif})\end{aligned}$$

THEOREM: value-append
$$\begin{aligned}&\text{value}\,(x,\ \text{append}\,(s1,\ s2))\\ =\quad &\textbf{if}\ x \in \text{domain}\,(s1)\ \textbf{then}\ \text{value}\,(x,\ s1)\\ &\textbf{else}\ \text{value}\,(x,\ s2)\ \textbf{endif}\end{aligned}$$

THEOREM: value-value-invert
$((x \in \text{range}\,(s)) \land \text{mapping}\,(s)) \rightarrow (\text{value}\,(\text{value}\,(x,\ \text{invert}\,(s)),\ s) = x)$

; MAPPING

THEOREM: mapping-append

$$\begin{array}{l} \text{mapping}\,(\text{append}\,(s1,\ s2)) \\ = \quad (\text{disjoint}\,(\text{domain}\,(s1),\ \text{domain}\,(s2)) \\ \qquad \land \quad \text{mapping}\,(\text{fix-properp}\,(s1)) \\ \qquad \land \quad \text{mapping}\,(s2)) \end{array}$$

EVENT: Disable mapping.

```
;; RESTRICT and CO-RESTRICT
```

THEOREM: alistp-restrict

$\text{alistp}\,(\text{restrict}\,(s,\ r))$

THEOREM: alistp-co-restrict

$\text{alistp}\,(\text{co-restrict}\,(s,\ r))$

THEOREM: value-restrict

$$\begin{array}{l} ((a \in r) \land (a \in \text{domain}\,(s))) \\ \to \quad (\text{value}\,(a,\ \text{restrict}\,(s,\ r)) = \text{value}\,(a,\ s)) \end{array}$$

THEOREM: value-co-restrict

$$\begin{array}{l} ((a \notin r) \land (a \in \text{domain}\,(s))) \\ \to \quad (\text{value}\,(a,\ \text{co-restrict}\,(s,\ r)) = \text{value}\,(a,\ s)) \end{array}$$

THEOREM: mapping-restrict

$\text{mapping}\,(s) \to \text{mapping}\,(\text{restrict}\,(s,\ x))$

THEOREM: mapping-co-restrict

$\text{mapping}\,(s) \to \text{mapping}\,(\text{co-restrict}\,(s,\ x))$

EVENT: Disable restrict.

EVENT: Disable co-restrict.

EVENT: Make the library `"alists"`.

Index